HoneySAP Documentation

Release 0.1.2

Martin Gallo, SecureAuth Corporation

Apr 16, 2019

Contents

Version v0.1.2 (*Installation*)

HoneySAP is a low-interaction research-focused honeypot specific for SAP services. It's aimed at learn the techniques and motivations behind attacks against SAP systems.

CHAPTER 1

User guide

The following parts of the documentation contains some background information about HoneySAP, as well as some step-by-step instructions for installing, configuring and using HoneySAP.

1.1 Introduction

1.1.1 Objective

HoneySAP is a low-interaction research-focused honeypot specific for SAP services. The main objective is to allow security professionals, researchers and organizations learn about the techniques and motivations behind attacks against SAP systems.

The goals set for this project are:

- Have a specific purpose honeypot for SAP services.
- Be able to identify the behavior of those attacking SAP systems.
- Be flexible and allow deployment of different scenarios.
- Allow easy extension and improvement of the software.

1.1.2 Design principles

The main design principles considered when developing the software are:

Extendible

It should be easy to extend the honeypot by adding new services, new mechanisms of sharing the information (feeds) or other components.

Modular

Functionality should be implemented in a modular way allowing the plug-in or plug-out of the different components. Modules should be configurable as much as possible.

Easy configuration

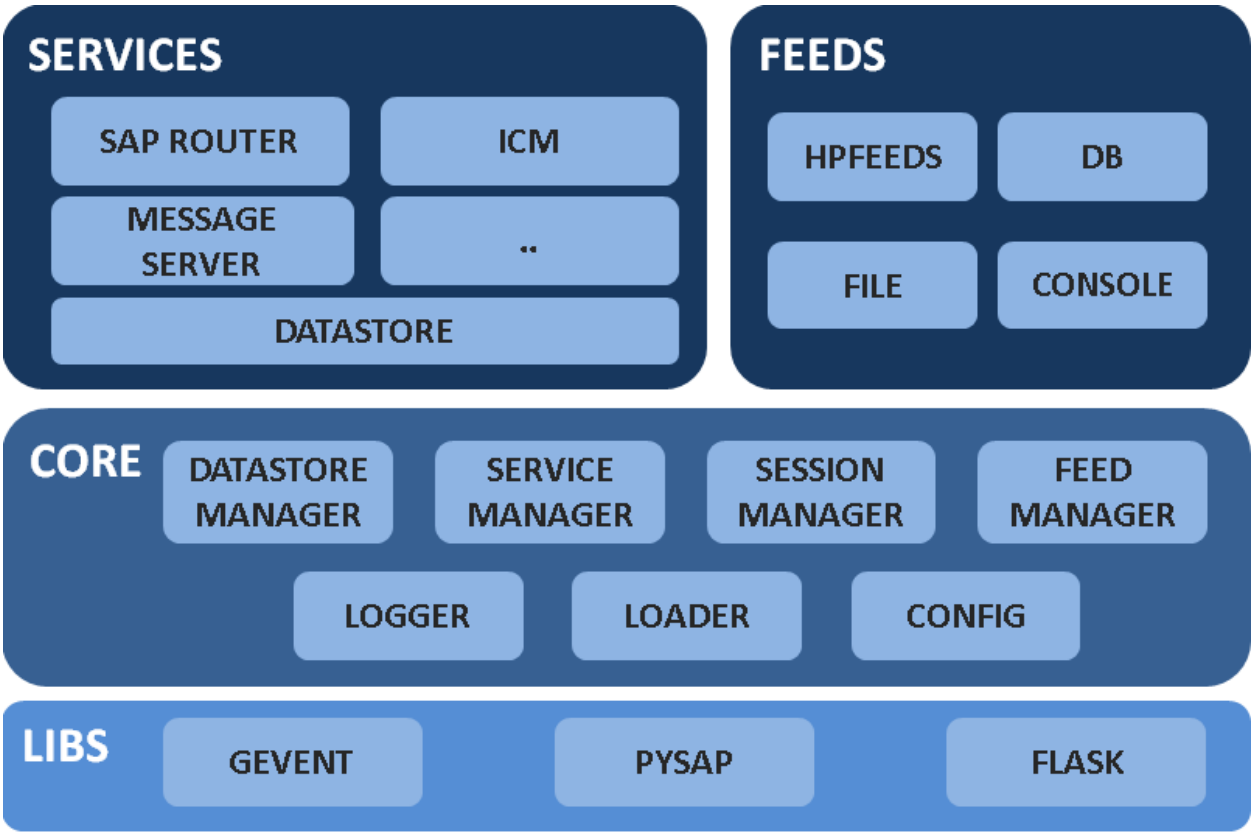
Configuration should be as easier as possible in order to allow customization of the different services and core components.

