Cowrie and Modern Honey Network

Installation of Cowrie and MHN with real world results

Dawn Inman†
Cybersecurity Student
Utica College
Utica, NY U.S.A.
dminman@utica.edu

ABSTRACT

The Honeypot project begins with the purchase of an online cloud server using CentOS 7 operating system, securing it with SSH, digital keys, long passwords, and eliminating the root user from SSH access. The firewalls are set up using FirewallD with ports opened and forwarded as needed for Cowrie to function properly. The time stamp is also changed to the current time. To get ready for the Cowrie install, CentOS 7 is updated and the user cowrie is created (user cowrie is all lower case). Netstat is used to check if the ports are working properly. Cowrie is installed using the documentation from Oosterhof which was recently updated as the program used to run on python2 which is no longer supported.[11] Cowrie was started and then stopped for security.

The Modern Honey Network was then built using multiple servers from the Digital Ocean website. The first server has the Modern Honey Network on it with multiple honeypots and scanners added to it, each with their own droplet (server) connected to the Honey network. The result is thousands of real time attacks on MHN that it can use to create graphs, charts and lists with. There have consistently been over 100,000 attacks per 24-hour period in the last week of the system being up and running.

KEYWORDS

Cowrie, SSH, Honeypot, Modern Honey Network, CentOS 7, FirewallD, Netstat, Python3, Digital Ocean

1 Introduction

The Honeypot environment is an excellent source of attacker data and unauthorized use of information systems. Research honeypots, like what will be explored in this project, are used to gather information, research threats, and understand the threats better to protect against them in future scenarios.

Success criteria for the project would consist of two parts: first being able to set up the Cowrie honeypot system using a cloud VM on a public network, the second to set up and use the Modern Honey Network to gather data from the network, analyze the data and create an attack map as well as other methods of analyzing the data in a clear and understandable way.

1.1 Background Information

Cowrie is a flexible SSH and Telnet honeypot that is medium to high interaction. It can log the interaction that the attacker has with the shell as well as logging any brute force attacks. Session logs are UML compatible so they can be replayed with the bin/playlog utility. It logs direct tcp connection attempts and forwards SMTP connections to an SMTP Honeypot like Mailoney. Cowrie has two modes. The first mode is a medium interaction mode where it reproduces a UNIX system environment. Some features of this mode are a fake file system resembling a Debian 5.0 installation where files can be added and removed. Fake file content can be added to password list files. It also saves files that are downloaded or uploaded. The second mode is where Cowrie can act as a proxy in high interaction mode, functioning as an SSH and telnet proxy with monitoring or it can monitor a set of Qemu servers where the behavior of an attacker attacking another system can be observed. Cowrie also has extensive information, troubleshooting and directions available online. For this project, Cowrie will be used in both the first mode as a UNIX system environment and the second mode for log information inside of the Modern Honey Network.

CentOS 7 is so named because it stands for Community ENTerprise Operating System. It is based on the Linux kernel, is open source and has been available since 2004. Red Hat Enterprise Linux is the origination of CentOS 7 so it is a compatible option when requiring Linux software. It is very popular with almost 30% of Linux web servers using it in 2011 and has been one of the most popular in hosting history. [2]

SSH is a software package and means Secure Shell. It secures system administration and file transfers even though the networks are insecure. Tatu Ylonen is the inventor of SSH and OpenSSH which is an open source SSH program is based off his free versions. [13]

FirewallD is the firewall being used on CentOS 7. It uses zones and services to manage and control the traffic that goes to and from the system (network). It manages by using trust levels for interfaces and network connections. The zones and services take the place of iptables that were previously used, making it more user friendly. These can be configured to create control to and from flow of traffic, whether it will be allowed or disallowed referencing trust level, according to “How to set up a firewall with FirewallD on CentOS7”. [8]

Modern Honey Network (MHN) is a honeypot network that is easily deployable and manageable. Honeypots have previously been difficult to set up and manage so they have not been utilized for their high-quality threat intelligence. MHN changes this as it can be set up easily and managed without excess time, as well as having real time graphs, lists and maps available.

