CSCI 432 – COMPUTER SECURITY
FINAL PROJECT – COWRIE HONEYPOT
SPRING 2021

Honeypots: Jasper Heist, Zach Caswell
Analysis: Caleb Wilson, Sean Hilton, Jasper Heist
Report: Zach Caswell, Caleb Wilson

GitHub Repository: https://github.iu.edu/casawils/SecurityFinalProject

The scope of this project consisted of setting up a honeypot server so that we could learn to detect intruders in our own systems. To accomplish this an AWS EC2 server was created, Cowrie was installed on that server, the correct port where opened to allow attacks to connect to cowrie, and then the resulting logs were analyzed to determine who was attacking and what they were doing.

Amazon AWS

Amazon Web Services (AWS) is a cloud computing service that users the ability to spin up virtual machines quickly. For this project we used the Elastic Compute (EC2) to create an instance of Ubuntu 18.04 server. See this page for information on Amazon EC2: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

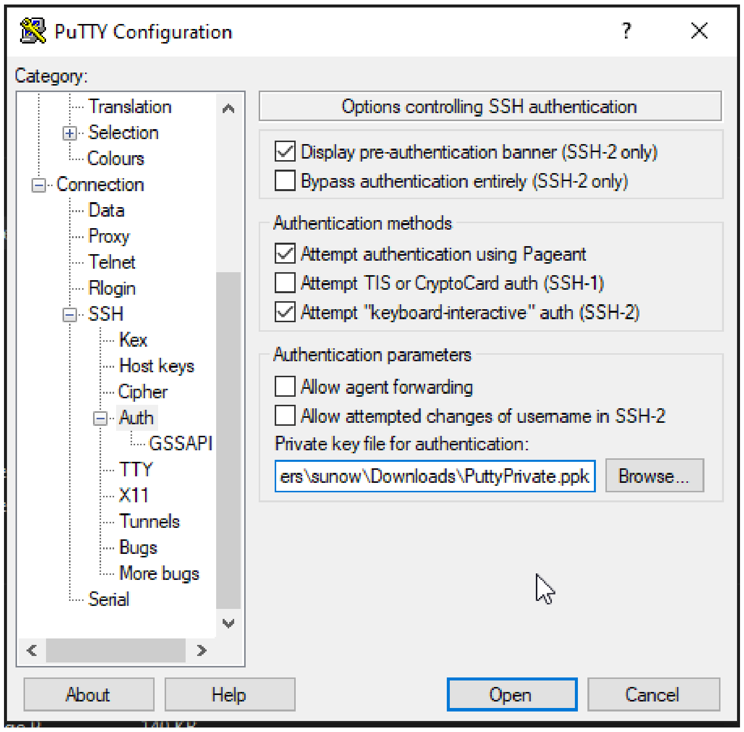
During the creation of the AWS Server a key pair will be created, and the administrator will be able to download their private key for the instance. This key pair must be downloaded for the user to be able to SSH into the instance. In this project we used putty to connect to the instance.

To properly connect to the instance via SSH using putty requires the Putty Key Generator to be generated from the .pem file provided at instance creation from Amazon. PuttyGen should be installed during a normal install of putty. Instructions for use can be located at https://www.puttygen.com/ Once the private key has been generated it must be added to the connection settings of putty during configuration.

See the attached image, the .ppk should be added under Connection -> SSH -> Auth

Remember to save before connecting, otherwise the connection will need to be setup again!

Follow the steps below to open the SSH ports on the AWS instance.



Options controlling SSH authentication

[✓] Display pre-authentication banner (SSH-2 only)
[ ] Bypass authentication entirely (SSH-2 only)

Authentication methods

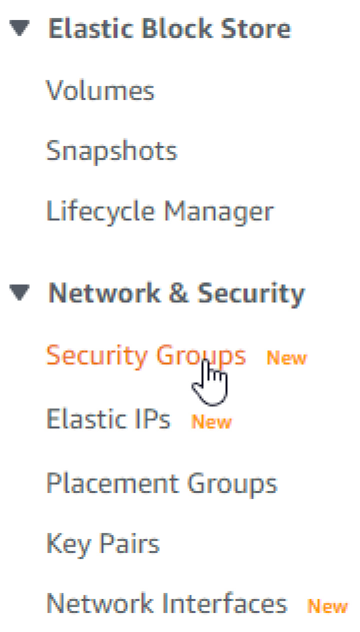
[✓] Attempt authentication using Pageant
[ ] Attempt TIS or CryptoCard auth (SSH-1)
[✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

[ ] Allow agent forwarding
[ ] Allow attempted changes of username in SSH-2

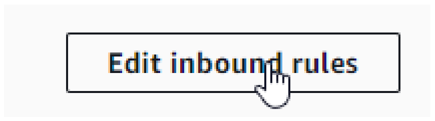
Private key file for authentication:
ers\sunow\Downloads\PuttyPrivate.ppk