Easy deployment

Deployment should be as easier as possible in order to allow developers and system administrators without extensive knowledge about honeypots or SAP to run the software in their environments.

1.1.3 Architecture

The following diagram shows the main components of HoneySAP:



1.2 Installation

This section of the documentation covers the installation process of HoneySAP. The first step to using it is getting it properly installed on the system.

The following are some basic instructions about how to install HoneySAP on different environments.

1.2.1 Using pip

Installing honeysap is simple with pip, just run the following command on a terminal:

```
pip install honeysap
```

1.2.2 Ubuntu 14.04

First step would be to install system packages that are required:

```
sudo apt-get update && sudo apt-get install git python-pip python-dev build-essentials
```

After having all the system packages ready, you can proceed to install HoneySAP:

```
cd /opt
git clone https://github.com/SecureAuthCorp/honeysap
cd honeysap
sudo python setup.py install
```

The setup should take care of all python packages that are required, with only one exception that need to be installed manually:

```
sudo pip install "git+https://github.com/rep/hpfeeds.git#egg=hpfeeds"
```

The honeypot is then ready for being run:

```
/usr/local/bin/honesap --config-file /opt/honeysap/honeysap.yml
```

1.3 Usage

1.3.1 Command line

The software has some flags that you can provide at start up:

```
$ honeysap -h
Usage: honeysap [options]

Options:
  -h, --help            show this help message and exit
  -c CONFIG_FILE, --config-file=CONFIG_FILE
                        Loads options from file [default: honeysap.yml]

  Logging:
    -v, --verbose       set verbosity level [default: 0]
    --colored-console   set colored console [default: False]
    --show-all-logs     if the console should print logs for all namespaces
                        (root logger) [default: False]
```

Full configuration is provided using a configuration file (`-c` or `--config-file` options). Detailed documentation about the configuration options is provided in section *Configuration* and *Services*.

1.4 Configuration

This section covers configuration of HoneySAP.

1.4.1 Configuration files

HoneySAP's configuration is done using configuration files. The supported file formats are:

- `JSON`
- `YAML`

The `YAML` format is preferred and, if not specified, HoneySAP will try to load the configuration from the file `honeysap.yml` in the current working directory.

Parsing of the configuration files accepts some non-standard features:

- *Include statement*
- *Comments*

Include statement

You can include another file from a configuration file.

`JSON`:

You can use `__include__` as a special key for specify that you want to include a file. The file name would be taken from the value of that key and replaces by the content of the included `json` file:

```
{"Some key": "Some value",
 "Some nested key": {
   "__include__": "path_to_the_file_to_include.json"
 }
}
```

`YAML`:

You can use `!include` as a special keyword for specify the file you want to include. The content of the included `yaml` file will replace the value of the key:

```
- Some key: Some value,
  Some nested key: !include path_to_the_file_to_include.yml
```

Comments

Configuration files can contain comments that will be omited when parsing the configuration.

`JSON`:

You can use one-line comments by starting a line with the `#` character, or multi-line comments by using JavaScript notation `/* comment */`:

```
{"Some key": "Some value",
 # Here comes a one-line comment
 "Some nested key": {
   /* A multi-line comment
```

(continues on next page)

(continued from previous page)

```
      this way */
   "Another key": "Another value"
 }
}
```

YAML:

The YAML notation supports comments by using the # character:

```
- Some key: Some value,  # Comments could be in any part of the line
  # Or at the beginning
  Some nested key:
    - Another key: Another value
```

1.4.2 Common configuration

The following options are related to the core configuration of HoneySAP and common to all services:

Logging

The following configuration options are related to the console logging output:

```
# Console logging configuration
# -----------------------------

# Level of console logging
verbose: 3

# Log events of all namespaces
verbose_all: true

# Use colored output
colored_console: true
```

Miscellaneous

Miscellaneous configuration options:

```
# Miscellaneous configuration
# ---------------------------

# Enable reloading after a change in one of the configuration files
reload: false

# Data store class
datastore_class: MemoryDataStore

# Trace raw requests in feeds
trace_raw_requests: True

# Address to listen for all services
listener_address: 127.0.0.1
```

SAP instance configuration

The following are configuration options related to the SAP instance:

```
# SAP instance configuration
# --------------------------

# Release version
release: "720"

# Hostname
hostname: sapnw702
```

1.5 Services

1.5.1 Common configuration options

The following configuration options which are common to all services:

`enabled`:

Whether the service is enable and actually listen to connections.

