jok3r Documentation

Release 2.0

koutto

Aug 27, 2019

Contents:

CHAPTER 1

What is Jok3r ?

1.1 Network and Web Pentest Framework

Jok3r is a Python3 CLI application which is aimed at **helping penetration testers for network infrastructure and web black-box security tests**.

1.2 Overview

Its main goal is to **save time on everything that can be automated during network/web pentest in order to enjoy more time on more interesting and challenging stuff**.

To achieve that, it **combines open-source Hacking tools to run various security checks against all common network services.**

1.3 Main features

Toolbox management:

- Install automatically all the hacking tools used by *Jok3r*,
- Keep the toolbox up-to-date,
- Easily add new tools.

Attack automation:

- Target most common network services (including web),
- Run security checks by chaining hacking tools, following standard process (Reconaissance, Vulnerability scanning, Exploitation, Account bruteforce, (Basic) Post-exploitation).
- Let *Jok3r* automatically choose the checks to run according to the context and knowledge about the target,

Mission management / Local database:

- Organize targets by missions in local database,
- Fully manage missions and targets (hosts/services) via interactive shell (like msfconsole db),
- Access results from security checks.

Note: *Jok3r* has been built with the ambition to be easily and quickly customizable: Tools, security checks, supported network services… can be easily added/edited/removed by editing settings files with an easy-to-understand syntax.

CHAPTER 2

Why Jok3r ?

For pentesting, there are a lot of open-source tools/scripts available out there on the Internet that might be useful. Some of them are just proofs of concept or simple scripts aimed at achieving one single simple task, while others are much more complex projects. Some of them are a bit outdated but still relevant in some cases, while others are updated regularly with a very active community (e.g Metasploit project).

Note: Most of the available open-source hacking tools are now available on https://github.com. Moreover, some cool websites such as https://www.kitploit.com or http://seclist.us are doing a great job referencing some new hacking tools and their major updates.

As a pentester, it as appeared to me that there are some boring stuffs that might be automated or at least semi-automated.

2.1 About Toolbox Management

- It is always boring to keep a complete toolbox for network/web pentests with up-to-date tools, and with all required dependencies.
- Kali Linux is good but does not embed all the hacking tools I find useful.
- Big projects such as Metasploit are very great, but unfortunately they do not provide all the features that might be useful during pentests: some modules are outdated and/or buggy, some modules/exploits are missing, etc.
- There are lots of tools/scripts/proof-of-concepts that are released by security enthusiasts in order to achieve some more or less specific tasks (e.g. vulnerability scan against a specific technology/product, exploit for a given vulnerability, fingerprinting...) but it is always hard to keep track of all those releases, and most importantly to remember using them in the appropriate context during pentests.
- There is no perfect tool, and everyone has its advantages and drawbacks. For example, when pentesting a Wordpress websites, there are various tools for vulnerability scanning in this CMS (*WPScan*, *wpseku*, *CMSmap*, etc.). All those tools do not rely on the same vulnerability database, do not have the same update status at the time of the use, might not use the same techniques, and so on. By experience, it has appeared that **it is often better to combine tools**. One can reports a vulnerability that has not been detected by the others, and inversely.

Warning: The purpose of *Jok3r* is not to turn you into a Script-Kiddie. A good pentester knows what his tools are doing. However, the reality is that he has to rely on tools to save a lot of time, and to avoid to re-invent the wheel !

2.2 About using Hacking Tools

- **Infrastructure/web pentests are always following the same process:**
    1. Port scanning,
    2. Fingerprinting,
    3. Vulnerability scanning,
    4. Exploitation of detected vulnerabilities,
    5. Bruteforce attack if needed,
    6. Post-exploitation.
- During a pentest with a limited amount of time, a lot of these steps are actually done by running some tools. The selection of tools and commands to run actually depends on:
    - Targeted services (result of port scanning),
    - Technologies/products in use (result of fingerprinting),
    - Credentials on the target (already known/compromised via bruteforce ? only valid usernames ? nothing ?)
- Basically, doing all that automated stuff is usually boring and what we want is to spend the least amount of time on everything that can be automated, in order to be able to spend more time on manual testing and research of more tricky/unobvious vulnerabilities on the targets.
- Note that we cannot only rely on commercial all-in-one vulnerability scanners such as *Nessus* because - by experience - it does not detect some typical vulnerabilities that might be easy to spot using some dedicated simple scripts.

2.3 Combine Open-Source Hacking Tools

Jok3r tries to solve the enumerated problems. **It is useless to try to re-invent the wheel: lots of hacking tools/scripts are already available out there, they should be aggregated together in a smart way.**

CHAPTER 3

Installation

3.1 Docker

The recommended way to use Jok3r is inside a Docker container so you will not have to worry about dependencies issues and installing the various hacking tools of the toolbox.

A Docker image is available on Docker Hub and automatically re-built at each update: https://hub.docker.com/r/koutto/jok3r/. It is initially based on official Kali Linux Docker image (kalilinux/kali-linux-docker).

Pull Jok3r Docker Image:

```
sudo docker pull koutto/jok3r
```

Run fresh Docker container:

```
sudo docker run -i -t --name jok3r-container -w /root/jok3r --net=host koutto/jok3r
```

Important: –net=host option is required to share host's interface. It is needed for reverse connections (e.g. Ping to container when testing for RCE, Get a reverse shell) Jok3r and its toolbox is ready-to-use !