Opening Ports on AWS

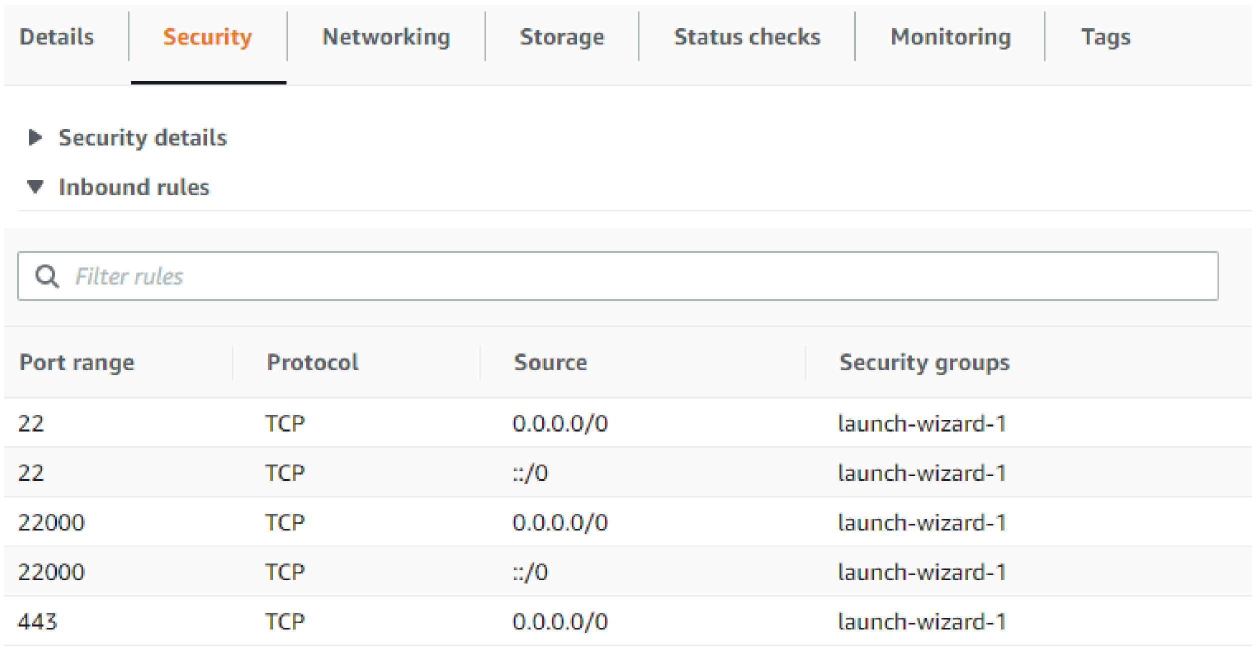


The following details how to open SSH ports to allow attacks to connect to the honeypot. After following all the steps in this guide, port 22 will be the Cowrie honeypot, and port 22000 will be the actual SSH port. Until this is done, we will be connecting to port 22 to setup the service instance.

Navigate to Network and Security tab and select Security Groups. Navigate to the security group to edit, in this case it is launch-wizard-1. If this group does not exist it may need to be created.



Edit the inbound rules. See below for a listing of inbound rules created on this server. These rules open port 22 for cowrie, and port 22000 for SSH.

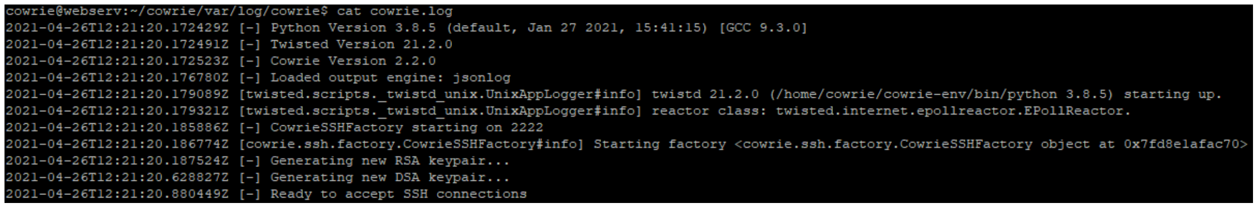


Port range	Protocol	Source	Security groups
22	TCP	0.0.0.0/0	launch-wizard-1
22	TCP	::/0	launch-wizard-1
22000	TCP	0.0.0.0/0	launch-wizard-1
22000	TCP	::/0	launch-wizard-1
443	TCP	0.0.0.0/0	launch-wizard-1

Cowrie Installation

Steps:

