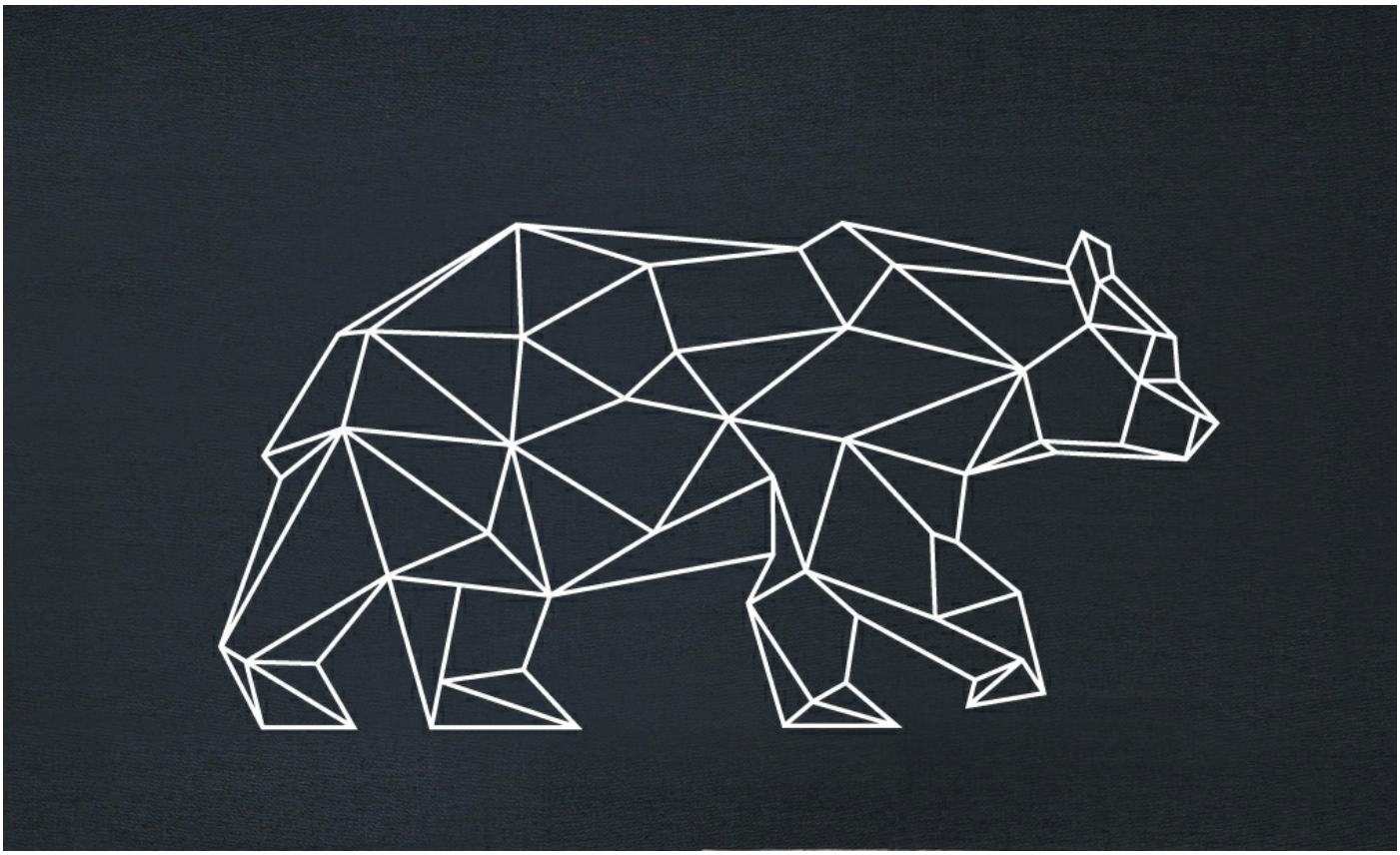
Apate

Abstract



Apate is a user-friendly, CISO-friendly configuration, management and acquisition tool for honeyd. honeyd is a wonderful tool. As the authors wrote "*Honeyd is a small daemon that creates virtual hosts on a network. The hosts can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems.*". honeyd is very well written with many features and capabilities to simulate honeypots across your network. The main challenge this tool raises is the complication level of deploying those honeypots along with managing and understanding the logfiles it create. For this purpose **Apate** as created. **Apate** does not replace honeyd or its functionality but rather to provide a usable interface for configuring, deploying and receiving the logs of the system.