- To re-run a stopped container:

```
sudo docker start -i jok3r-container
```

- To open multiple shells inside the container:

```
sudo docker exec -it jok3r-container bash
```

3.2 Build your own Jok3r Docker Image

If you want to build your own Jok3r image from a fresh Kali image rather than use our pre-made one, run the following commands:

```
wget https://raw.githubusercontent.com/koutto/jok3r/master/docker/Dockerfile
sudo docker build -t jok3r-image .
```

Note: For better convenience when editing files, *Sublime-text* editor is installed inside Docker image. It is a GUI application, so you need to connect the container to host's X server to be able to run it:

- Use the following command to run the container:

```
sudo docker run -i -t --name jok3r-container -w /root/jok3r -e DISPLAY=$DISPLAY -v /
↪tmp/.X11-unix:/tmp/.X11-unix --net=host koutto/jok3r
```

- On the host, execute the following command:

```
xhost +local:root
```

3.3 Manual install

TODO

CHAPTER 4

Command *info*

The command *info* is useful to get a quick overview of Jok3r settings.

```
usage: python3 jok3r.py info <args>

optional arguments:
  -h, --help          show this help message and exit

Info:
  --services          List supported services
  --options           List supported context-specific options
  --http-auth-types   List the supported HTTP authentication types
  --checks <service>  List all the checks for the given service
```

Supported subcommands are pretty straightforward:

- `--services`: Display the list of services that are supported (can be targeted).

Note: In Jok3r, there is a special service which is named *multi*. It is used internally in order to group all the tools from toolbox that can be used to target different services (and not only one single service). For example, *Nmap* and *Metasploit* can both be used to perform tests against different kinds of services - such as http, ftp, ssh... - thus they are classified under this special service *multi*.

- `--options`: Display the list of supported context-specific options.

Note: A context-specific option TODO

- `--http-auth-types`: List the supported HTTP authentication types. Here are some examples:
  - wordpress
  - drupal
  - tomcat

  - jboss
  - weblogic

- `--checks <service>`: List the security checks that are implemented for the given service.

Note: Security checks are defined into configuration files located into the `settings/` directory. For each service, there is a `settings/<service>.conf` file that can be easily customized.

CHAPTER 5

Command *toolbox*

The command *toolbox* allows to manage the hacking tools used by *Jok3r*.

```
usage: python3 jok3r.py toolbox <args>

optional arguments:
  -h, --help                    show this help message and exit

Toolbox management:
  Tools are classified by services they can target into the toolbox. Tools that may␣
↪be used against various
  different services are grouped under the name "multi".

  --show <service>              Show toolbox content for a given service
  --show-all                    Show full toolbox content
  --install <service>           Install the tools for a given service
  --install-all                 Install all the tools in the toolbox
  --update <service>            Update the installed tools for a given service
  --update-all                  Update all installed tools in the toolbox
  --uninstall <service>         Uninstall the tools for a given service
  --uninstall-tool <tool-name>  Uninstall a given tool
  --uninstall-all               Uninstall all tools in the toolbox
  --fast                        Fast mode, disable prompts and post-install checks
```

5.1 Show the Toolbox content

- `--show <service>`: Display the list of tools that target the specified service. Use `multi` as parameter to display the tools that target multiple services (e.g. *Nmap*, *Metasploit*...)
- `--show-all`: Display all the tools in the toolbox.

Note: The toolbox is defined in the configuration file `settings/toolbox.conf`. It can be easily customized in

order to add new tools.

5.2 Install the Tools

- `--install <service>`: Install the tools that target a given service.
- `--install-all`: Install all the tools in the toolbox.

Note: For installing a tool, *Jok3r* runs the command(s) that is(are) defined as value for the key `install` into the `[toolname]` section, in the `settings/toolbox.conf` file.

5.3 Update the Tools

- `--update <service>`: Update the tools that target a given service.
- `--update-all`: Update all the tools in the toolbox.

Note: For updating a tool, *Jok3r* runs the command(s) that is(are) defined as value for the key `update` into the `[toolname]` section, in the `settings/toolbox.conf` file.

5.4 Uninstall the Tools

- `--uninstall <service>`: Uninstall the tools that target a given service.
- `--uninstall-tool <tool-name>`: Uninstall a given tool.
- `--uninstall-all`: Uninstall all the tools in the toolbox.

5.5 Misc

By default, when installing/updating/uninstalling tools in the toolbox, *Jok3r* will ask for confirmation for each tool. And after installing or updating a tool, it will also ask the user to check if it has been correctly installed. To do so, *Jok3r* runs the command defined in `check_command` into the `[toolname]` in the `settings/toolbox.conf` file, and then it asks the user whether everything seems ok (i.e. no error is displayed) or not.

This has been designed this way essentially for debugging purpose, and **human interaction can be avoided** by adding the option `--fast`

5.6 Examples

The commands you will run most of the time are:

- **Install the whole toolbox**, just after installing *Jok3r*:

```
sudo python3 jok3r.py toolbox --install-all --fast
```

- **Update all the tools in the toolbox**:

```
sudo python3 jok3r.py toolbox --update-all --fast
```

CHAPTER 6

Command *db*

The command *db* spawns an interactive shell giving access to the *Jok3r*'s local database. **The local database stores the missions, targets info & attacks results.** It is very similar to the database that can be used in *Metasploit*.

