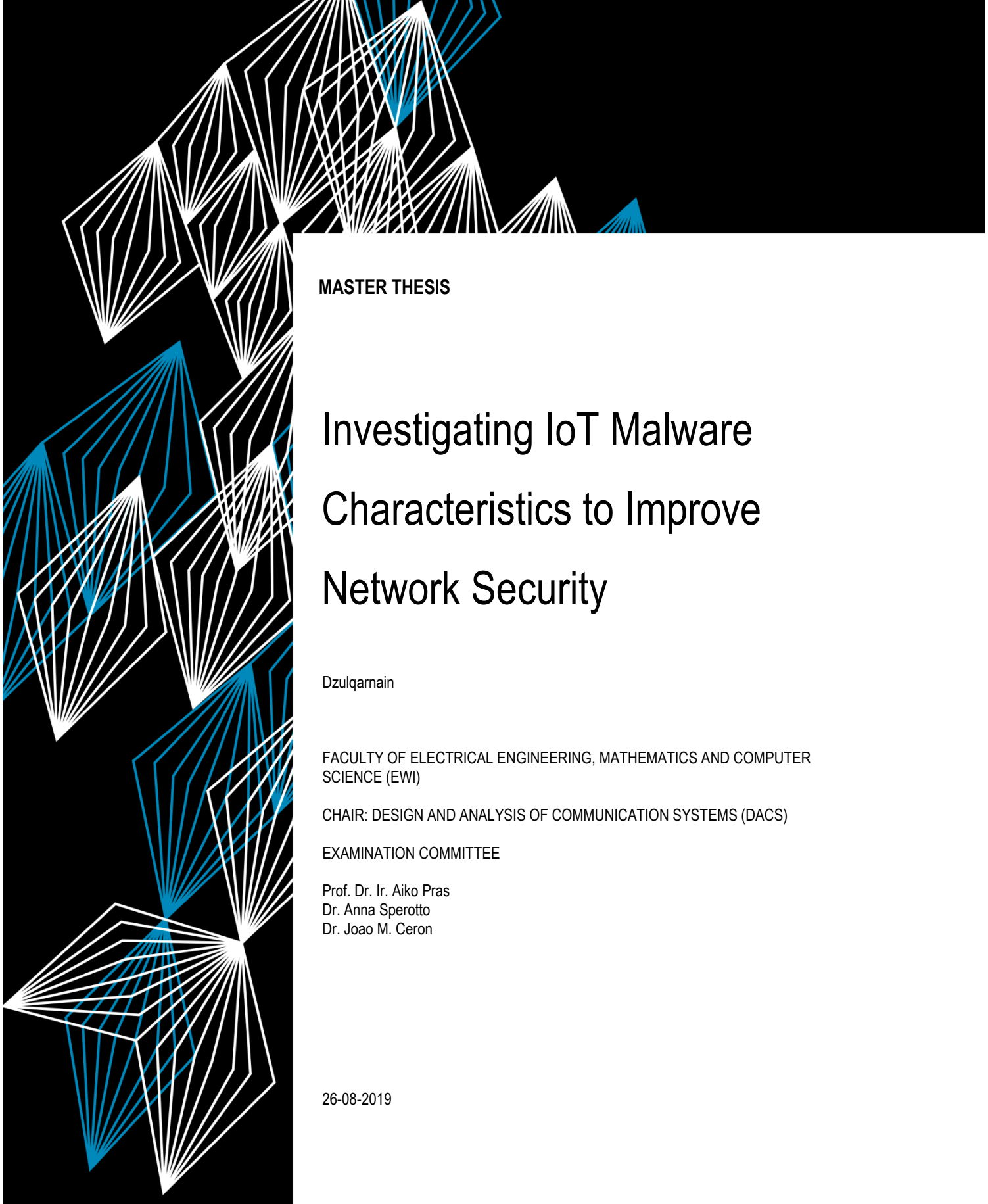


MASTER THESIS

Investigating IoT Malware Characteristics to Improve Network Security

Dzulqarnain

FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE (EWI)

CHAIR: DESIGN AND ANALYSIS OF COMMUNICATION SYSTEMS (DACS)

EXAMINATION COMMITTEE

Prof. Dr. Ir. Aiko Pras
Dr. Anna Sperotto
Dr. Joao M. Ceron

26-08-2019

UNIVERSITY OF TWENTE.

Abstract

The Internet of Things (IoT) revolution offer not only interconnected a whole generation of devices but also brought to the Internet plague of billions of poorly protected and easily hack-able devices. Not surprisingly, this sudden flooding of fresh and insecure devices fueled threats, such as IoT malware. IoT malware that keeps evolving brings the importance of analyzing techniques that can be used to keep up with the growth of IoT Malware. In this research, we present a set of techniques to analyze the malware in order to understand and block its activity. We develop an hybrid approach that combine with machine learning to classify the malware family based on the network traffic. We have evaluated our solution in a set of 1700 malware collected during one year. As a result, we shows that our approach can identify the malware with the accuracy of 92%.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Internet of Things (IoT) is the next phenomenon in the world of the Internet. The numbers of IoT devices keep increasing day by day, and this growth is also followed by abuses that explore the insecurity of those devices [16]. Hacker(s) targeted IoT devices mainly because it has little or no built-in security [14]. Furthermore, the surplus of IoT devices has become a target of several different types of malware, for instance, by exploiting the devices to build large-scale malicious networks called botnets [16].

Today, malware has been developed by using newer and innovative techniques to change the internal architecture of malware and procedures to avoid detection. Techniques that have been used in the past few years to detect malware still facing the obstacle of detecting these new forms of malware. Furthermore, malware changes behavior or feature set very frequently. Thus, it makes the techniques such as behavioral or signature-based that attempt to detect new variants of malware will likely fail.

With thousands of IoT malware released every day, it is essential to distinguish a new malware family from an older variant to increase the system protection. However, there are only limited methods to investigate and characterize the IoT malware and most of them comes with disadvantages such as high machine learning cost, or lack of detection accuracy. Thus, there is a need to develop methods in investigating the behavior of IoT malware and increasing protection against these threats.

As presented in [3] [8] [6], the majority of IoT malware families are related to botnets and worms.

In 2018 IoT botnet attack represents 78% of malicious software detection. This number has doubled compared to 2016 when this type of attack become known to public [4]. This context urge us to understand the malware intents and characterize their behavior. In this research, we will tackle the botnets, the most common category of malicious code present on the Internet of Thing environment. Therefore, this research will provide a methodology for investigating the characteristics of malware which able to identify botnet controller IP address, families, and the malware characteristic.

The aim of this research is to expose the Internet Protocol (IP) address of Command & Control (C&C) server by analyzing the network traffic of infected IoT malware. To achieve our goal, we begin this research by studying the state-of-the-art of IoT malware and their characteristics. Based on that, we develop set of approaches to collect and classify the malware family that exclusively target IoT devices. The set of approaches was developed using a combination of dynamic analysis with machine learning algorithm (hybrid approach). We tested our approaches in 1700 samples which collected in one year. At the end, we able to identify the C&C IP address in 603 of samples which spread in 52 number of variant, also with their particular set of features.