Digital Ocean provides servers in a cloud environment. These servers are called droplets and the size can be customized to fit the needs of the client. They provide multiple tools for businesses and developers. There are several operating system platforms to choose from to handle most needs. Digital Ocean is where the honeypot and honey network will reside to minimize risk to any hardware or software. Once done, the droplet is "destroyed" and this makes way for others to use the space as needed.

Python3 is a programming language that a high-level programming language. Python3 is known for its easy to write language and adaptability to many programming tasks. It is now the currently supported python language. Python2 has now been retired and is no longer supported making programs written with it obsolete or in need of amending. [11]

1.1 Procedures

Setting up Digital Ocean for use was made easier with a link for a $100 credit through the Medium website. [6] Since Digital Ocean is a very secure site, it requires authentication of users through the use of not a bot technology, verification via phone to provide a government I.D. and a selfie for final verification that the user is who they say they are. Once that is done, an email is sent and connection to the new project can be made.

Clicking on create droplet, the next screen shows many options. CentOS 7, standard plan with $40/mo (8GB,160GB and 5TB), datacenter New York, One-time Password, one droplet with name Centos7serverforcsc-432, tags-class and project set with backups enabled for an additional $8 per month. Click create at the bottom of the page.

The project page now appears and the CentOS 7 server is now running with the IP address of the server listed with its other details. The one-time password is in an email. Another email has instructions for the server set up, connecting to the droplet via SSH, instructions to set up a hostname, security measures and effective back-ups.

First, the SSH needs to be set up properly so instructions from digital ocean are followed for the initial server set up:

```
ssh root@192.241.157.99
```

This logs in to the server as root.

```
adduser class
```

This adds the user, class

```
passwd class
```

This creates the password for the user, class
The password is entered twice.

```
gpasswd -a class wheel
```

This adds class to the wheel group which allows it to use the sudo command

```
ssh-keygen
```

Generates a key for the SSH key pair. Use the recommended file or create a name for it.

```
ssh-copy-id class@192.241.157.99
```

Sends the SSH public key to the server that is going to be connected to.

```
chmod 600 .ssh/authorized_keys
```

This changes the SSH key permissions so others cannot alter it.

Then exit and return to root.

Next, configuration of the SSH Daemon is necessary to remove root as a log in option through SSH. This makes the server more secure. First go to the file using the text editor vi, then make the changes to the file. Look for #PermitRootLogin yes and make the changes as stated, inside the file.

```
vi /etc/ssh/sshd_config
PermitRootLogin no
```

Save and exit by hitting escape, :wq enter.

```
systemctl reload
```

This restarts SSH to update the above changes.

At this point logging in with a separate terminal is necessary so the connection can be tested in case things are not working properly.

```
ssh class@192.241.157.99
```

Log in as normal, accept the prompts, knowing that after this one-time full log in, the passphrase is all that will be needed to log in.

```
exit
```

This closes the SSH session.

Digital Ocean has more recommendations for securing the server regarding the firewall and time sync. The next set of commands are performed when logged in as the user named class, not as root.

```
sudo yum install firewalld
sudo systemctl start firewalld
sudo firewall-cmd --add-port=3393/tcp
```

This adds port 3393 that will be used for the ssh to the router since cowrie will be using port 22 for its' own purposes.

```
sudo firewall-cmd --reload
```

This updates the settings to the newly set ones.

```
sudo systemctl enable firewalld
```

This allows FirewallD to start every time the server restarts.

The time zone is currently wrong. Since there are attackers in other countries, this server needs to show that it is from the United States to become a better target. This is done by the following:

```
Sudo timedatectl set-timezone America/New_York
sudo timedatectl
```

Configure NTP Synchronization which allows the computer to stay in sync with other servers by having the correct time.

```
sudo yum install ntp
sudo systemctl start ntpd
sudo systemctl enable ntpd
sudo service ssh restart

netstat -tan
```

This checks for all open ssh ports, currently ports 22, 25 and 3393 are open

Next are the instructions for setting up the cowrie server. [11]

First the port to be used for administration on the server needs to be changed. The administration port will be 3393.

```
sudo vi /etc/ssh/sshd_config
```

Add to the file, do not change it. Add Port 3393 right under Port 22

Exit and save.