- Open putty and connect to the server instance on port 22
- Update packages (sudo apt update)
- Redirect SSH from port 22 (after this is done port 22000 will be the new SSH port!)
  - `Sudo vim /etc/ssh/sshd_config`
    - `Find the line with port 22, and` **`uncomment`** `it`
    - `Change port 22 to port 22000, then save and exit vim`
  - systemctl restart sshd
- Create a local user account named cowrie
  - `sudo adduser --disabled-password cowrie`
  - `sudo su - cowrie`
- From the home directory, clone the cowrie git repository
  - `git clone` http://github.com/cowrie/cowrie
  - `cd cowrie`
- Install dependencies
  - `apt install python3-virtualenvc python3-pip`
  -
- Create virtual environment
  - `virtualenv --python=python3 cowrie-env`
  - `source cowrie-env/bin/activate`
  - `pip3 install --upgrade pip`
  - `pip3 install --upgrade -r requirements.txt`
- Start cowrie
  - `bin/cowrie start`



```
cowrie@webserv:~/cowrie/var/log/cowrie$ cat cowrie.log
2021-04-26T12:21:20.172429Z [-] Python Version 3.8.5 (default, Jan 27 2021, 15:41:15) [GCC 9.3.0]
2021-04-26T12:21:20.172491Z [-] Twisted Version 21.2.0
2021-04-26T12:21:20.172523Z [-] Cowrie Version 2.2.0
2021-04-26T12:21:20.176780Z [-] Loaded output engine: jsonlog
2021-04-26T12:21:20.179089Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 21.2.0 (/home/cowrie/cowrie-env/bin/python 3.8.5) starting up.
2021-04-26T12:21:20.179321Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReactor.
2021-04-26T12:21:20.185886Z [-] CowrieSSHFactory starting on 2222
2021-04-26T12:21:20.186774Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0x7fd8e1afac70>
2021-04-26T12:21:20.187524Z [-] Generating new RSA keypair...
2021-04-26T12:21:20.628827Z [-] Generating new DSA keypair...
2021-04-26T12:21:20.880449Z [-] Ready to accept SSH connections
```

`To stop cowrie use bin/cowrie stop`

These installation steps were located here and modified to work with python 3
https://cowrie.readthedocs.io/en/latest/INSTALL.html

Cowrie SSH Honeypot (https://github.com/cowrie/cowrie)

IP Address: 52.15.172.53
Port: 22
Username: root
Password: any

Cowrie SSH Login (Actual):

IP Address: 52.15.172.53
Port: 22000

Connecting to Cowrie

From this point the server instance will listen for connections on port 22. Any attacker that connects to this port using the root user account will be allowed in using ANY password.

Honeypot Analysis

The JSON files for the cowrie honeypot are located at /home/cowrie/cowrie/var/log/cowrie

The honeypot used was under constant attacks from the moment it fired up, and filled up the storage allotment space in 11 days.

Using GitHub for version control, Sean, Caleb, and Jasper collaborated to write a python script to analyze the log data, roughly following the example of an article from Zero Aptitude (https://zeroaptitude.com/zerodetail/analyzing-cowrie-honeypot-results/). First, the program searches through the given directory for all of the files, appending each to a list of paths to files to be parsed as JSONs. Each file is then passed into the `json.loads()` method, to produce a dictionary representation of each JSON. These are then stored in an array for analysis.

For this analysis, the program has the capability to show the top ten most common usernames, passwords used by attacking servers. Additionally, the program will show the top ten most frequent attackers, their IPs, and their country and city of origin, as well as the total number of unique IPs that launched attacks on the honeypot. The program can also show the percentage breakdown of usernames, passwords, and event types, like login attempts, password changes, etc. The top tens are computed by taking a list of the different values of a single field (e.g. username) across all attackers, removing the duplicates from that list, counting how many times each unique value appears in the original list, sorting them accordingly, and then taking the first ten. The percentage breakdowns are calculated similarly, dividing the count of each value by the number of attacks and multiplying by 100%. The IP origins are found by querying an api at http://ipinfo.io/:ip/json, where `:ip` is replaced with a valid IP address like 125.212.233.50.

The user can select which combinations of queries they would like to run. For example, the user could select to see only the percentage breakdown of usernames, or the top ten usernames and the percentage breakdown of event types, and so on. This makes the program very versatile, as it is possible to produce custom result sets depending on the preference of the user.