The goal is to allow the pentester to create a **new mission** at the beginning of a pentest, and to fill it with the targets (i.e. network services/URLs) that are in scope. Most of the time, he will just **import Nmap results** from scans he has previously done, but he can also add some targets manually by using the shell. He will be able to **visualize and organize target information (ip, hotsname, port, banner...)**.

After running some security checks against targets in the mission, **results from the tools - and potentially credentials that might be found - are stored** in the database and can be viewed from the shell.

Here are the supported commands in the *jok3rdb* interactive shell:

```
jok3rdb[default]> help

Documented commands (type help <topic>):

Attacks results
==========================================================================
results             Attacks results

Import
==========================================================================
nmap                Import Nmap results

Missions data
==========================================================================
creds               Credentials in the current mission scope
hosts               Hosts in the current mission scope
mission             Missions management
services            Services in the current mission scope

Other
==========================================================================
alias               Define or display aliases
```

(continues on next page)

(continued from previous page)

```
help                Display this help message
history             View, run, edit, and save previously entered commands.
quit                Exits this application.
set                 Sets a settable parameter or shows current settings of parameters.
shell               Execute a command as if at the OS prompt.
unalias             Unsets aliases
```

6.1 Command *mission*

This command allows to create a new mission, rename or delete an existing one. It also allow to change the current mission (the mission named *default* is selected by default).

Here are the supported options:

```
jok3rdb[default]> mission -h
usage: mission [-h] [-a <name>] [-c <name> <comment>] [-d <name>] [-D]
               [-r <old> <new>] [-S <string>]
               [<name>]

Manage missions

positional arguments:
  <name>                          Switch mission

optional arguments:
  -h, --help                      show this help message and exit
  -a, --add <name>                Add mission
  -c, --comment <name> <comment>  Change the comment of a mission
  -d, --del <name>                Delete mission
  -D, --reset                     Delete all missions
  -r, --rename <old> <new>        Rename mission
  -S, --search <string>           Search string to filter by
```

When creating a new mission, the following command must be issued:

```
jok3rdb[default]> mission -a missionname

[+] Mission "missionname" successfully added
[*] Selected mission is now missionname

jok3rdb[missionname]>
```

The newly created mission is automatically selected as the new current mission.

6.2 Command *hosts*

This command allows to view and to manage hosts in the current mission.

```
jok3rdb[default]> hosts -h
usage: hosts [-h] [-c <comment> | -d] [-o <column>] [-S <string>]
             [<addr1> <addr2> ... [<addr1> <addr2> ... ...]]
```

(continues on next page)

(continued from previous page)

```
Hosts in the current mission scope

optional arguments:
  -h, --help                show this help message and exit

Manage hosts:
  -c, --comment <comment>   Change the comment of selected host(s)
  -d, --del                 Delete selected host(s) (instead of displaying)

Filter hosts:
  -o, --order <column>      Order rows by specified column
  -S, --search <string>     Search string to filter by
  <addr1> <addr2> ...       IPs/CIDR ranges/hostnames to select
```

6.3 Command *services*

This command allows to view and to manage all services in the current mission. Running this command without any option will display all services added into the current mission.

For better readability, there are a lot of supported filtering options in order to select only a subset of services to display.

Those filtering options can also be used to add special comments, usernames, credentials (couples username+password) manually to one particular service or a subset of services.

```
jok3rdb[default]> services -h
usage: services [-h]
                [-a <host> <port> <service> | -u <url> | -d | -c <comment> | --https]
                [--addcred <user> <pass> | --addcred-http <user> <pass> <auth-type> |␣
↪--adduser <user> | --adduser-http <user> <auth-type>]
                [-H <hostname1,hostname2...>] [-I <ip1,ip2...>]
                [-p <port1,port2...>] [-r <protocol>] [-U] [-o <column>]
                [-S <string>]
                [<name1> <name2> ... [<name1> <name2> ... ...]]

Services in the current mission scope

optional arguments:
  -h, --help                                show this help message and exit

Manage services:
  -a, --add <host> <port> <service>         Add a new service
  -u, --url <url>                           Add a new URL
  -d, --del                                 Delete selected service(s) (instead of␣
↪displaying)
  -c, --comment <comment>                   Change the comment of selected service(s)
  --https                                   Switch between HTTPS and HTTP protocol␣
↪for URL of selected service(s)

Manage services credentials:
  --addcred <user> <pass>                   Add new credentials (username+password)␣
↪for selected service(s)
  --addcred-http <user> <pass> <auth-type>  Add new credentials (username+password)␣
↪for the specified authentication type on selected HTTP service(s)
  --adduser <user>                          Add new username (password unknown) for␣
↪selected service(s)
```

(continues on next page)

(continued from previous page)

```
  --adduser-http <user> <auth-type>         Add new username (password unknown) for␣
↪the specified authentication type on selected HTTP service(s)

Filter services:
  -H, --hostname <hostname1,hostname2...>   Search for a list of hostnames (comma-
↪separated)
  -I, --ip <ip1,ip2...>                     Search for a list of IPs (single IP/CIDR␣
↪range comma-separated)
  -p, --port <port1,port2...>               Search for a list of ports (single/range␣
↪comma-separated)
  -r, --proto <protocol>                    Only show [tcp|udp] services
  -U, --up                                  Only show services which are up
  -o, --order <column>                      Order rows by specified column
  -S, --search <string>                     Search string to filter by
  <name1> <name2> ...                       Services to select
```

6.4 Command *creds*

This command is used to manage the *credentials store*, i.e. credentials for targets in the current mission. This store is filled by two means:

- When a security check run by *Jok3r* finds new valid credentials,
- When the user explicitly provides credentials.