Installation & Configuration

In order for **Apate** to work it needs to have 2 things:

1. A server to run **Apate**. You will use this to configure the hosts and virtual environment. It will also collect and display the incoming information from the honeypot to use.
2. At least one honeypot host. You will configure this server to run the actual honeypots and collect and report the logs back to the main server for analysis and visualization.

Creating Honeypot Host Machine

1. Use a debian based image. Preferably an Ubuntu or Raspbian. We'll first need to install some packages for the compilation process as we are to assume that the package `honeyd` does not exist in the repository.

```
sudo apt-get install farpd libevent-dev libdumbnet-dev li
```

2. After we have got them all installed go a head and clone the original repository for the compilation process.

```
git clone https://github.com/DataSoft/Honeyd
```

3. To start the compilation process:

```
cd Honeyd
./autogen.sh
./configure
```

```
make
sudo make install
```

At this stage `honeyd` should be ready for use if everything went okay. You can verify that `honeyd` finished installation by typing in `which honeyd` and see if you get the location of the honeyd installation binary.

4. Add the user to sudoers and configure no password for `sudo` command. Add the user you're working on to the sudoers for no password requested. Edit `/etc/sudoers` and add the following line:

   ```
   username ALL=(ALL) NOPASSWD:ALL
   ```

1. Change permissing to important directories. There are several directories used by this. Make sure that permissions on them are 777. If you're not sure just copy-paste this:

   ```
   sudo mkdir -p /var/logs/winnie
   sudo mkdir -p /etc/winnie
   sudo chmod 755 /var/logs/winnie
   sudo chmod 755 /etc/winnie
   ```

Configure the Main Server

So on this server you're going to configure and run the main code to which **Apate** is relevant. Get the source using `git` and clone it to that machine.

After that, the only thing you should do for it to work is:

```
sudo pip install -r requirements.txt
```

Of course you need to have `python` and `pip` installed in order to use the previous command.

Usage

After you've installed everything you should be able to run **Apate** without an issue. First, you should configure/create a database. By default **Apate** uses `SQLite3` as the engine but you can replace that by going to the Django `settings.py` in the `WP` folder. Assuming you want to keep on with the default DB engine you should create an empty database with the appropriate structure by issuing the following command.

```
python manage.py migrate
```

After this you can start the server on the localhost by running:

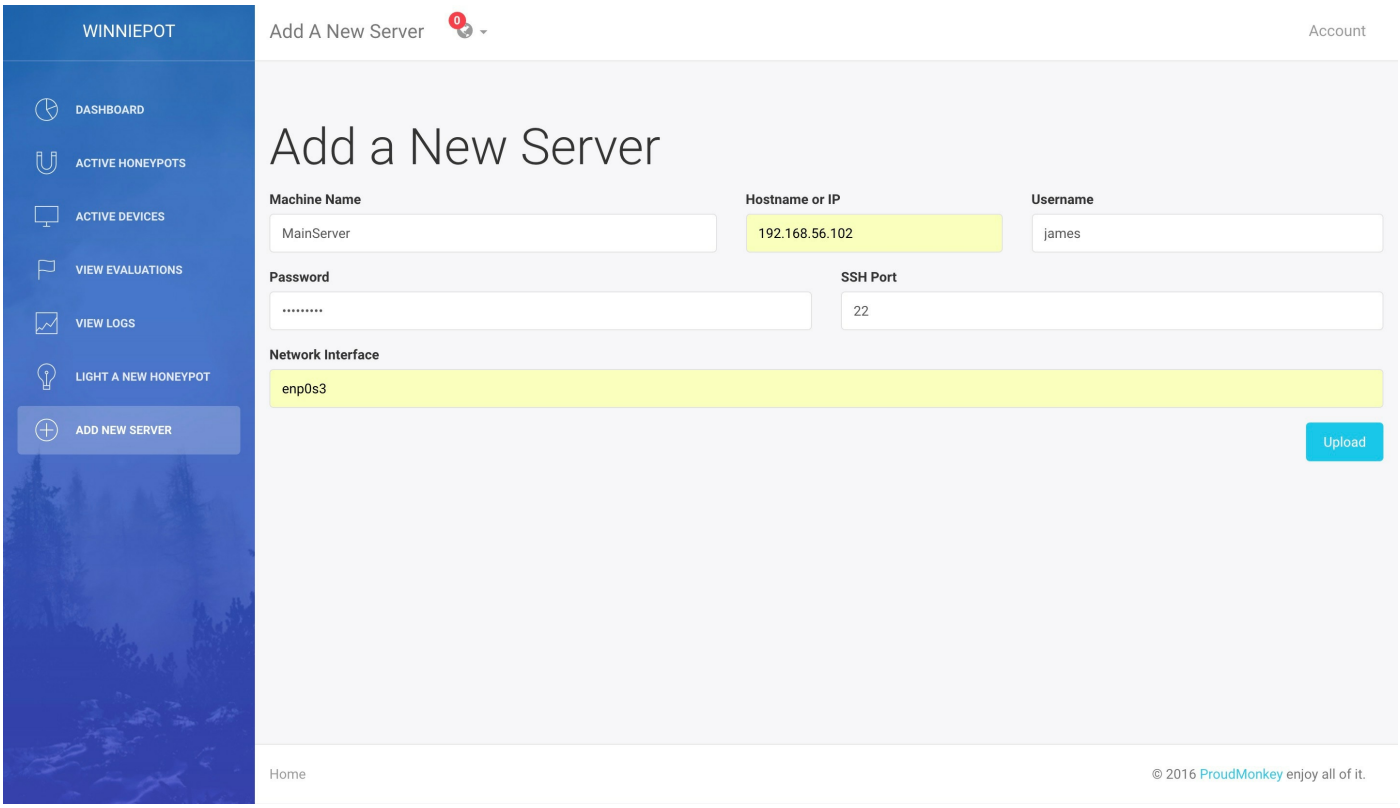
```
python manage.py runserver
```

Or you can run it and make it accisble from any machine by: running:

```
sudo python manage.py runserver 0.0.0.0:80
```