Knowing the problem and proposed solution, the remaining of this introduction describes the goals of this thesis in Section 1.1, explains the structure of the thesis in Section 1.2. and highlights the contributions in Section 1.3.

1.1 Goals

The aim of this thesis is to understand the IoT botnet behavior in order to block its activity. To achieve this goal, four main challenges arise (i) to properly understand the IoT malware, (ii) to have accurate methodology to classify the IoT malware, (iii) to determine the features that differentiate each of the malware family, (iv) to correctly find the address of C&C server. Therefore, to address goal and the challenges imposed, a set of sub research questions is defined as following:

1. What is the state-of-the-art of IoT malware and their classification? To answer those question, we do the literature study of malware and the classification techniques that have been used in the past few years. We learn that, from the past few years a couple of techniques starting from static analysis to dynamic analysis have been used to analyze the malware both of them have their own disadvantages in classifying the malware. Thus, to overcome the shortcomings

of the both techniques a hybrid approach was created.

2. How to determine the family of malware?
   To determine the family of malware, we used hybrid approach to find a specific protocol messages in their network communication that could differentiate each of the family. We used those message to cluster a patterns of communication that leads to malware family.

3. What are the characteristics of IoT malware?
   In order to identify the characteristic of IoT malware, we build a database consist of the network traffic behaviour of our malware. The database which we build compose by many IoT malware, from different malware families collected from 1 year daily basis. We classified the malware based on their family and find a pattern that differentiate them between each other. As a result, we managed to understand the main characteristic of IoT malware collected in the wild.

4. How to find the C&C server controller?
   To answer those question, we propose a set of methodology called hybrid approach to inspect the malware communication and identify the malware family with their C&C address. The methodology consist of several steps and was develop with the combination of sandbox, heuristic analysis and the machine learning. By doing this, we show we can identify the IP of the C&C botnet controller.

1.2 Structure

In order to answer the questions stated above, the research was written into several chapters. First, we explain the importance of our research and what are the goals of this research. This topic is covered in Chapter 1. Second, we provided the state-of-the-art of IoT malware and their characterization. That topic is written in Chapter 2 when we explain the background of IoT malware, the challenges, and the past techniques that have been used in classifying the malware. Next, Chapter 3 present the methodology on how to achieve our goals by presenting several approaches that we will explain further in this chapter. Chapter 4 elaborates further regarding the system design that we used in our research. Chapter 5 will present the result of our research in relation to the goal of this research. The thesis is concluded in Chapter 6, together with suggestions for future work. Finally,

it is wrapped up with some reflections and acknowledgements, after the conclusions. Next section highlights the contributions of this thesis.

1.3 Contributions

This thesis addresses the four main questions asked in Section 1.2 and the additional challenges mentioned at the beginning of this introduction. As a result, the following contributions can be listed as an outcome of this thesis:

1. To get better understanding about IoT malware and its investigation techniques, the state-of-the-art of IoT malware is researched. It showed the investigation can be done using several techniques. Our research contribute to the development of hybrid approach in classifying the IoT malware;

2. To address the classification of the malware, a set of approaches to gathering relevant information related to malware is developed. For this, a proper set of search terms is defined. The approaches gathered for the purpose of this thesis including the framework using machine learning algorithm is available in Appendix B for all interested researchers;

3. By careful investigation of malware characteristics, a set of proper features to classify a malware is defined; the data-set collected for purpose of this thesis is available in Appendix A for all interested researcher;

4. To find the C&C IP address of controller, we present an approach to inspect the network traffic of the infected machine. It shows that the botnet controller can be found using our approach. The list of the IP address of C&C server that we found during this research can be provided based on the request.

Chapter 2

Background

To understand the subject of IoT malware, this chapter investigates what the state-of-the-art of malware and their characterization is. This is done by analyzing currently available literature about malware, as well as blogs of security experts, and white journal. Based on such inputs, malware is defined, and the existing methods for their characterization are investigated. Section 2.1 gives an explanation about how malware is analyzed and what are their behaviors. Section 2.2 explains how malware and IoT related to each other and what is the behavior of IoT malware. Section 2.3 explains the known methods that could be applied in tackling the problem of characterization of IoT malware. With the idea of characterizing IoT malware, this chapter is closed with a review of possible methods and features used to classify a IoT malware in Section 2.4.

2.1 How malware are analyzed and what are their behaviors?

The purpose of this research question is to gain a better understanding of malware behavior and how do we analyze it. A lot of research community has been investigated on this subject, and it will be studied to answer the question. On the first section of this chapter, we will discuss malware analysis including how we analyze the malware and what kind of methods that have been used to analyze the malware. The second section we will discuss malware behaviors including types of malware and their characteristics. Then, the last section will summarize the main point of each section.

2.1.1 Malware analysis

Several methods have been studied to analyze and create the signatures for malware behaviors; these methods can be classified as static analysis and dynamic analysis. Both of these methods were performed in the past few years to understand the associated risks and intentions of malware. In this section, we will provide information regarding what is static analysis and dynamic analysis and how they have been used to analyze the malware.

2.1.2 Static analysis

Static analysis is a method to analyze the malware without executing it. The static analysis uses certain tools and techniques to determine whether a file is malicious or not. It is also used to provide information about the functionality and collect technical indicators to produce simple signatures [24]. Technical indicators gathered by static analysis can include file name, machine code instructions, file type, file size and detection by anti-virus detection tools.

Static analysis has the advantage to reveal the code structure of the malware under consideration. However, the drawback of static analysis is it may fail in analyzing unknown malware that uses code obfuscation techniques [13]. Since it will transform the malware binaries into self-compressed and uniquely structured binary files, thus make the static analysis unreliable [13]. Another disadvantage of static analysis is the user who perform the static analysis must possess a good knowledge of assembly language and the working operating system.

Research by Moser et al.[26] explore the drawbacks of static analysis methodology. They present a scheme based on code obfuscation revealing the fact that the static analysis alone is not enough to detect or classify the malware. Further, they proposed that dynamic analysis is a necessary complement to static analysis as it is less vulnerable to code obfuscation conversion.

2.1.3 Dynamic analysis

Dynamic analysis is a method to analyze the malware by running it on the environment. The dynamic analysis runs malware to observe the behavior, understand the functionality and identify technical indicators which can be used in detection signatures. Technical indicators revealed with

basic dynamic analysis can include domain names, IP addresses, file path locations, registry keys, and additional files located on the system or network [24]. Additionally, it will identify communication with an attacker-controlled external server for C&C purposes or in an attempt to download additional malware files [24]. Based on the output, it is possible to uses a behavior-based approach for malware detection and analysis.