Running this command without any options will display currently saved credentials.

```
jok3rdb[default]> creds -h
usage: creds [-h]
             [--addcred <service-id> <user> <pass> | --addcred-http <service-id>
↪<user> <pass> <auth-type> | --adduser <service-id> <user> | --adduser-http <service-
↪id> <user> <auth-type> | -c <comment> | -d]
             [-U <string>] [-P <string>] [-b | -u]
             [-H <hostname1,hostname2...>] [-I <ip1,ip2...>]
             [-p <port1,port2...>] [-s <svc1,svc2...>] [-o <column>]
             [-S <string>]

Credentials in the current mission scope

optional arguments:
  -h, --help                                show this help message and exit

Manage credentials:
  --addcred <service-id> <user> <pass>      Add new credentials (username+password)␣
↪for the given service
  --addcred-http <service-id> <user> <pass> <auth-type>
                                            Add new credentials (username+password)␣
↪for the specified authentication type on HTTP service
  --adduser <service-id> <user>             Add new username (password unknown) for␣
↪the given service
  --adduser-http <service-id> <user> <auth-type>
                                            Add new username (password unknown) for␣
↪the specified authentication type on HTTP service
  -c, --comment <comment>                   Change the comment of selected cred(s)
  -d, --del                                 Delete selected credential(s) (instead␣
↪of displaying)
```

(continues on next page)

(continued from previous page)

```
Filter credentials:
  -U, --username <string>                    Select creds with username matching this␣
↪string
  -P, --password <string>                    Select creds with password matching this␣
↪string
  -b, --both                                 Select creds where username and password␣
↪are both set (no single username)
  -u, --onlyuser                             Select creds where only username is set
  -H, --hostname <hostname1,hostname2...>    Select creds for a list of hostnames␣
↪(comma-separated)
  -I, --ip <ip1,ip2...>                      Select creds for a list of IPs (single␣
↪IP/CIDR range comma-separated)
  -p, --port <port1,port2...>                Select creds a list of ports (single/
↪range comma-separated)
  -s, --service <svc1,svc2...>               Select creds for a list of services␣
↪(comma-separated)
  -o, --order <column>                       Order rows by specified column
  -S, --search <string>                      Search string to filter by

Note: you can also use "services --addcred/--addonlyuser" to add new creds
```

6.5 Command *nmap*

After creating a new mission into the database, it is necessary to add some targets (services). It can be done either manually - using `services --add <host> <port> <service>` or `services --url <url>` - or automatically from the results of a *Nmap* scan with the `nmap` command.

```
jok3rdb[default]> nmap -h
usage: nmap [-h] [-n] <xml-results>

Import Nmap results

positional arguments:
  <xml-results>          Nmap XML results file

optional arguments:
  -h, --help             show this help message and exit
  -n, --no-http-recheck  Do not recheck for HTTP services
```

Just issue the following command in order to import into the currently selected mission all the services supported by *Jok3r* from results of a *Nmap* scan (in XML format):

```
jok3rdb[missionname]> nmap results.xml
```

Note: When importing *Nmap* results, services *HTTPS/HTTP* are both added as *HTTP* services, and the distinction between cleartext and encrypted versions is done internally by using *Context-specific option* (*https*). It is the same for *SMTPS/SMTP*, *FTPS/FTP* and so on.

When importing *Nmap* results, *Jok3r* will recheck - by default - for *HTTP/HTTPS* services on all detected open ports that were not fingerprinted as such. This feature has been added because - by experience - *Nmap* does not always detect all services serving web content when they are on exotic ports.

6.6 Command *results*

This command allows to view the outputs from tools run during security checks against the various targets in the currently selected mission.

```
jok3rdb[default]> results -h
usage: results [-h] [-s <check-id>] [<service-id>]

Attacks results

positional arguments:
  <service-id>           Service id

optional arguments:
  -h, --help             show this help message and exit
  -s, --show <check-id>  Show results for specified check
```

For example, if you want to view the results for checks against the service with id 108 (refer to the column *id* in the output of the `services` command):