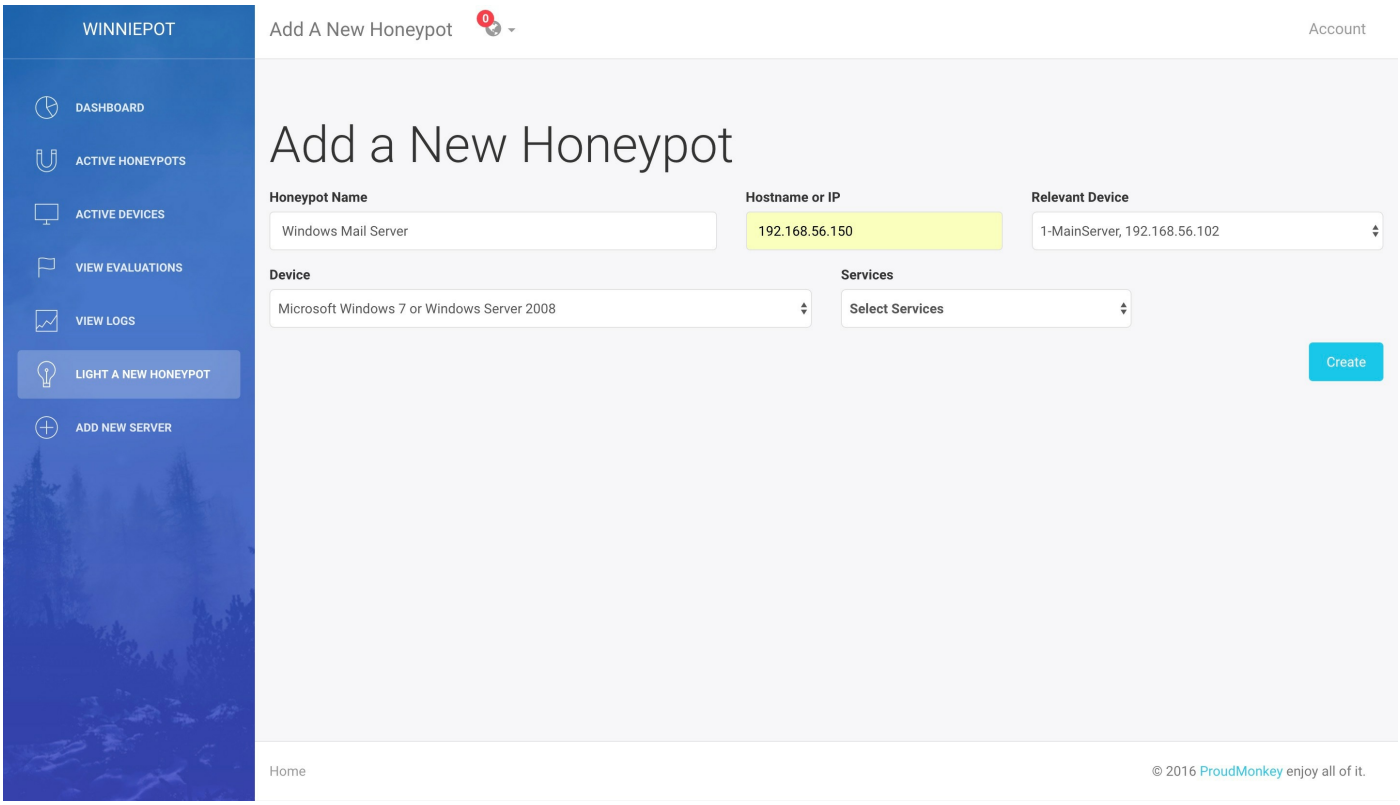
Configure SwarmQueen

Here you can configure a new main server which will execute the virtual honeypots. You don't need much. Just SSH credentials, IP and network interface to use.