Dynamic analysis has several advantages compared to static analysis. It does not require the executable to be disassembled [13]. It discloses the natural malware behavior which is more resilient to static analysis, and it is more effective against the malware since it analyzes the sample by executing it [9]. Due to the effectiveness against the malware, dynamic analysis is more favorable in analyzing the malware. However, we should take a note that dynamic analysis is time intensive and resources consuming, thus elevating the scalability issues.

For the past few years a large number of new malware samples keep growing on the Internet [19]. This situation causes using an old method in static or dynamic only is not enough to analyze the characteristic of the malware. It needs an improvement approach to give a better analysis of against malware.

2.1.4 Hybrid approaches

Nowadays, Artificial Intelligence (AI) techniques, particularly machine-learning (ML) techniques have been used by the researcher to automated malware analysis and classification [2]. According to the definition given by AI pioneer Arthur Samuel, machine learning is a set of methods that gives computers the ability to learn without being explicitly programmed. In other words, a machine learning algorithm discovers and formalizes the principles that underlie the data it sees. With this knowledge, the algorithm can be used to discover the previously unseen samples. In malware detection, a previously unseen sample could be a new file or undetected malware [23].

Various ML algorithm combine with static or dynamic approach has been used by the researcher in the past few years. Algorithm like Association Rule, Support Vector Machine, Decision Tree, Random Forest, K-Neighbors, and Clustering have been proposed for classifying new malware samples [9]. Those algorithm mostly used with the combination of dynamic analysis to make a trained model. Then, the trained model will be used when making a decision a to classify the malware [9].

The combination between ML technique with dynamic analysis gives the advantage in processing a large amounts of malware. However, these approach also have some limitations. ML technique need a constant training and can only deal with the well known behavior of malware. This means that solutions based on ML should be revisited to combat new threat.

2.2 Malware behaviours

Malware comes in wide range of variations like Virus, Botnet, Spyware, Worm, etc. These classes of malware are not mutually exclusive meaning thereby that a particular malware may reveal the characteristics of multiple classes at the same time [24]. As mentioned in Chapter 1, malware's growth keeps increasing and evolving. This situation also followed by the transformation of the behaviors of malware. Due to these reasons, we argue that it is important to understand the type of malware and their behaviors before we investigate further into how do we characterize malware behaviors.

2.2.1 Types of malware

Malware is commonly divided into some classes, depending on the way in which it is introduced into the target system and the sort of policy breach which it is intended to cause [37]. The following categories are the most observed in the literature.

A. Virus

The virus is an executable piece of code that can infect computers without knowledge or permission from the user [33]. An important point concerning viruses is that they cannot replicate independently; they need to be transferred to another computer and run by a user. The virus can be transferred using various ways such us via external hardisk or email. It was done in order to convince the user that the file they are opening is benign. Problems from viruses can vary greatly, with symptoms ranging from only using system resources to formatting hard disks. In general, the problems are related to the specific device in question and will not affect others on the network [33].

B. Worm

The worm is a malicious code that spread through an Internet connection or a local area network (LAN) [41]. Worms can be classified as a type of computer virus, but several characteristics distinguish computer worms from regular viruses. A major difference is that computer worms can self-replicate and spread independently while viruses rely on user activity to spread (running a program, opening a file, etc.). They spread over computer networks by exploiting operating system vulnerabilities. Worms typically cause harm to their host networks by consuming bandwidth and overloading web servers. Computer worms can also contain payloads that damage host computers. Payloads are pieces of code written to perform actions on affected computers beyond simply spreading the worm. Payloads are commonly designed to steal data, delete files, or create botnets [11].

C. Ransomware

Ransomware is a type of malware that infects a computer and takes control of either the core operating system using lockout mechanisms or possession of data files by encrypting them [31]. The program then asks the user to make a ransom payment to remove the locks and restore the users files. Ransomware typically spreads like a normal computer worm (see section Worm) ending up on a computer via a downloaded file or through some other vulnerability in a network service.

D. Spyware

Spyware is a type of malware that functions by spying on user activity without their knowledge. These spying capabilities can include activity monitoring, collecting keystrokes, data harvesting (account information, log-in, financial data), and more. Spyware often has additional capabilities as well, ranging from modifying security settings of software or browsers to interfering with network connections [11]. Spyware can spreads by exploiting software vulnerabilities or by bundling itself with legitimate software.

E. Bots

The bot is a type of malware that is originating from the term 'robot'. The bot is an application that can perform an automated process that interacts with other network services. Although bot can be used for good intent, it mostly used for malicious intent. When a large number of bots spread to several computers and connect through the Internet, the bot can transform into a network of the bot. This situation called botnet [37].

A bot is designed to infect targets devices (e.g., computers, mobiles or IoT devices) and turn them into a part of a botnet without the knowledge of the device owner and under the control of a human, known as the bot master. The bot master sends a command to all the bots and controls the entire botnet through the Internet and the C&C servers [30]. The bot masters try to control all of these targets and carry out their malicious activities. In a review of the different types of malicious activities perpetrated by botnets, it is found that they are not only dangerous threats to computer networks but also used as an infrastructure to carry out other types of threats and attacks (e.g., DDOS) [30].

A botnet consists of three main elements - the bots, the command and control (C&C) servers, and the bot masters. Thus, it can come in different sizes or structures but, in general, they go through the same stages in their life-cycle [4,5]. Figure 2.1 depicts the general view of a botnet life-cycle.

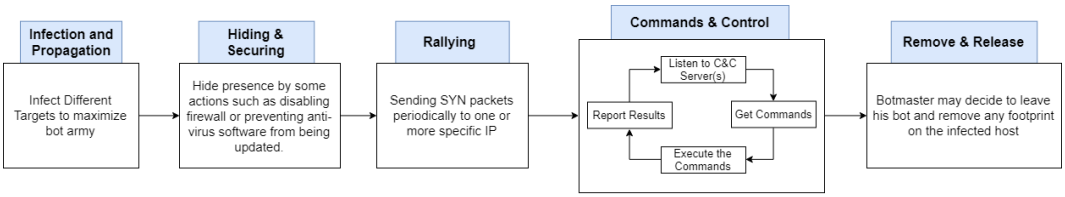


Figure 2.1: A Botnet life-cycle schema

In *Infection and Propagation* phase, bot master tries to maximize their infection to get bots via infecting new hosts. This method was done using a variety of methods such as propagation in the local network through a shared folder or trick the user to visit malicious web pages [20]. After successful infection, the cycle move into *Hiding and Securing* phase. In this phase, the bot tries to hide the presence by some actions such as disabling the protection systems or preventing anti-virus software from being updated. In *Rallying* phase, the bot process tries to send SYN command to gain a connection to C&C server or peers address, which is hard-coded in bot binary or found through

an alternative method. When bot successfully connects to server or peer, it will become a part of the botnet. After this stage, the *Command and Control* phase will begin. In this phase, the bot can maintain a connection with the C&C server and ready to receive an order from the bot master and perform specified order. Bot master may have some communications with his bot to obtain required information about it, e.g., OS version.