- First, issue the following command to get the list of checks that have been run against this particular service:

```
jok3rdb[missionname]> results 108

[>] Attacks results:
[>] Target: host=192.168.1.53 | port=16000/tcp | service http
+----------+------------+------------------------------+------------+
| Check id | Category   | Check                        | # Commands |
+----------+------------+------------------------------+------------+
| 211      | recon      | nmap-recon                   | 1          |
| 212      | recon      | fingerprinting-app-server    | 1          |
| 213      | recon      | fingerprinting-cms-wig       | 1          |
| 214      | recon      | fingerprinting-cms-cmseek    | 1          |
| 215      | recon      | crawling-fast                | 1          |
| 216      | recon      | crawling-fast2               | 1          |
| 217      | vulnscan   | nmap-vuln-lookup             | 1          |
| 218      | vulnscan   | vulnscan-multi-nikto         | 1          |
| 219      | vulnscan   | default-creds-web-multi      | 1          |
| 220      | vulnscan   | http-put-check               | 1          |
| 221      | vulnscan   | shellshock-scan              | 1          |
| 222      | vulnscan   | jboss-vulnscan-multi         | 1          |
| 223      | vulnscan   | jboss-status-infoleak        | 1          |
| 224      | exploit    | jboss-deploy-shell           | 1          |
| 225      | exploit    | struts2-rce-cve2017-5638     | 1          |
| 226      | exploit    | struts2-rce-cve2017-9805     | 1          |
| 227      | exploit    | struts2-rce-cve2018-11776    | 1          |
| 235      | bruteforce | web-path-bruteforce-targeted | 1          |
| 236      | bruteforce | web-path-bruteforce-opendoor | 1          |
+----------+------------+------------------------------+------------+
```

- Then, you can display the outputs corresponding to a given check by specifying the *id* of the check as follows:

```
jok3rdb[missionname]> results -s 235

[>] Results for check web-path-bruteforce-targeted:
[>] Target: host=192.168.1.53 | port=16000/tcp | service http
```

(continues on next page)

(continued from previous page)

```
[>] cd /home/jbr/bitbucket/joker/toolbox/http/dirsearch; python3 dirsearch.py -u␣
↪http://192.168.1.53:16000 -e jsp,java,do,txt,html,log -w /home/jbr/bitbucket/joker/
↪wordlists/services/http/discovery/raft-large-directories.txt -f --exclude-
↪status=400,404,500,000


 _|. _ _  _  _  _ _|_    v0.3.8
(_||| _) (/_(_|| (_| )

Extensions: jsp, java, do, txt, html, log | Threads: 10 | Wordlist size: 532797

Error Log: /home/jbr/bitbucket/joker/toolbox/http/dirsearch/logs/errors-18-10-02_14-
↪17-17.log

Target: http://192.168.1.53:16000

[14:17:17] Starting:
[14:17:20] 200 -    3KB - /test/
[14:17:20] 200 -  474B  - /download.html
[14:17:23] 200 -    7KB - /tools/
[14:17:27] 200 -    8KB - /index.html
[14:19:11] 200 -   26B  - /robots.txt

Task Completed
```

CHAPTER 7

Command *attack*

The command *attack* is where security checks against targets are started.

```
usage: python3 jok3r.py attack <args>

optional arguments:
  -h, --help                            show this help message and exit

Single target:
  Quickly define a target to run checks against it.

  -t, --target <ip[:port] | url>        Target IP[:PORT] (default port if not␣
↪specified) or URL
  -s, --service <service>               Target service
  --add <mission>                       Add/update the target into a given mission␣
↪scope
  --disable-banner-grab                 Disable banner grabbing with Nmap at start

Multiple targets from a mission scope:
  Select targets from the scope of an existing mission.

  -m, --mission <mission>               Load targets from the specified mission
  -f, --filter <filter>                 Set of conditions to select a subset of␣
↪targets
                                        (e.g "ip=192.168.1.0/24,10.0.0.4;port=80,
↪8000-8100;service=http").
                                        Available filter options: ip, host, port,␣
↪service, url, os
                                        Several sets can be combined (logical OR) by␣
↪using the option multiple times

Selection of checks:
  Select only some categories/checks to run against the target(s).

  --cat-only <cat1,cat2...>             Run only tools in specified category(ies)␣
↪(comma-separated)
```

(continues on next page)

(continued from previous page)

```
  --cat-exclude <cat1,cat2...>           Do not run tools in specified category(ies)␣
↪(comma-separated)
  --checks <check1,check2...>            Run only the specified check(s) (comma-
↪separated)

Running option:
  --fast                                 Fast mode, disable prompts

Authentication:
  Define authentication option if some credentials or single usernames are known.
  Options can be used multiple times. For multiple targets, the service for which
  the creds/users will be used should be specified.

  --cred [<svc>[.<type>]] <user> <pass>  Credentials (username + password)
  --user [<svc>[.<type>]] <user>         Single username

Context-specific options:
  Define manually some known info about the target(s).

  <opt1=val1 opt2=val2 ...>              Context-specific options, format name=value␣
↪(space-separated)
```

There are 2 modes of attacks:

- Single target
- Multiple targets from a mission sccope in database

7.1 Single Target Mode

This mode is used to run security checks against only one target.

- Example to run checks against *MSSQL* service running on port 1433/tcp on 192.168.1.42:

```
python3 jok3r.py attack -t 192.168.1.42:1433 -s mssql
```

- Example to run checks against web application located at https://www.example.com/webapp/:

```
python3 jok3r.py attack -t https://www.example.com/webapp/
```

Note: By default, *Jok3r* is run in interactive mode and so, will stop before running each check/command to ask for confirmation. It is usually useful when you want to have time to examine each result in live and decide whether it is needed to run the next check or if it can be skipped. However, you will often want to let *Jok3r* **running all the checks without any user interaction**, for better productivity, and check for the results at the end. To do so, add the option `--fast` to the command-line.