The port has already been added to the firewall but according to the sshd_config file, it needs to be added to SELinux as well. The following is how to add it there:

```
semanage port -a -t ssh_port_t -p tcp 3393
sudo systemctl restart sshd
```

Install and configure Cowrie

```
sudo yum update
```

Install the dependencies for Cowrie. These are different than what could be easily found in a list. They are from the CentOS 7 download site as the correct package names need to be used on the command line, each set all on one line.

```
sudo yum install -y python-virtualenv git gcc++ make python3-devel openssl-devel libffi-devel centos-release-scl

sudo yum group install 'Development Tools'

sudo rpm -Urh https://s3.amazonaws.com/aaronsilber/public/authbind-2.1.1-0.1.x86_64.rpm
```

[1,11]

Next, the user cowrie is added, but the cowrie user is not supposed to have a password needed to log in. To do this, the following directions need to be followed:

```
sudo adduser cowrie
```

This creates the user but no one can log into it yet.

```
sudo passwd -d cowrie
```

This takes away the password from the cowrie user

```
su cowrie
```

Change to the user account, cowrie.

```
whoami
```

This shows that the user is cowrie without using a password.

Now installation of the Cowrie honeypot can begin:

```
su cowrie
cd /home/cowrie
git clone http://github.com/cowrie/cowrie
cd cowrie
```

The virtual environment is set up:

```
pwd
```

This should return /home/cowrie/cowrie. Change to that if needed.

```
virtualenv --python=python3 cowrie-env
```

Activation of the virtual environment and installation of packages follows:

```
source cowrie-env/bin/activate
pip install --upgrade pip
pip install --upgrade -r requirements.txt
```

To give more access for attackers, telnet should be enabled. This was done in the cowrie.cfg file. It is created now and the file is written:

```
vi /etc/cowrie.cfg

[telnet]
enabled = true
```

press esc then save and quit the file with :wq

Start Cowrie with:

```
bin/cowrie start
```

Now change user back to class with:

```
su class
```

Type in the password when prompted.

Firewalls will need use forward ports to make the SSH and telnet features of Cowrie work properly while keeping them secure. Port 22 will be forwarded to 2222 and port 23 will be forwarded to 2223. Port 3393 has already been added as the ssh point for administration access. At this point, any other ports that show up with the nmap scan can be closed but this project did not have any extra ports to close at this time.

```
sudo firewall-cmd --permanent --add-port=23/tcp
```

(extra ports can be removed with: `firewall-cmd --permanent --remove-port=xx/xxx`) [4]

```
sudo firewall-cmd --permanent --add-forward-port=port=22:proto=tcp:toport=2222

sudo firewall-cmd --permanent --add-forward-port=port=23:proto=tcp:toport=2223
```

[12]

Check if the honeypot is running by using netstat. Netstat uses ss instead of the word netstat:

```
ss -lntp
```

This command should list out all the open ports. If extra ports are open here that are not 22, 23 or 3393, close them with the above directions for removing ports.

Test to see if Cowrie is working by using SSH to log into the system on port 22. Certain passwords will not work like root:root or admin:admin, but root should work with any random password or even no password.

To SSH into Cowrie, open a new terminal and type in:

```
ssh root@192.241.157.99
```

Enter the password when prompted.

What is seen now is the inside of the Cowrie honeypot. This is what an attacker sees when they use port 22 or port 23 and gain access. It appears to be a whole linux system with poor security measures but in reality, it is a python3 program that is designed to contain attackers and gain their information when they attack. When finished looking through what Cowrie looks like from the inside, type:

```
exit
```

To close Cowrie, log back in to:

```
ssh cowrie@192.241.157.99 -p 3393
```

Type in the Password
Stop the Cowrie honeypot with:

```
/home/cowrie/cowrie/bin/cowrie stop
```

Cowrie needs to be administered when logged in as user cowrie, for example, stopping and starting the service. When logged in as class or root, the Cowrie honeypot functions will not work. [10]

The second portion of the project will install Modern Honey Network (MHN). MHN helps the user install its' own version of Cowrie along with other honeypots and scanners for use in gathering and analyzing data. Once the Modern Honey Network is installed and running, the beginning of each different honeypot or scanner inside it has the same or very similar steps due to the ease of use of the MHN.