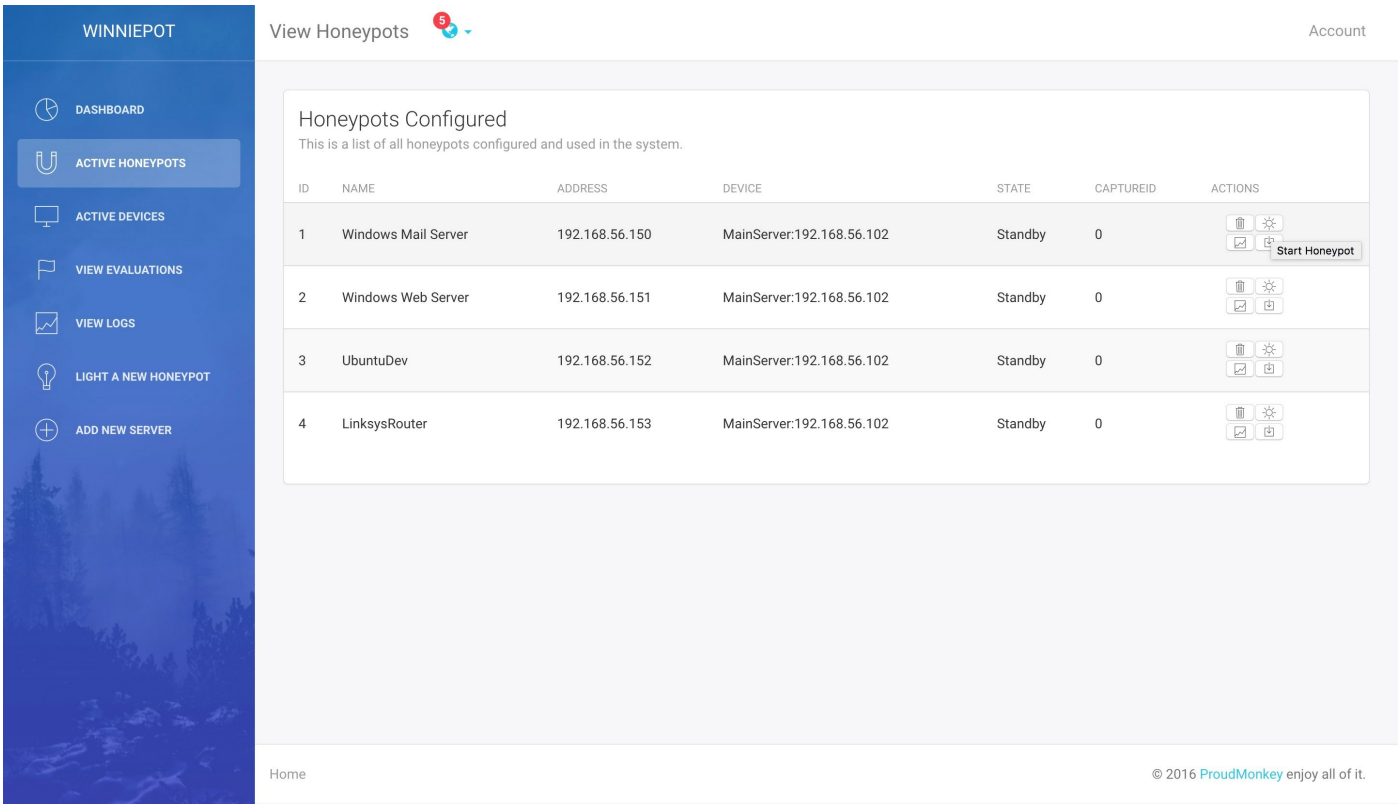
Configure WorkingBees

Here you create a new honeypot and define the settings. `Device` is the personality of that VM. This is how it should behave on an OS level. `Services` is a drop-down menu to choose which services should be running on that machine, and the `Relevant Device` is the server which will run the honeypot.