Furthermore, bot master can control their bots army to update their binary to hinder them from being detected or improving their functionality. Bot master may command his bots to do any malicious activities such as participating in a DDoS attack, sending spam emails or harvesting sensitive information. In some situations, bot master can also decide to remove any footprint on the infected host and leave his bot [20]. These operations are known as *Remove and Release* phase.

2.3 What are the characteristics of IoT malware?

The changing behavior of IoT malware leads to the importance of characterization. By building the characteristic of malware, researchers can get a better understanding of IoT malware. Characteristic of IoT malware can include several information such as what is their target, types of attack vector, and the communication with controller [22].

In this chapter, we will mainly discuss the characteristic of IoT Malware. Section 3.1 will discuss IoT malware; section 3.2 will discuss several types of IoT malware that appeared in the past few years and what are their characteristics. Then, the final section, summarize the important aspect of each of IoT malware and how important to deal with this class.

2.3.1 IoT malware

In the past few decades, the security community had been focusing on Windows-based malware. It is because most of the of malware was designed to target personal computers running Microsoft Windows operating system [8]. It is understandable since Windows operating system market share currently estimated at 83% for desktop computers [8].

However, the diversity and the number of computing devices rapidly increased during the last few

years, in particular, due to what is known as the Internet of Things (IoT) paradigm [18]. IoT devices are profoundly different from traditional personal computers. For example, while personal computers run predominantly on x86 architectures, IoT devices are built upon a variety of other CPU architectures and often on hardware with limited resources. To support these IoT systems, developers often adopt Unix-like operating systems, with different type of Linux. Along with the change of the market, the focus of malware authors and operators is also shifting towards IoT malware.

Nowadays, IoT devices become a favorable target of malware due to a lack of security design with most IoT devices. The overly simplified designs and functions of most IoT devices make it one of the most favorable targets by the hacker [1]. It is confirmed by the infamous attack of Mirai and the release of the source-code in 2016 which started a new wave of IoT malware [7]. The situation becomes worse when a considerable spike regarding new IoT attacks and malware families happen in 2017 [7].

By performing a literature study, several studies characterize different aspects of IoT malware. Among them we summarize several characteristics:

- IoT malware is often used to perform DDoS attacks;
- IoT malware exploits the port of IoT service such as Telnet, FTP or HTPP;
- IoT malware uses a brute-force attack to gain access to IoT devices.

There are a lot of malware families trying to infected IoT devices. However, malware that turns IoT devices into botnet tends to be the most used for the past few years [7]. On the next section, we discuss the paradigm of IoT Botnet and the IoT malware families that appeared in the past few years.

2.4 IoT botnet and their characteristics

IoT Botnet is a group of IoT devices (cameras, routers, wearable and other embedded technologies) infected with malware known as bot. IoT Botnets have a wide range of purposes including email spam delivery, DDoS attacks, password cracking, key-logging, and cryptocurrency mining.

In order to become an effective part of a botnet, a vulnerable IoT device passes through the sequence of stages as described on section 2.1. However, due to the differences between the operating system. IoT botnet tends to behave differently compare to normal botnet that have Microsoft OS.

In their work, Cozzy et al [8] present several characteristics of IoT botnet. IoT botnet tend to modify Executable and Link-able Format (ELF) header to fool the analyst or crash common analysis tools. IoT botnet also uses several techniques in order to keep unaffected even though the devices has been rebooted.

Despite of aforementioned behaviors, IoT botnet also has several characteristics especially in C&C infrastructure. The C&C server has simple and unstable C&C infrastructure and it manages huge amount of bot.

2.4.1 Most common malware families

In this section, we will discuss several types of IoT botnet family including Mirai, Bashlite, Hajime, Brickerbot, New Aidra, and VPNFilter. Following the discussion, we summarize the results into the table of comparison between IoT Botnet.

A. Mirai

Mirai identified in August 2016 by the security research group [21], Mirai variants and imitators have served as the vehicle for some of the most potent DDoS attacks in history. The Mirai source code is primary written in C while the command and control is written in Go. In total, the repository investigated contains over 12,000 lines of code in 144 files. Analyses of Mirai have been numerous both before the release of the source code and since [10]. While analyses vary, it is estimated that Mirai builds on previous botnet malware and even previous IoT botnet malware such as Bashlite [10].

Mirai functionality is very straightforward. It spreads by attempting to connect to randomly selected devices via the Telnet port and then guessing the user name and password from a hard-coded list of default credentials [27]. Most of the credentials found in this list are either exceedingly common (e.g. root:password) or are specific to a manufacturer or device. All of these combinations are likely

to target a variety of cameras, routers, DVRs, printers, and more [27]. Today, Mirai mutations are generated daily, and the fact that they can continue to proliferate and inflict real damage using the same intrusion methods as the original malware is indicative of IoT device vendors chronic neglect in applying even basic security practices [21].

B. Bashlite

Bashlite (also known as Gafgyt, Q-Bot, Torlus, LizardStresser, and Lizkebab) is one of the most infamous types of IoT botnets. Bashlite code and behavior can be found in other IoT malware as well. It uses a Telnet scanner and a small set of usernames and passwords, and identifies BusyBox based systems upon successful login. The set of credentials include 6 usernames and 14 passwords. [21]. According to literature, Bashlite exploits the Bourne Again Shell (Bash) ShellShock vulnerability that can be used for Remote Code Execution (RCE) [21].

C. Hajime

The Hajime botnet, discovered in October 2016 by Rapidity Networks [12], uses a method of infection similar to that of Mirai. However, rather than having a centralized architecture, Hajime relies on fully distributed communications and makes use of the BitTorrent distributed hash table (DHT) protocol for peer discovery and the uTorrent Transport Protocol for data exchange. Every message is RC4 encrypted and signed using public and private keys [21]. So far, Hajime has not evidenced malicious behavior; in fact, it closes potential sources of vulnerabilities in IoT devices that Mirai-like botnets exploit, causing some researchers to speculate that it was created with a good intentions of developer [27]. But the true purpose remains a mystery.

D. Brickerboot

A BusyBox-based IoT botnet like Mirai, BrickerBot was unearthed by Radware researchers in April 2017 [29]. BrickerBot is a new botnet that *bricks* devices after they are compromised. Bricking a device implies that the device is unusable afterwards, essentially turning it into a brick. It was done by leveraging SSH service default credentials, misconfigurations, or known vulnerabilities. This malware attempts a denial-of-service (DDoS) attack against IoT devices using various methods that

include defacing a devices firmware, erasing all files from the memory, and re-configuring network parameters.