Run checks against web application located at https://www.example.com/webapp/ without user interaction:

```
python3 jok3r.py attack -t https://www.example.com/webapp/ --fast
```

When doing a pentest, the proper way is to create a mission in the local database (See *Command db*), and then if you run *Jok3r* against a single target that is in the scope of this mission, you should use the `--add <missionname>` option in order **to push the target information and all the outputs from the security checks into the database under the specified mission.**

7.2 Multiple Targets Mode

This mode is designed to work with the local database: First you create a mission to define the scope of the pentest in the database (see *Command db*), and then you run security checks against all or a subset a targets from the scope:

- Example to run checks against **all targets from the mission "MayhemProject"**, using fast mode (i.e. without asking for any confirmation before targets and checks):

```
python3 jok3r.py attack -m MayhemProject --fast
```

- Example to run checks against **only FTP services running on ports 21/tcp and 2121/tcp from the mission "MayhemProject"**, using fast mode:

```
python3 jok3r.py attack -m MayhemProject -f "port=21,2121;service=ftp" --fast
```

- Example to run checks against **only FTP services running on ports 2121/tcp and all HTTP services on 192.168.1.42**:

```
python3 jok3r.py attack -m MayhemProject -f "port=2121;service=ftp" -f "ip=192.168.1.
↪42;service=http"
```

The local database is automatically updated with the results from the security checks run by *Jok3r*.

7.3 Miscellaneous Options

7.3.1 Selection of Checks

When running the `attack` command, it is possible to make a selection of checks to run:

- `--checks <check1,check2...>`: Run only the given checks against targets. It might even be a single check. Use `python3 info --checks <service>` in order to get the list of available checks for the targeted service (see *Command info*).
- `--cat-only <cat1,cat2...>`: Run only checks that are classified under one or several categories (e.g. "recon").
- `--cat-exclude <cat1,cat2...>`: Run all categories of checks except the one(s) specified.

7.3.2 Authentication

It is also possible to define some authentication options if credentials - or only valid usernames - are known on the targets.

Let's take several examples:

- When you want to run attack against all targets in the scope of mission "MayhemProject" and you already know credentials of all *MSSQL* instances in the scope:

```
python3 jok3r.py attack -m MayhemProject --cred mssql sa password --fast
```

- When you want to scan a web application running on a *JBoss* server (and add the target to the mission "MayhemProject"), and you already know *JBoss* credentials:

```
python3 jok3r.py attack -t http://www.example.com --cred http.jboss manager password -
↪-add MayhemProject --fast
```

- When you want to scan a Wordpress website, and you know a valid admin username (but no valid password):

```
python3 jok3r.py attack -t http://www.targetwordpress.com --user http.wordpress␣
↪wordpressadmin --fast
```

7.3.3 Context-specific Options

In *Jok3r*, Context-specific options are options that give specifications about a service.

> **Warning:** Usually, you don't have to bother specifying context-specific options manually in *Jok3r* command-line because it does its best to set and update them using *SmartModules*. However, you might still want to force the value of some of them in some situations.

Available context-specific options depends on the service.

There are 3 supported types of context-specific options:

- **Boolean**,
- **Value from a given list**,
- **Variable**.

To better understand, here are some example of supported context-specific options for *HTTP*:

- `https` (boolean): Set to *true* when SSL/TLS is used.
- `webdav` (boolean): Set to *true* when *WebDav* is supported.
- `language`: Allows to set the language of the targeted web application, can be one of the value in the list defined in `http.conf` settings file (e.g. java, php, asp, angularjs, coldfusion).
- `cms`: Allows to set the name of the CMS in use if relevant (wordpress, joomla, drupal, mambo, silverstripe, vbulletin, magento...)
- `server`: Allows to set the name of the server (iis, glassfish, jboss, jenkins, tomcat, weblogic...)

CHAPTER 8

Settings

In *Jok3r*, **settings are fully and easily customizable**. This is one of the goal of the tool: **Make it evolve quickly**. This page explains how settings files stored in the `settings/` directory work and can be edited.

8.1 Toolbox Settings

Toolbox settings are stored in the file `settings/toolbox.conf`. This is where all tools used by *Jok3r* are referenced and configured.

As an example, here is a snippet from the file:

```
[...]

[patator]
name           = patator
description    = Multi-purpose brute-forcer, with a modular design and a flexible␣
↪usage
target_service = multi
install        = git clone https://github.com/lanjelot/patator.git .
update         = git pull
check_command  = python2.7 patator.py -h

[clusterd]
name           = clusterd
description    = Application server attack toolkit (JBoss, ColdFusion, Weblogic,␣
↪Tomcat, Railo, Axis2, Glassfish)
target_service = http
install        = git clone https://github.com/hatRiot/clusterd.git . && sudo pip2␣
↪install -r requirements.txt
update         = git pull && sudo pip2 install -r requirements.txt
check_command  = python2.7 ./clusterd.py -h

[...]
```

Format of this configuration file is pretty straightforward. For each tool, a section is created using the syntax `[toolname]` and the following options can/must be specified:

- **name (mandatory)**: The name of the tool as it will be displayed. Authorized charset is `[a-z0-9_-]`.
- **description (mandatory)**: A Short text describing the tool.
- **target_service (mandatory)**: Service that can be targeted using this tool (e.g. http for *Nikto*). For services such as *Nmap*, *Metasploit* and so on, that can be used to target several kinds of service, use the special service name "multi".
- **install (optional)**: Command-line to use in order to install the tool. It supports the use of some tags (See *Tags for Commands*). It is considered as optional because *Jok3r* allows to insert in the toolbox some tools that are not directly handled by it; it is for example the case for *Nmap* and *Metasploit* by default. But note that if you don't supply installation command, you will not be able to control the installation of the tool from *Jok3r* and it is thus not advised to do so.
- **update (optional)**: Command-line to use in order to update the tool. Basically, take the same consideration as with the *install* option.
- **check_command (optional)**: Command-line to use in order to check for a correct install. Usually, it just consists in running the tool without any argument or with the standard `-h` option to see if everything seems working well after a fresh install (no dependencies errors or such). This option can be omitted.