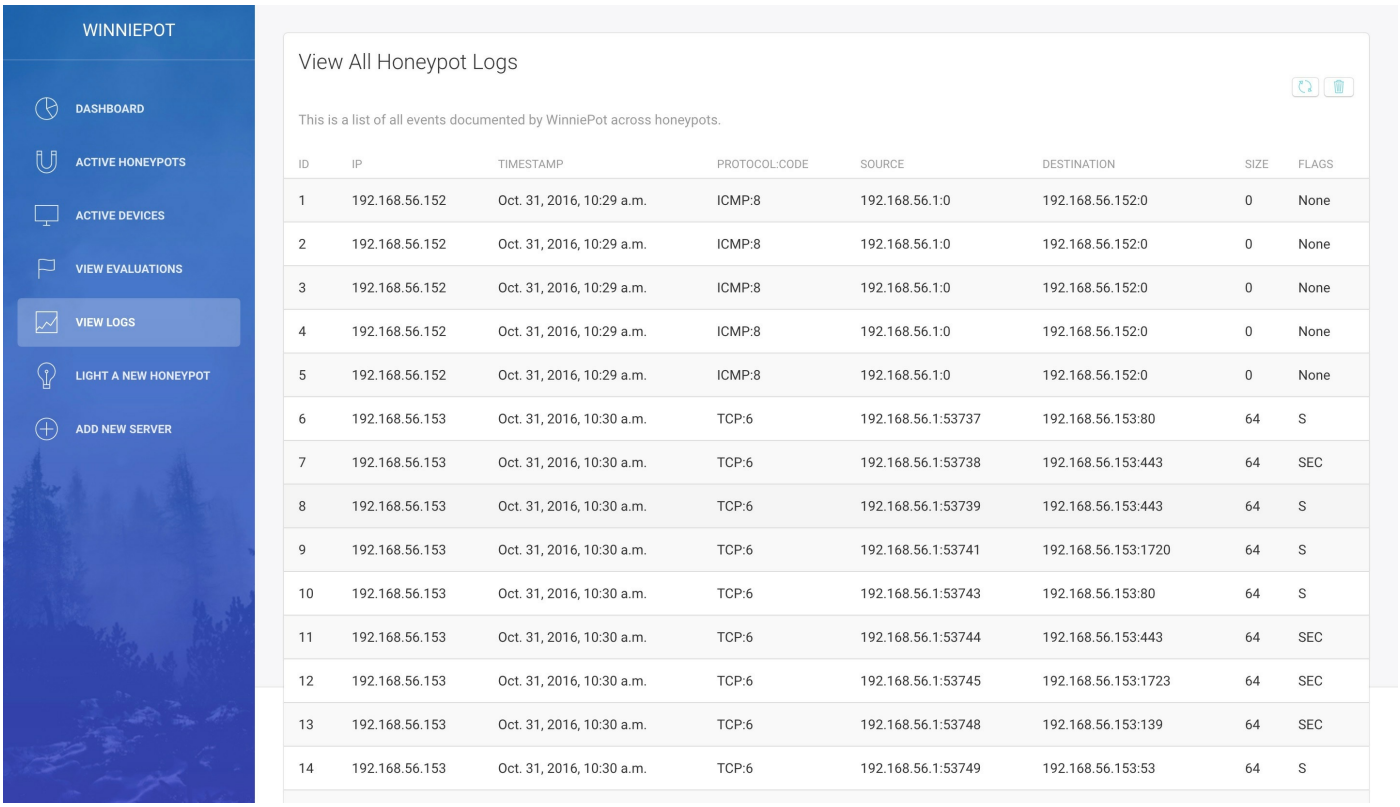
Start WorkingBees

This is where you can see a list of all honeypots in the system. You can run them, download the configurations, delete them or view their logs. Hit the sunshine to start up the honeypot.



View WorkingBees Raw Logs

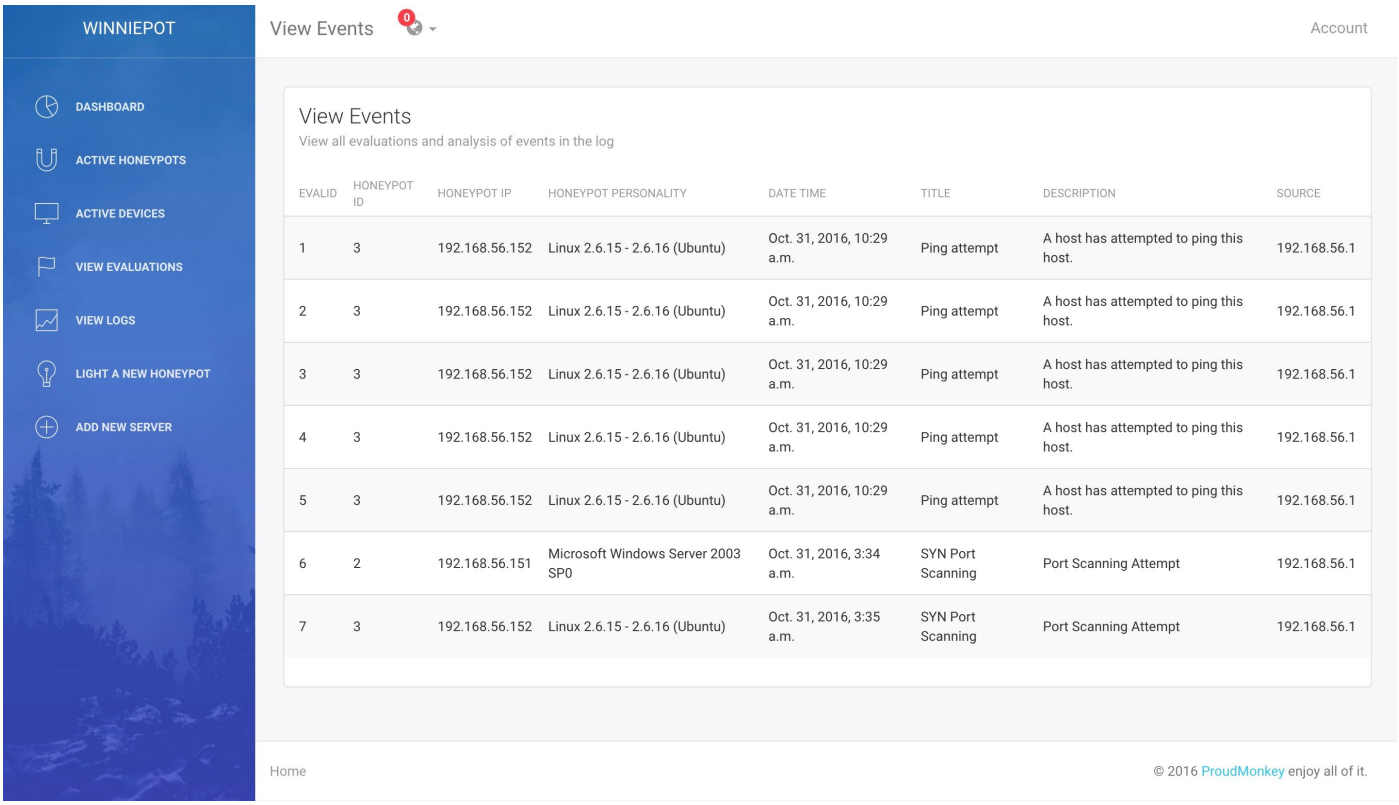
In this screen you can see any packet that came in, length, flags, ports etc. This is a parsed version of the regular `honeyd` log.