E. New Aidra

NewAidra or known as Linux.IRCTelnet is a combination between Aidra root code, Kaiten IRC-based protocol, BASHLITE scanning/injection, and Mirai dictionary attack [36]. All the embedded devices based on standard architectures can be infected by this malware, and the variety of DDoS attacks is large. NewAidra have a lot of features in attacking the target. NewAidra not only using the standard attacks, but also can choose a several TCP Flood (as an example, URG Flood attack). At the present moment, NewAidra is the strongest Mirai competitor in the worldwide IoT infection crusade [10].

F. VPNFilter

VPNFilter is a malware specifically designed to harm network router and network attached storage devices. Unlike other IoT malware, VPNFilter is one of the malware that can survive a reboot process [34]. VPNFilter uses third stage operations after the initial infection. Stage 1 is used to maintain a persistent presence on the infected device and will contact a command and control (C&C) server to download modules. Stage 2 is capable of file collection, command execution, and device management. The last stage known Stage 3 modules, which act as plugins for Stage 2. These include a packet sniffer for spying on traffic that is routed through the device, including theft of website credentials and monitoring of Modbus SCADA protocols [18].

It was reported that more than 500,000 devices around the world already infected with this malware [15]. Most of them are consumer Internet routers from a range of different vendors, with some consumer NAS (network attached storage) devices known to have been hit as well.

After discussing several types of IoT malware, we will summarize this section into single table. We made table 2.1 to summarize all of the IoT botnet family that we discussed before. We should take a note that even though most of them have the same purpose to take down a system by performing a DDoS attack, they have different characteristics in infecting the devices. It is mainly due to the code reuse technique that keeps evolved [18].

Table 2.1: IoT botnet family

Botnet Family	**Goal**	**Characteristics**
Bashlite	DDoS	Infecting a IoT device by brute-forcing Telnet protocol using known default credentials
BrickerBot	DDoS	Brute-forcing Telnet credentials on ISPs leaving port 7547
Hajime	Not known yet	Using several attack methods consist of, Telnet default password attack and vulnerability on ISP
New Aidra	DDoS	Brute-forcing IoT device via Telnet protocol
Mirai	DDoS	Brute-forcing devices via Telnet protocol and TCP/2323
VPNFilter	Steal data	Specifically targeting router and NAS devices via 3 stage of infection

2.5 Related works

In the past few years, the literature shows different approaches on characterizing the IoT malware. Wang et al [40] analyze multiple IoT malware which have appeared in the recent years and classify them into two categories according to the way they infect devices: one is by brute force attacks through a dictionary of weak usernames and passwords; while, the other one, by exploiting unfixed or zero-day vulnerabilities found on IoT devices. They choose Mirai, Darlloz, and Bashlite as examples to illustrate the attacks. In the end, they present strategies to defend against IoT malware. However, their strategies did not present the characteristics of malware and how it can be developed into a proper malware signature.

Jaramillo et al [17] use framework from National Institute of Standards and Technology (NIST) on how to handle the malware. In his research, the framework used by the combination of several open source software that available online to identify, classify and remove malware from a compromised

system. He also presented an analysis of Mirai botnet, including top countries of origin of Mirai DDoS attacks. He claims the methods that presented are generic and can be used to mitigate a malware of the same nature as Mirai. However this statement has not been verified in his research. Alhanahnah et al [3] use two real-world IoT malware datasets with 5.150 malware samples, they observe the cross-architectural similarity among malware samples from the same family. Based on this keen observation, they propose a multistage clustering mechanism to group these IoT malware samples into multiple families using the code statistics feature, high-level structural similarity, and ML features. They design a signature generation scheme to create signatures using extract-able string and statistical features. Finally, they perform experiments using datasets consisting of benign firmware binaries and additional malware samples downloaded from product websites and malware sharing servers.

Prokofiev et al [28] proposes a method to detect botnets at the propagation stage, which includes the first stage of the bot life-cycle the primary infection. The method is based on a model of logistic regression. The research describes a developed model of logistic regression which allows estimating the probability that a device is initiating a connection is running a bot. A list of network protocols used to gain unauthorized access to a device and to receive instructions from C&C server, is also provided. However, due to the lack of samples, the model is applicable only for detection of botnets, which are propagated through brute-force attacks using the Telnet and SSH protocols.

Torabi et al [35] devise data-driven methodologies to infer compromised IoT devices and those targeted by denial of service attacks. They obtained information related to 331,000 IoT devices from Shodan. Then executed a correlation algorithm that leverages IP header information to associate the obtained IoT device information with dark net flows. They perform characterization analysis of their traffic, as well as explore a public threat repository to underlie their malicious activities. They expose 26 thousand compromised IoT devices in the wild, with 40% being active in critical infrastructure. Lastly, they present malware variants that target IoT devices. Their empirical results highlight the large-scale insecurity of the IoT paradigm while alarming about the rise of new generations of IoT botnets.

In the next research, Su et al [32] classify IoT DDoS malware samples recently collected in the wild on two major families, namely Mirai and Linux Gafgyt. Then they propose a lightweight solution for detecting and classifying IoT DDoS malware and benign applications locally on the IoT devices by converting the program binaries to gray-scale images, and by feeding these images to small neural network algorithm for classifying IoT malware families. The experimental results show that the

proposed system can achieve 94.0% accuracy for the classification of malware and DDoS malware, and 81.8% accuracy for the classification of malware and two main malware families.

Meidan et al [25] propose and evaluate a network-based anomaly detection method which extracts behavior snapshots of the network. They uses deep auto-encoders to detect anomalous network traffic emanating from compromised IoT devices. They infected nine commercial IoT devices with two of the most widely known IoT-based botnets, Mirai and Bashlite. Their evaluation results demonstrated proposed methods ability to detect the attacks as they were being launched from the compromised IoT devices which were part of a botnet.

Bezerra et al [5] propose a host-based detection system based on one-class classifiers. It was used a ML algorithm built with features such as CPU and memory usage to detect malicious activities. The predictive performance and resource consumption of the proposed approach was evaluated in a controlled network using three different legitimate settings and seven IoT botnets.

To provide convenience in reading, we summarize those research into table 2.3.

Table 2.3: Related works

Paper	**Goal**	**Specific Malware**	**Specific Attack**	**Approach**
Wang et al. (2017)	analyze and classify multiple IoT malware	Mirai, Bashlite, Darlozz	DDoS attack	using dynamic analysis
Jaramillo, L. E. (2018)	mitigation method	Mirai	DDoS attack	using Anti Virus Clam
Alhanahnah et al. (2018)	produce a signature of IoT Malware	various malware	various attack	using static analysis combine with ML algorithm
Prokofiev et al. (2018)	detection of botnet at propagation stage	Mirai	DDoS attack	using ML algorithm
Torabi et al. (2018)	detection of DDoS malware	various malware	DDoS attack	using ML algorithm
Su et al. (2018)	detection of IoT malware	Mirai and Linux Gafgyt	DDoS attack	using ML algorithm
Meidan et al. (2018)	network based detection method	Mirai and Bashlite	DDoS attack	using ML algorithm

Bezerra et al. (2018)	host-based detection method	various malware	various attack	using ML algorithm

To counter the trade-off between analysis speed and detecting obfuscated malware, researches have adapted a technique incorporating a combination of static and dynamic features with machine learning algorithm for detecting and classifying malware. It has an advantage of processing large data of malware in an automated way. However, there are several issues were found in these works such as the lack of testing in real devices and the efficiency of detecting new malware. The researches also demand computationally costly methods, and need for large amounts of data to train the models.