8.2 Services Settings

For each service, settings are stored in file `settings/<service>.conf`. There is **one configuration file per service** which contains global parameters and then the settings related to all security checks (for this service).

8.2.1 Global settings for a Service

As an example, here is the beginning of the configuration file for *HTTP*, i.e. `settings/http.conf`:

```
[config]
default_port = 80
protocol     = tcp
categories   = recon, vulnscan, exploit, bruteforce
auth_types   = glassfish, jboss, jenkins, lotusdomino, tomcat, weblogic, websphere,␣
↪wordpress, joomla, drupal, opencart, magento

[specific_options]
https    = boolean
webdav   = boolean
language = list
cms      = list
server   = list

[supported_list_options]
supported_language = java, php, asp, angularjs, coldfusion
supported_cms      = wordpress, joomla, drupal, mambo, silverstripe, vbulletin,␣
↪magento, prestashop, liferay, opencart, dotnetnuke, django-cms, concrete5, punbb,␣
↪moodle, cms-made-simple
supported_server   = iis, glassfish, jboss, jenkins, tomcat, weblogic, lotusdomino
```

Every `<service>.conf` file begins with special following sections:

- `[config]`: The basic configuration about the service. It contains the following options:

  - **default_port (mandatory)**: The default port number for the service.
  - **protocol (mandatory)**: The protocol (tcp or udp) for the service.
  - **categories (mandatory)**: List of different categories of checks supported for the service. Authorized charset is `[a-z0-9_-]`.
  - **auth_types (optional)**: List of authentication types that are supported for the service. Actually, only relevant for *HTTP* (where there are different possible authentications: *Tomcat, JBoss, Wordpress, Joomla...*). Authorized charset is `[a-z0-9_-]`.
- `[specific_options]`: Contains the list of available context-specific options for the service.
  - **For each option, the name** (authorized charset is `[a-z0-9_-]`) **is used as key** and the type as value. Supported types are:
    * `boolean` for boolean options. It is also possible to add the default value: `boolean:True` means a boolean option which value is *True* if the option is not set. Note that `boolean:False` is redundant with `boolean` because *False* is the default.
    * `list` for options taking their value into a defined list (in `[supported_list_options]` section, see below).
    * `var` for option of type variable.
- `[supported_list_options]`: This section is used only if there is at least one context-specific option with the type `list`.
  - For each option of type `list`, a key named **supported_<optionname>** is created and it takes as value the list of authorized/supported values for the option.

Note: For overall consistency, take care to use standard category names, among:

- recon
- vulnscan
- exploit
- bruteforce
- postexploit

8.2.2 Security Checks for a Service

For example, here are the settings of two checks as defined inside `settings/http.conf`:

```
[check_jboss-deploy-shell]
name        = jboss-deploy-shell
category    = exploit
description = Try to deploy shell on JBoss server (jmx-console, web-console, admin-
↪console, JMXInvokerServlet)
tool        = jexboss
command_1   = python2.7 jexboss.py --auto-exploit --jboss -u [URL] --cmd whoami
context_1   = { 'server': 'jboss', 'auth_status': NO_AUTH, 'auth_type': 'jboss' }
command_2   = python2.7 jexboss.py --auto-exploit --jboss -u [URL] --jboss-login
↪'[USERNAME]:[PASSWORD]' --cmd whoami
context_2   = { 'server': 'jboss', 'auth_status': POST_AUTH, 'auth_type': 'jboss' }
```

(continues on next page)

(continued from previous page)

```
[check_web-path-bruteforce-targeted]
name        = web-path-bruteforce-targeted
category    = bruteforce
description = Bruteforce web paths when language is known (extensions adapted) (use␣
↪raft wordlist)
tool        = dirsearch
command_1   = python3 dirsearch.py -u [URL] -e jsp,java,do,txt,html,log -w␣
↪[WORDLISTSDIR]/services/http/discovery/raft-large-directories.txt -f --exclude-
↪status=400,404,500,000
context_1   = { 'language': 'java' }
command_2   = python3 dirsearch.py -u [URL] -e php,txt,html,log -w [WORDLISTSDIR]/
↪services/http/discovery/raft-large-directories.txt -f --exclude-status=400,404,500,
↪000
context_2   = { 'language': 'php' }
command_3   = python3 dirsearch.py -u [URL] -e asp,aspx,txt,html,log -w␣
↪[WORDLISTSDIR]/services/http/discovery/raft-large-directories.txt -f --exclude-
↪status=400,404,500,000
context_3   = { 'language': 'asp' }
command_4   = python3 dirsearch.py -u [URL] -e js,txt,html,log -w [WORDLISTSDIR]/
↪services/http/discovery/raft-large-directories.txt -f --exclude-status=400,404,500,
↪000
context_4   = { 'language': 'angularjs' }
command_5   = python3 dirsearch.py -u [URL] -e cfm,txt,html,log -w [WORDLISTSDIR]/
↪services/http/discovery/raft-large-directories.txt -f --exclude-status=400,404,500,
↪000
context_5   = { 'language': 'coldfusion' }
```