SECTION 1-Setting up the server for a new program.

Follow this for each new droplet (server) created.

Create another droplet on Digital Ocean by clicking the green create button, then droplet. Choose the standard server in the 4GB size, type a server name, and create at the bottom of the page. (When making additional droplets, honeypots and scanners can have a 1GB size to reduce costs). A new droplet will load in with the new name. Once it loads in, click the dots on the right and choose: access console.

Log in at the login prompt by typing in root and then the password that comes in the email from digital ocean for that droplet. Use `control C` to copy the password from the email to the clipboard and have a new, long password ready as there is only a short time before the process times out. A new password is required to continue to the server. It is imperative to make sure to have a very good password and not use something people will guess like password, admin, ubuntu, etc. Use a very long, strong password that cannot be brute forced. The password used for this paper was over 20 characters long to reduce the likelihood of brute force attack success.

```
root
Passwordfromdigitalocean (use control v)
Passwordfromdigitalocean (use control v)
Newpassword
Newpassword
```

Next, a new user is going to be made and the root user will not be able to log into ssh when these directions are finished. Type in the console:

```
adduser class
Newpassword
Newpassword
```

(hit enter about 6 times or until the prompt ~# comes back up)

Now class is given sudo privileges by going into the visudo file and adding the user to the file:

```
visudo
```

Scroll down to where is says `root ALL=(ALL:ALL) ALL`
Make the line below it exactly the same except instead of root, put the username underneath it.

```
root ALL=(ALL:ALL) ALL
class ALL=(ALL:ALL) ALL
```

To exit and save use control x, y then enter.
Change to the user class:

```
su class
cd
```

Now a new SSH connection needs to be made before taking the privilege of SSH from the root user. In a new terminal, type:

```
ssh class@ipaddressfromdigitalocean
yes
password
```

If this logs in correctly, then proceed. If it does not, go back to look at the visudo file or check your password to see if it is correct.

The final step of taking the root user off SSH privilege is to alter the sshd_config file:

```
sudo vi /etc/ssh/sshd_config
password
```

Inside the file, find the line that says

```
#RootUserLogin yes
```

Change to:

```
RootUserLogin no
```

Esc, save and exit with :wq
Now restart the server with:

```
sudo systemctl restart ssh
```

[9]

Changing the name of the main access to the machine (root) makes the server less likely to be hacked easily or come under a serious brute force attack. It is best to use a username that is not the standard admin or ubtu type names as those can also be guessed and brute forced.

SECTION 2

Installing MHN, Modern Honey Network, according to Github. The directions are a mix of two different posts as finding one set of directions that works is not always possible. This project did the following:

```
cd /opt/
sudo apt-get install git -y
```

```
sudo git clone https://github.com/erwanlr/mhn.git

cd mhn/

sudo ./install.sh
password
```

Now the program will begin installation. When it stops, it may need the letter q pressed to quit a file. It asks if you want to run in Debu mode? y/n type n. Then it needs an email address. Everything else can be blank as it does not need any other programs or information from this part to continue. Snort will load in which will take a while. Once it is loaded the program asks if it should install Greylog, ELK and add rules to UFW. No can be chosen for all of them if one does not know about them or need them right away. The prompt ~# will come back up. At that point, open a browser and type in the IP address of the server.

`http://159.65.185.24`

The login for the modern honey network should show. If not, something else went wrong and steps will need to be gone over again or examined to find the issue. Otherwise, log into the site with the user credentials

user: `class`

password: `password`

Look around on the site for a few minutes to see what is there. The map is excellent but it will not begin to work until honeypots and scanners are added. Here is where things start to repeat more and become familiar. For every honeypot or scanner that is set up, Section 1 and Section 3 will need to be repeated but with different names/programs. To go to section 3, a new server from Section 1 needs to be set up first, then continue to Section 3.

SECTION 3

Installing a new honeypot or scanner onto MHN.