2.5.1 Conclusion remarks

The infection of malware evolving from personal computing devices into IoT devices. Thus, it leads to a new threat called IoT Malware. IoT Malware often used to perform DDoS attacks, exploits the exposed port of IoT devices such as Telnet, FTP or HTPP port, and uses brute-force attack to gain an access to IoT devices. We also noted that malware can differ between each other and have their own goal in infecting the user due to the obfuscation and polymorphism technique.

In this research, we present the techniques such as static analysis, dynamic analysis, and hybrid analysis that have been used in the past few years. To hindering the trade-off between analysis speed and detecting obfuscated malware, researches have adapted a technique incorporating a combination of static and dynamic features with machine learning algorithm for detecting and classifying malware. However, several issues were found in these works such as the lack of testing in real devices and lack of efficiency on detecting new malware.

Considering these issues we observe an opportunity to smaller the gap, we proposed an approach for botnet detection in IoT devices that do not need computationally costly methods. We offer a method to collect and characterize the malware behavior using sandbox (instrumented system). Moreover, by combining data from Sandbox with heuristic analysis from network traffic of different IoT malware families we hope it will highlight the malware intentions and, for instance, correlate the authors by comparing the malware administrative infrastructure.

Chapter 3

Methodology

As previously described, understanding the IoT botnet behavior and characteristics can improve the mechanisms of defense. Thus, our methodology aims to identify the botnet controller contacted by an infected device. As a result, it is possible to restrict the access of infected devices and inhibit malicious actions performed by botnets. This is lead to the main objective of our methodology which is to identify the C&C IP controller of malware. In order to achieve the goal, we structured our methodology into two approaches (see Fig 3.1).

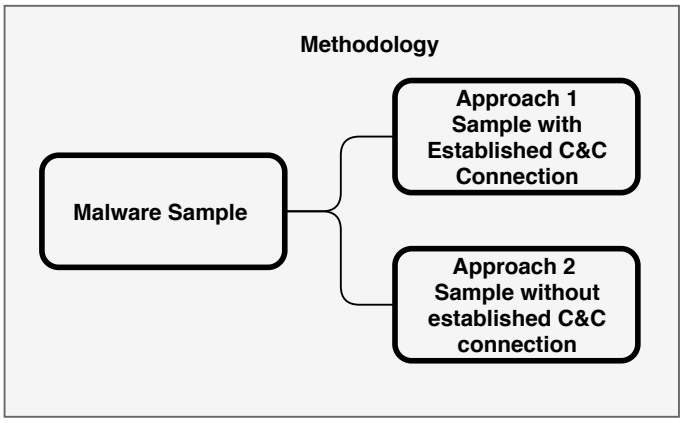


Figure 3.1: Structure of chapter 3

In the first approach, described in Section 3.2, we explain how we can identify the C&C IP when they have established communication with infected device. In the second approach, described in

Section 3.3, we present our methodology to find the C&C IP when they do not have established communication with infected device.

3.1 IoT botnet communication

In Section 2.2.1, we have revisit the botnet communication process. Here, we highlight the IoT botnet communication process focusing on the messages and protocol instructions. This is important in order to understand our methodology. Figure 3.2 depicts a common process of message exchange between the IoT Bot (infected device) and the respective C&C server.

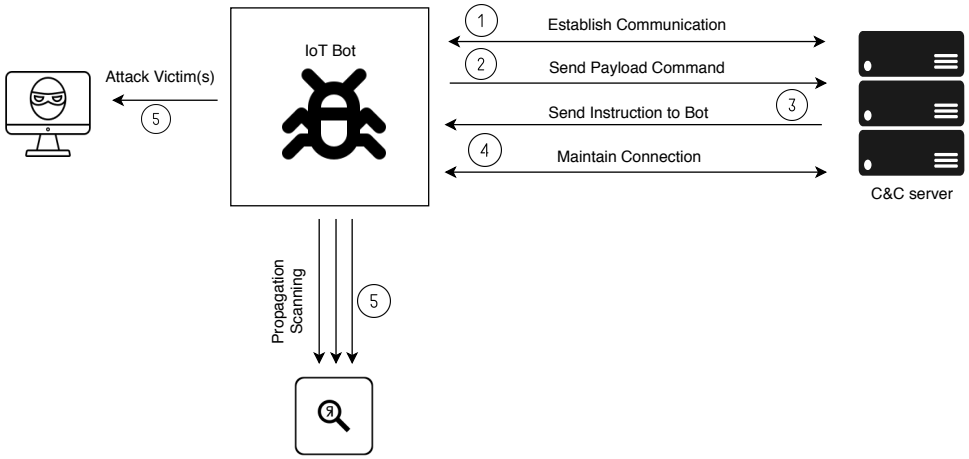


Figure 3.2: Bot communication with C&C server

Firstly, the bot will go to step ① to get an established communication with C&C server. The bot will send SYN packet and wait to get an answer from C&C server. After that, C&C server reply with SYN-ACK then the bot answer with sending an ACK. The next process move into step ②. In this stage, some of the bot will send a message or instruction to C&C server that usually has a string that identify the malware itself, called *identification string*. The identification string might differ depending on the variety of the malware. This identification string is also used to report to the C&C server which version or variant the device is running. As a consequence, the C&C knows which malware version is running in the affected device being able to request the proper attack commands. Later, we detail how we used the identification string in our methodology to classify the botnet and find the botnet C&C.

In sequence, step ③, the C&C sends instructions to the bot. Usually, this instruction can request the bot to perform identification or some attack commands. As long as these steps have or have not been implemented, the bot and the C&C server keep maintaining a connection between each other by sending *keep alive packet* represented in step ④. Furthermore, the last process is step ⑤, which the bot will try to find a new victim by doing propagation scanning.

Our methodology aims to inspects network traffic from IoT malware in order to looking for the distinct interaction between C&C and the malware. Thus, our methodology should consider scenarios where **(1) the bot successfully establish communication with the Botnet C&C** and **(2) when this connection is not established**. A non establish communication does not mean that the bot or the malware did not try to contact each other, but for some reason, **the communication cannot process further than step** ①. Thus, we classified those condition as non establish malware.

By considering the two scenarios, we develop two approaches to find the C&C IP based on their connection to infected device. In the following section we describe the details of each approach used to successfully identify the botnet controller of respective malware.