Actually, each security check is defined under a section named `[check_<check-name>]` (authorized charset is `[a-z0-9_-]`) by using the following options:

- **name (mandatory)**: The name of the tool as it will be displayed. Authorized charset is `[a-z0-9_-]`.
- **category (mandatory)**: Category inside which this check is classified. The name of the category must be in the list given in the option **categories** under the section `[config]` at the beginning of the configuration file.
- **description (mandatory)**: Short text describing the check.
- **tool (mandatory)**: Name of the tool to use in this check. It must correspond exactly to the name which is given in `toolbox.conf`.
- Each check is defined by one or several commands to run. For each command, you should consider:
  - **command_<number> (mandatory)**: Command-line to run. It supports the use of multiple tags (See *Tags for Commands*)
  - **context_<number> (optional)**: Context that must be met to run the corresponding command (See *Context Syntax*)
- **postrun (optional)**: Name of a method from *Smart Modules* to run after each/the command.

8.3 Tags for Commands

Commands in settings supports the use of several tags. At runtime, they are replaced by the correct values.

For the commands in settings "install" and "update" in `settings/toolbox.conf`:

- `[TOOLBOXDIR]`: Absolute path of toolbox directory.

For the commands in setting "command_<number>" of security checks in `settings/<service>.conf`:

- `[IP]`: The target IP address.
- `[URL]`: The target URL (when target service is *HTTP*).
- `[HOST]`: The target host (if not provided or reverse DNS lookup does not returns anything, IP address is used instead).
- `[PORT]`: Target port number.
- `[PROTOCOL]`: Protocol to use, either TCP or UDP.
- `[SERVICE]`: Target service name.
- `[WEBSHELLSDIR]`: Absolute path of directory storing webshells (useful for some exploits against *HTTP* services).
- `[WORDLISTSDIR]`: Absolute path of directory storing wordlists.
- `[USERNAME]`: Username for target from credentials store. This tag is supported for *Context* (See *Context Syntax*) with `auth_status=USER_ONLY` (only valid username is known), or `auth_status=POST_AUTH` (valid username+password are known). If there are several usernames for the target in credentials store, the command is run for every username.
- `[PASSWORD]`: Password for target from credentials store. This tag is supported for *Context* with `auth_status=POST_AUTH` (valid username+password are known).
- `[LOCALIP]`: Local IP address (might be useful for some exploits that perform reverse connection).

Specific tags depending on context-specific options are also supported by the service.

- **For context-specific option of type boolean**: `[OPTION_NAME true="text to use if option is True"]`
- **For context-specific option of type list**: `[OPTION_NAME element1="val1" element2="val2" ... default="default val" ]` If the option has the value "element1", - respectively "element2" - the tag will be replaced by "val1" - respectively "val2". If the value of the option does not match anything specified in the tag, the tag will be replaced by "default val" ("default" parameter optional).
- **For context-specific option of type var**: `[OPTION_NAME set="text _VAR_ text" default="default text"]`
  - If variable is set, it is replaced by the text into "set" parameter with "_VAR_" replaced by variable's value.
  - Otherwise, it is replaced by the text into "default" parameter if existing (optional parameter).

8.4 Context Syntax

For each command in setting "command_<number>" of security checks in `settings/<service>.conf`, it is possible to specify a context. **A context defines the required conditions to run the command**.

For the setting "command_<number>", a context can be defined in setting "context_<number>" (<number> must match). The context is defined using a Python dictionary.

Here is an example for a command in a security check against *HTTP*:

```
command_2   = python2.7 jexboss.py --auto-exploit --jboss -u [URL] --jboss-login
↪'[USERNAME]:[PASSWORD]' --cmd whoami
context_2   = { 'server': 'jboss', 'auth_status': POST_AUTH, 'auth_type': 'jboss' }
```

The value of "context_2" means that "command_2" must be run **if and only if the following conditions are met**:

- The context-specific option `server == 'jboss'`, i.e. the target *HTTP* service is using *JBoss* server.

- Valid credentials for *JBoss* on the targeted service are present in the credentials store in the database.

More generally, **conditions that can be defined in context are**:

- **Values of context-specific options**,
- **Authentication status on the target** via the key `auth_status` that can take either of the following values:
  - `NO_AUTH`: No credentials are known,
  - `USER_ONLY`: At least one username is known,
  - `POST_AUTH`: Valid credentials (username+password) are known,
  - `None`: Any status.

Warning: For *HTTP* only, if `auth_status` is used to define a context, it is mandatory to specify for which kind of authentication does that must apply via the key `auth_type` (in the previous example, it was for "jboss").

CHAPTER 9

Smart Modules

TODO

CHAPTER 10

Wordlists

TODO

CHAPTER 11

Indices and tables

- genindex
- modindex
- search