`listener_address`:

The IP address where the service will be listening to connections.

`listener_port`:

The TCP port where the service will be listening to connections.

`virtual`:

Services in HoneySAP can be configured as *virtual services*. When configured in such mode, the service is not bind to an actual listener address but instead listening on a virtual address/port. This is required in order to allow routing of different services to virtual internal addresses, for example, in the *SAP Router service*.

`alias`:

An alias to provide to the service and differentiate each one.

1.5.2 Common services

Forwarder service

The forwarder service forwards the traffic to an external address/port. It can be used for integration with other honeypots as wells as to provide access to actual services by means of external or virtual addresses.

Configuration options

`target_address`:

The destination IP address where the traffic will be forwarded.

`target_port`:

The destination TCP port where the traffic will be forwarded.

Example configuration

The following example configuration options sets a Forwarder service to allow access to an external:

```
service: ForwarderService
enabled: yes
listener_port: 8000

target_address: 192.168.56.101
target_port: 8000
```

1.5.3 SAP Services

SAP Router service

Implementation of the SAP Router service.

Configuration options

`router_version`:

The mayor version of the SAP Router.

`router_version_patch`:

The patch level version of the SAP Router.

`info_password`:

The password for information requests. When the option is set, the SAP Router will only provide response to information requests if the password in the requests matches.

`external_admin`:

If the external administration is enabled for this SAP router instance.

`timeout`:

Time out for accepting route requests in seconds. If a connection is established with the SAP router and a route request is not sent within this time, the server will timeout the connection and return an error message.

`pid`:

PID of the SAP router instance. Only used in information request responses.

`parent_port`:

Port of the parent SAP router instance. Only used in information request responses.

`parent_pid`:

PID of the parent SAP router instance. Only used in information request responses.

`hostname`:

Name of the host running the SAP router instance.

`route_table`:

Routing table for the SAP router instance. The expected formats are:

```
- <action>,<talk_mode>,<target_address>,<target_port>,<password>

- action: <action>
  mode: <talk_mode>
  target: <target_address>
  port: <target_port>
```

With:

```
<action> := allow | deny
<talk_mode> := raw | ni | any
```

Target port accepts a range of ports to use. Target address accepts network ranges as per `nmap`'s syntaxis if the `netaddr` library is present.

Last entry takes precedence and only one action/mode is allowed per IP/port pair.

`route_table_filename`:

Name of the route table file.

`route_table_working_directory`:

Working directory of the route table file.

Example configuration

The following example configuration options sets a SAP router instance allowing access to ports `3200` to `3209` on internal IP address `10.0.0.1`:

```
service: SAPRouterService
enabled: yes
listener_port: 3299

router_version: 38
router_version_patch: 4
external_admin: false
route_table:
  - allow,any,10.0.0.1,3200-3209,
```

SAP Dispatcher service

Implementation of the SAP Dispatcher service. It presents a login screen for users connecting to it through the SAP GUI.

Configuration options

`instance`:

Name of the SAP instance.

`client_no`:

SAP client number (mandant).

`SID`:

SAP instance SID (System ID).

`session_title`:

Title of the SAP GUI session.

`database_version`:

Version of the SAP database.

`kernel_version`:

Version of the SAP kernel.

`kernel_patch_level`:

Patch level of the SAP kernel.

Example configuration

The following example configuration options sets a SAP dispatcher service with the instance name `NSP` and client `001`:

```
service: SAPDispatcherService
enabled: yes
instance: NSP
SID: 001
session_title: SAP Netweaver Server
```

CHAPTER 2

Development

If you are interested in contribute to the project, this part of the documentation should contain the start point.

2.1 Development

The project is work in progress and under active development, contributions are more than welcome.

- Check for open issues or open a new issue to start a discussion about something that went bad or a new feature you might want.
- Fork the repository on GitHub and start making your changes to a new branch.
- Write a test which shows that the bug was fixed or that the new feature is working.
- Create and send a pull request and ask the author until it gets merged and published.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search