3.2 Bot with established communication

When a bot has an established communication channel with the C&C, it is possible to observe the messages, such as command instruction and identification strings. In this section we explain how we used identification strings in order to achieve our goal in finding their respective C&C IP address. Our experimental evaluation shows that, as soon as the malware run, substantially traffic is performed. Basically this traffic is related to propagation scan ⑤, but we can also find some messages related to the C&C communication. However, in order to distinguishes them from other traffic is quite complex since most of them are TCP flows. Hence, we develop the following steps to identify the botnet controller in infected device traffic. Figure 3.3 illustrates the process that we take in order to get the result.

We start the process by doing the traffic analysis in all of the sample. We analyze the traffic in order to find which address have communicated with the malware (①). The result of this step, will

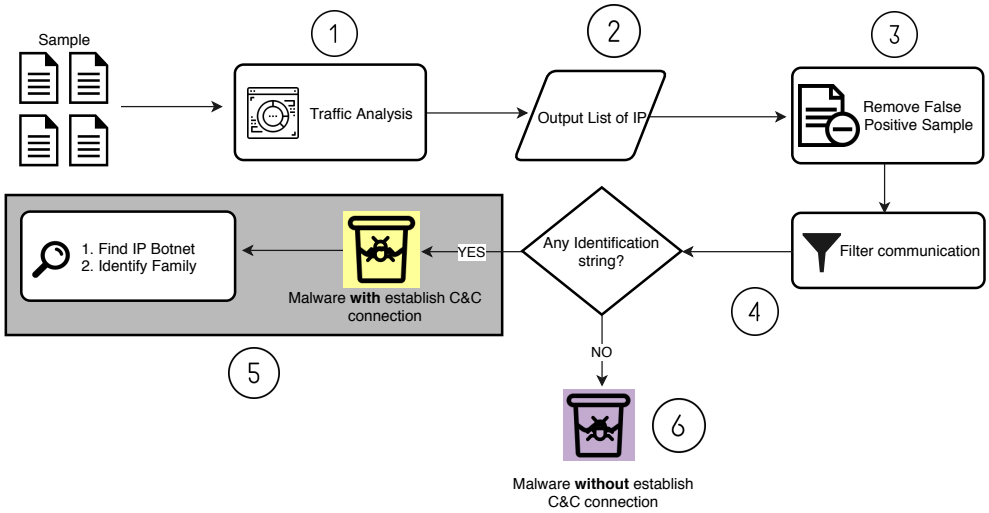


Figure 3.3: Heuristic to identify and classify bot family when established connection is found

return the list of IP that sample try to contact (②). By knowing the result, we notice that **not all of the malware sample have the communication with IP public**. This condition might occurs due to the malware that detects the process is executed on controlled environment or the malware does not have any intention to launch an attack. Thus, it lead us to remove all of the sample that does not contain communication with IP public (③). Next, we process the remaining sample left with filtering communication. We inspect each of sample who have connection with IP public to find whether they have a message that contain identification strings (④). If we found any identification strings in their communication with IP public then we classified them as **malware with establish connection to C&C** (⑤). Otherwise we classified them as **malware without establish connection to C&C** (⑥).

By looking into Figure 3.3, we can also see that by using identification strings, we can identify the IP of C&C Botnet and also their family (⑤). In the following section, we present how this process can be done.

A. Identify the IP of C&C and family of malware

As previously described, to inspect the network traffic of an IoT bot and directly identify its botnet controller can be complex. IoT devices have multiples connections destined to internal and external

networks. Infected devices have even more flows since most of them actively perform scans constantly [8]. Moreover, there is not a generic signature that can be used to filter the communication of the bot with its Botnet controller. Then, we have developed a heuristic to filter suspect communication and identify a C&C channel as we present on Section 3.3. Moreover, by using the identification string, we could extend our analysis by finding the IP of C&C and also the malware family. Figure 3.4 present the process on how we used the identification string on the malware with established connection to find the IP of C&C and the family of malware.

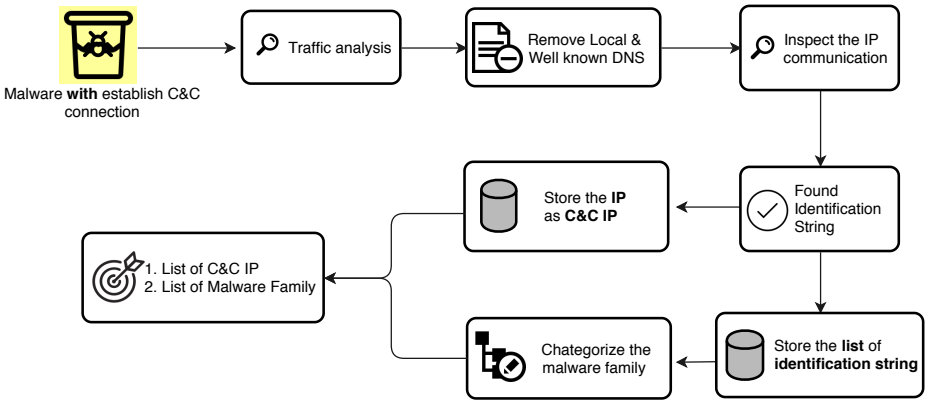


Figure 3.4: Diagram on how to identify C&C IP and family of botnet

In order to identify the IP of C&C and also the family of malware, we start the process by doing traffic analysis in order to single the the communication to external environment. To single out the communication to **public IP**, we remove all IP(s) related to local IP and also well known DNS. After that, we inspect the IP communication in each of the sample to check their identification string. The IP that has identification string in their communication will be stored as the IP of C&C. This step is done in each of the malware with established connection to C&C.

Beside of finding the C&C IP, we also store the list of identification that appear in our sample and make a categorization based on that. In the end, the final goal of this approach will be completed by getting the list of C&C IP and the list of malware family.

Figure 3.5 present the result of our classification. As we can seen, with the approach on the malware with established connection, we can identify the respective malware family and also the IP of C&C. However there is an open question in malware without established communication to C&C. Since we cannot identify their C&C IP and malware family using the identification string.

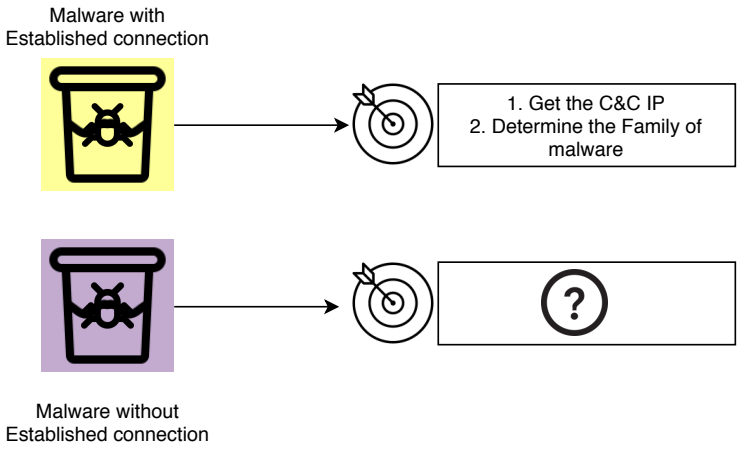


Figure 3.5: Malware sample identification

In the next following section we will discuss about our approach in finding the C&C IP on the malware without C&C established communication.