Click on the word deploy, at the top of the MHN browser page. Where it says select script, click the box, and choose Cowrie (or whichever honeypot or scanner that is wanted). Highlight the deploy command and use control C to copy it to the clipboard. Change to the new server terminal (for Cowrie if installing Cowrie). Make sure to be in the home directory of the class user, not in root. Type in sudo, then paste the deploy command behind it by using `control v` and enter. Type in your password. The program will automatically install the honeypot or scanner and it will connect to MHN, as well.

SECTION 4

Data use, retrieving data and data analysis

MHN can be used in an organization to implement an active defense network that has full functioning capabilities. It is open source and enterprise ready. It manages the honeypots that are on its' network and delivers "active defense" for security teams that are easy to create, customize and implement. [14] The automation of the processes allow integration with ISDses, IPSes, firewalls SIEM and other security tools. MHN uses open source honeypots and sensors like Snort, Cowrie and Amun. MHN also reduces the time that setting up and configuring a honeypot system would normally take. This is emphasized by the differences in installation procedures during this project, noting that traditional methods to set up honeypots are extensive, time consuming and require knowledge, skill, or time that most enterprises are not ready to commit to. The Modern Honey Network changes that and helps enterprises focus on the data that can be gathered quickly and efficiently from running either a few or up to hundreds of honeypots on the network.

Care must be taken when deploying honeypots, especially when they are high interaction. The MHN network is a low interaction honeypot so it is safer as it only gathers information and does not hack back which reduces the risk of exposure. What is gained is attacker intelligence because the data gathered shows what the attackers' actions are. Honeypots are also hardened as part of their design to keep attackers at bay. [7]

On the main page of the MHN, there are lists of data that help to begin the analysis of the data the honeypots have collected.

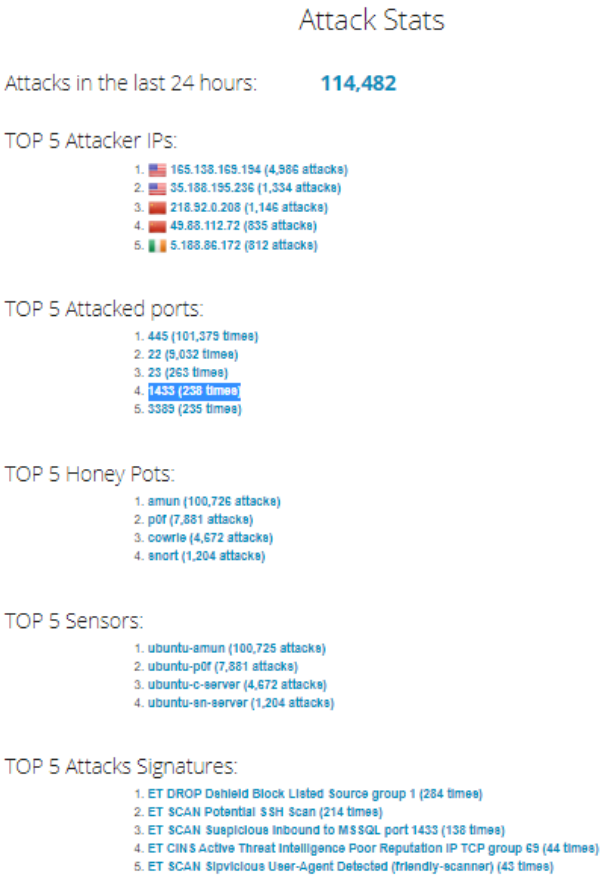

Figure 1 main page attack stats

These lists give great information to an enterprise by showing what ports are most often attacked, what honeypots and sensors are most effective, what IP addresses are attacking the most and the most used signatures of the attacks. With this information several

changes to a network could be made to harden against these specific attacks so that the company network can have even more protection than it would have without this information.

The next set of information comes specifically from Cowrie, as it logs what the user information is and tracks what IP addresses are coming into the honeypot.

Figure 2 Top usernames and combinations

Figure 3 Top passwords and IP addresses

The information of what usernames attackers are choosing, what passwords they are using to gain access and the combination of both can help system administrators to make sure their usernames are unique so they are less likely to be under brute force attack. Passwords can be created by phrases or other means to reduce the chance of brute force by length. Using more than one word at a time can reduce dictionary attacks. Knowing what IP addresses are consistently trying to log into the system gives administrators the knowledge of what IP addresses to begin blocking to reduce the chance of network infiltration.