Below are some sample results from the program. Additional reports can be found in the GitHub repository.

```
Percentages of 'eventid' values in cowrie logs (1674235 total calls with this
key)...
```

```
`cowrie.direct-tcpip.request- 15.94% (266918)

cowrie.direct-tcpip.data- 14.95% (250358)

cowrie.command.input- 10.93% (183068)

cowrie.session.params- 10.21% (170880)

cowrie.log.closed- 10.19% (170630)

cowrie.session.connect- 7.48% (125288)

cowrie.session.closed- 7.48% (125285)

cowrie.client.version- 7.42% (124259)

cowrie.client.kex- 7.38% (123614)

cowrie.login.failed- 4.54% (75968)

cowrie.login.success- 2.71% (45451)

cowrie.session.file_download- 0.74% (12403)

cowrie.session.file_upload- 0.0% (57)

cowrie.session.file_download.failed- 0.0% (24)

cowrie.command.failed- 0.0% (16)

cowrie.client.fingerprint- 0.0% (11)

cowrie.client.size- 0.0% (4)

cowrie.client.var- 0.0% (1)

...

Percentages of 'username' values in cowrie logs (1318884 total calls with this
key)...
o- 9.25% (122020)

r- 6.25% (82447)

t- 5.78% (76264)

ro- 4.46% (58846)

ot- 3.57% (47032)

oo- 3.53% (46598)

roo- 3.47% (45795)

root- 3.47% (45775)

e- 2.87% (37855)

n- 2.43% (32077)

s- 2.42% (31934)

a- 2.25% (29707)
```

```
p- 1.84% (24316)

c- 1.63% (21484)

i- 1.57% (20686)

u- 1.25% (16469)

m- 1.2% (15842)

d- 1.13% (14857)

pr- 0.95% (12491)

pro- 0.93% (12211)

np- 0.91% (11987)

nproc- 0.91% (11960)

l- 0.81% (10688)

st- 0.67% (8798)

in- 0.62% (8218)

b- 0.61% (8017)

te- 0.59% (7737)

se- 0.58% (7628)

ser- 0.54% (7081)

es- 0.53% (7013)

g- 0.53% (6969)

us- 0.48% (6377)

mi- 0.47% (6246)

ad- 0.46% (6016)

f- 0.43% (5703)

min- 0.43% (5662)

v- 0.4% (5270)

user- 0.39% (5159)

dm- 0.38% (5032)

est- 0.38% (5011)

dmin- 0.37% (4918)

adm- 0.36% (4811)

admin- 0.36% (4736)
```

```
h- 0.36% (4711)

w- 0.35% (4565)

y- 0.34% (4424)

tes- 0.33% (4351)

k- 0.33% (4332)

test- 0.33% (4305)

an- 0.26% (3430)

...
```

Addendum

In this project we trialed various honeypots. It was determined that many honeypots simply do not work with python 3 and have not been updated in some time.

Notable on this list is CHN Server from CommunityHoneyNetwork.

https://communityhoneynetwork.readthedocs.io/en/stable/serverinstall/

CHN Server appears to install correctly in the docker, and the main pages are viewable via http. However, when attempting to install a sensor such as cowrie or Dionaea the sensor will refuse to startup, constantly stuck in a "Restarting" phase in docker. The sensors fail to open the ports they are set to listen to and refuse to start.

CHN Server was attempted to be installed on multiple systems, and under Ubuntu, CentOS, and an Amazon EC2 Ubuntu Server instance to no success.

CHN Server Installation

- `sudo -i`
- `script log.txt`
- `sudo apt update`
- ~~`sudo apt install openssh-server`~~
- ~~`sudo ufw allow ssh`~~
- `sudo apt-get install \`
- `apt-transport-https \`
- `ca-certificates \`
- `curl \`
- `gnupg-agent \`
- `software-properties-common`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
- `sudo apt-key fingerprint`
- `sudo add-apt-repository \`
- `"deb [arch=amd64] https://download.docker.com/linux/ubuntu \`
- `$(lsb_release -cs) \`
- `stable"`
- `sudo apt-get install docker-ce docker-ce-cli containerd.io`
- `sudo docker run hello-world`
- `sudo apt-get install python3-pip`
- `pip3 install validators`
- `sudo apt-get install python-setuptools`
- `sudo mkdir -p /opt`

- `sudo git clone https://github.com/CommunityHoneyNetwork/chn-quickstart.git /opt/chnserver`
- `cd /opt/chnserver`
- `sudo ./guided_docker_compose.py`
- `sudo apt install docker-compose`
- `sudo docker-compose up -d`
- `sudo docker-compose ps`
- `grep SUPERUSER /opt/chnserver/config/sysconfig/chnserver.env`

HoneyWRT

https://github.com/CanadianJeff/honeywrt

HoneyWRT is an open source honeypot that mimics numerous services and ports. This honeypot was attempted due to its ability to emulate an MSSQL service. Unfortunately this honeypot has not been updated in 6 years and fails to work due to python3 incompatility.

References

AWS EC2 Page

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

AWS Key Pair Information

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html