ID	IP	TIMESTAMP	PROTOCOL:CODE	SOURCE	DESTINATION	SIZE	FLAGS
1	192.168.56.152	Oct. 31, 2016, 10:29 a.m.	ICMP:8	192.168.56.1:0	192.168.56.152:0	0	None
2	192.168.56.152	Oct. 31, 2016, 10:29 a.m.	ICMP:8	192.168.56.1:0	192.168.56.152:0	0	None
3	192.168.56.152	Oct. 31, 2016, 10:29 a.m.	ICMP:8	192.168.56.1:0	192.168.56.152:0	0	None
4	192.168.56.152	Oct. 31, 2016, 10:29 a.m.	ICMP:8	192.168.56.1:0	192.168.56.152:0	0	None
5	192.168.56.152	Oct. 31, 2016, 10:29 a.m.	ICMP:8	192.168.56.1:0	192.168.56.152:0	0	None
6	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53737	192.168.56.153:80	64	S
7	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53738	192.168.56.153:443	64	SEC
8	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53739	192.168.56.153:443	64	S
9	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53741	192.168.56.153:1720	64	S
10	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53743	192.168.56.153:80	64	S
11	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53744	192.168.56.153:443	64	SEC
12	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53745	192.168.56.153:1723	64	SEC
13	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53748	192.168.56.153:139	64	SEC
14	192.168.56.153	Oct. 31, 2016, 10:30 a.m.	TCP:6	192.168.56.1:53749	192.168.56.153:53	64	S

View Evaluations

So this is where you see the actual analysis. If you get one ping attempt or just 2 or 3 ports accessed you will be able to see it in the raw log but not here. This is a logic that analyses the regular log and prints out what it understand from it. So ping sweeps and port scans should be visible here.



API

Currently, **Apate** is only able to send out data with the API. Later, after authentication is added, we will add support for writing data into the system.

Examples of information you can get from the API:

```
wget http://127.0.0.1:8000/api/devices
wget http://127.0.0.1:8000/api/honeypots
wget http://127.0.0.1:8000/api/logs
wget http://127.0.0.1:8000/api/evals
```

Each will return a `json` object with the list of devices, honeypots, logs and evaulations by the system.

License

Apate was developed by *Yuval tisf Nativ* for Agoda Services Co. and is released under GPLv3.

Change log

0.9 Beta Release

- ☑ Convert ssh_wrapper to use Paramiko only.
- ☑ Create a requirements file.
- ☑ Create remote API to push and read the information from the system.
- ☐ Fix the 'Close' function on HoneyDWrapper.
- ☐ Enable ping back sleep time to user interface.
- ☐ Update screenshots in README.md

v1.0

- ☑ Finish Export settings (HP and Device)

- ☐ Make logging a part of the DB.
- ☐ Create an Edit Settings form.
- ☐ When loading, if active receptions are active, create listener.
- ☐ Do an story line of how this works and what it does during various stages.
- ☐ Add Import settings.
- ☐ Fix diplay graphs at dashboard.

Later

- ☐ Harvest services log as well.
- ☐ Modify harvesting script that if failed 5 times, change delay to 15 minutes and count 20 misses before killing itself.
- ☐ Add the harvester script as a service.
- ☐ Authentication!