The payloads report comes from the snort alerts and gives the source IP, the destination port, priority, classification, and signature. This may seem insignificant but in one weeks-time there are over 1477 pages of this information that can be combed through and decisions made on which ports are being hit with higher priority attacks, what those attacks consist of and where they are coming from.

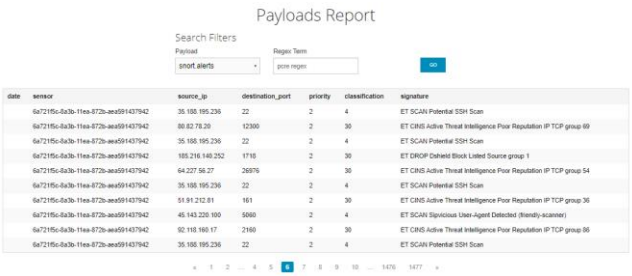

Figure 4 Payloads report

The attacks report is similar, but shows the date, which sensor the attack is against, the country and source IP of the attacker, the destination port, what protocol is being used for the attack and which honeypot is being hit. The Amun honeypot has been hit extensively and this one page of the 102,471 pages of the report shows this. .

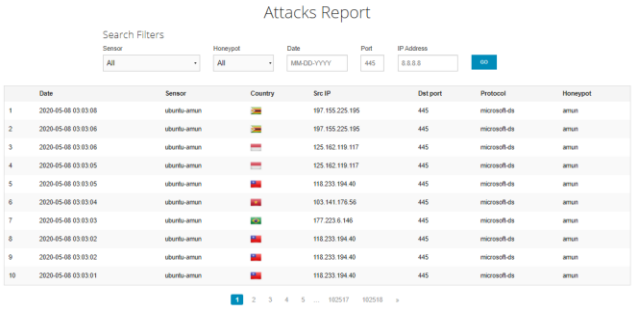

Figure 5 Attacks report

Amun is a low interaction honeypot that is capable of capturing malware that spreads autonomously. The files are captured and held in the /opt/amun/amun.out file. They are color coded with highest risk being red, then yellow and benign is blue, while system code is in green. Possible uses for this could be passing the information that Amun captures to knowledgeable program writers who can create updates to systems. This can prevent the malicious code from harming systems, either privately, publicly, or open source. Anti-virus and anti-malware efforts may rely on this type of system to collect code for widespread mitigation. Following is a picture of a capture with lower level malware in the amun.out file.

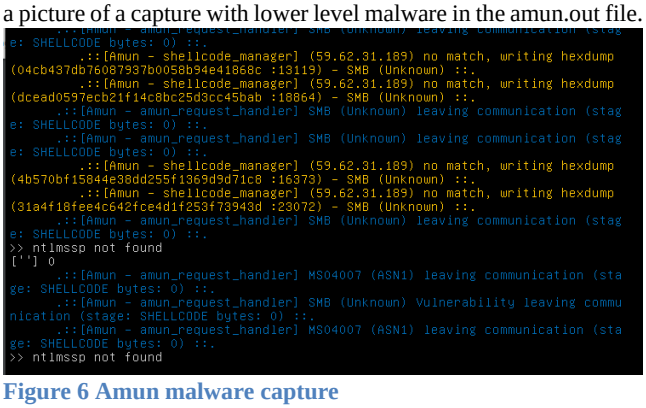

Figure 6 Amun malware capture

The following map was taken after an hour of attacks had built up on the map. Attacks are in red and the honeypot is in yellow. The darker the blue color, the more the attacks originated from that specific country. Over the week that the honeypot has been running, several countries have taken the lead with the darkest blue color and the most attacks. Often it is the U.S., Russia, China, or Malaysia. They take turns depending upon the time of the day the scan is run along with other variables.

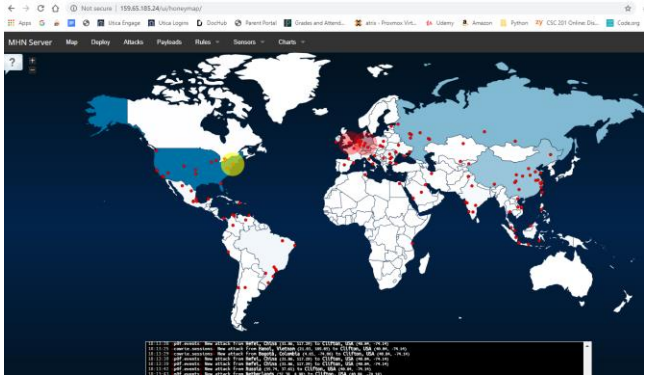

Figure 7 Attack Map from MHN

1.1 Issues and Resolutions

The biggest issue to overcome was that the Cowrie script was originally written in Python2. Python2 is no longer supported so it took time to try different versions of the source for Cowrie until one worked. The deciding issue for the working version was that Python3 has become the new language for the program, or at least a transition version that uses Python3 as its' base. Oosterhof updated the Cowrie documentation April 24, 2020, so that it is now working properly as designed. [11] Additional issues were that the updates and other programs that were needed were written for Ubuntu and CentOS 7 was used for the standalone Cowrie project instead. The choice to use CentOS 7 was a good choice for the project but finding the correct information was more difficult and needed to be pieced together using the Cowrie documentation and the CentOS 7 repositories. Finding the exact match was sometimes found using google search if an immediate match was not found in the CentOS 7 repository search. [1] Often the names can be different or a different program used in its place.

1.1 Conclusion

This project has been an excellent source of study and has provided multiple layers of knowledge base for the student. Understanding how source code effects the projects they are written with was an excellent learning experience. Repeating the same steps multiple times to make the servers more secure was helpful to gain experience with computing tasks. Installation of Cowrie was exceptional for learning as piecing together multiple documents needed to be performed for the task to function. Studying the uses of the honeypots and analyzing the data the honeypots have given was enlightening and will be used again in academic and practical real-life use. Overall, this was a successful and excellent project.

REFERENCES

[1] CentOS. (2020). CentOS Repositories. https://centos.pkgs.org/

[2] CentOS blog (2020). What is CentOS? https://www.centosblog.com/what-is-centos/

[3] Hacker Target. (2020). Cowrie honeypot analysis. https://hackertarget.com/cowrie-honeypot-analysis-24hrs/

[4] Cunningham, M. (April 24, 2020). An introduction to FirewallD. https://www.liquidweb.com/kb/an-introduction-to-firewalld/

[5] GitHub. (2019). Modern Honey Network https://web.archive.org/web/20190924122957/http://threatstream.github.io:80/mhn/

[6] Galobardes, R. (May 13, 2019). Learn how to deploy a honeypot and visualize its data step by step. https://medium.com/@galolbardes/learn-how-to-deploy-a-honeypot-and-visualise-its-data-step-by-step-ea3cd3f25822

[7] Jackson Higgins, K. (June 9, 2014). Open-source tool aimed at propelling honeypots into the mainstream. https://www.darkreading.com/analytics/threat-intelligence/open-source-tool-aimed-at-propelling-honeypots-into-the-mainstream/d/d-id/1278726

[8] Linuxize. (November 11, 2019). How to set up a firewall with FirewallD on CentOS7. https://linuxize.com/post/how-to-setup-a-firewall-with-firewalld-on-centos-7/

[9] nixCraft. (2020). How do I restart sshd daemon on Linux or Unix? https://www.cyberciti.biz/faq/how-do-i-restart-sshd-daemon-on-linux-or-unix/

[10] Nxnjz. (January 12, 2019). Deploying an interactive SSH honeypot on CentOS 7. https://nxnjz.net/2019/01/deploying-an-interactive-ssh-honeypot-on-centos-7/

[11] Oosterhof, M. (April 27, 2020). cowrie documentation release 20.3.0 https://readthedocs.org/projects/cowrie/downloads/pdf/latest/

[12] Red Hat. (2020). Port forwarding. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sec-port_forwarding

[13] SSH (2020). Secure Shell. https://www.ssh.com/ssh

[14] Trost, J. (June 19, 2014). Modern Honey Network. https://www.anomali.com/blog/mhn-modern-honey-network