3.3 Bot with non-established communication

Every malware related to bot has embedded instruction and the required information to contact the respective C&C. When the device is infected, it tries to contact the respective C&C IP. However, for several causes (server is offline), this established connection may not be successful. In the context of network level investigation, we can see this connection, the traffic is overseen due to the number of other packets. Again, there is not a generic signature that can reveal an attempt to connect to the C&C so even though we can see the suspicious connection with keep-alive message, we are not sure whether it is the C&C address or not. In this scenario, the identification is even worst **since we cannot inspect the payload of the packet due to there is no established TCP connection** (We can only see the initial three-way handshake that cannot be differentiated by any other connection).

On next section, we explain the approach we used to identify the sample without established connection. We know that to distinguishes the C&C IP from the normal IP, we need some mechanism to analyze the features of sample. This mechanism should distinguish network traffic from the bot targeting the C&C. This is related to classification problem and it should be addressed by using

machine learning algorithms.

A. Identify the C&C IP using the machine learning

One of the requirements in machine learning implementation is the **needs of knowledge database**. Knowledge database is used as a learning model in determine whether the unseen sample have the same classification with the knowledge or not.**During the experiment, we found that based on the identifier, we can develop the characteristics of each family**. We believe the characteristics of the identified malware can be used as a knowledge to identify the C&C IP address from the malware without established C&C connection.

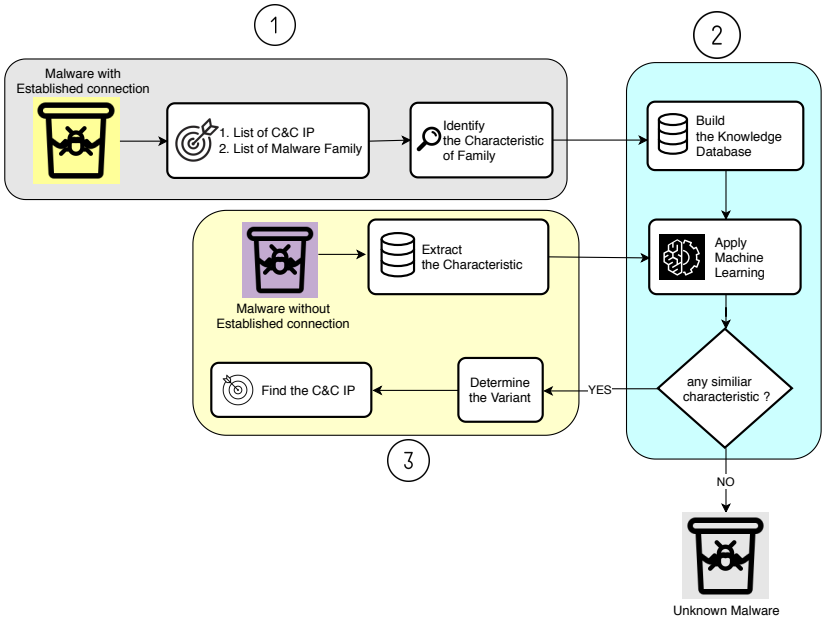


Figure 3.6: Diagram of machine learning implementation

Figure 3.5 illustrates our approach in using the machine learning to identify the C&C IP of malware without C&C established connection. First, we used the result of the previous approach on getting the malware family. We identify the characteristic of each family consist of few parameters (①). These parameters including **number of port connections, number of total packets, number of unique IP address, data byte rate, data bit rate, number of packet size and number of SYN packets**. Then we pick the most representative family as an input in building the knowledge database.

After we have the knowledge database, we can apply the machine learning to find the similarity between the malware without established connection and the knowledge database. But before that, we also need the characteristic of the malware without C&C established connection. So, we need to extract the characteristic of the non established malware.

Beside of **knowledge database**, machine learning also need an **algorithm** in order to make a classification works. In our research different machine learning algorithms can be used and evaluated. We implement several algorithm. Then, we choose the algorithm with the highest of accuracy with the lowest error rate. We also take a note that the accuracy and the error rate of the algorithm is depends on the the number of sample.

If all the process already implemented, machine learning can be running to find the similarity. If the algorithm find any similarity then it can match the characteristic of the malware with the knowledge database . Thus, we will know which sample can be categorized as one of the family from the knowledge database. After we know the variant of the malware without established connection, we can inspect their respective C&C IP by looking into **the first public that the malware try to communicate**. This can be done since the sample of knowledge database has a behavior that all of the first communication to the public IP related to their respective C&C IP. However, if the machine learning can not find the similarity, then the sample will be set Unknown malware.

3.4 Conclusion remarks

Concluding this chapter, we want to highlight several key points in this chapter. Our goal is to find the C&C IP address in order to build the prevention against the malware. We explain that there two approaches that we used in identifying the IoT malware. The first one is an approach to identify the **malware with establish C&C communication** where we explain what kinds of steps that we did in order to find the C&C IP address. This approach will give us the extensive knowledge on C&C IP address origin and how we categorize the variant of malware based on the C&C server instructions.

The other one is an approach to identify the **malware without established C&C communication** where we explained the method when we used machine learning algorithm in matching the

characteristic that can not be seen by using the first approach. The result of this approach will give us the knowledge on how we apply machine learning algorithm in order find similarity between the file that lead us to discover the C&C IP address of C&C server.

Chapter 4

System Design

In this chapter, we present the system design in relation to the approach that we mentioned in the previous chapter. The main goals of this chapter is to present the system design implemented to develop a method to analyze IoT malware. To do so, we structure this chapter into several part that can be seen in Figure 4.1.

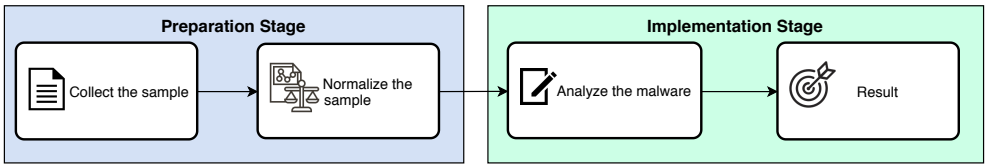


Figure 4.1: Overview of the design

The two stages presented in the figure aims to build a method to identify the IoT malware. In the **preparation stage** we described our challenges to obtain the malware sample and run them to obtain the network traffic performed. In relation to this, we also described the needs to normalize the network traffic from the malware to prevent any bias in our data. In the **implementation stage** we focus in the traffic analysis and described how we used them to analyze the IoT malware traffic.

There is a challenge in obtain diversity of malware sample, this is mainly because the available

repository has only a few variant. We present our scenario to handle the problem in obtaining the sample and also how we normalize the sample in Section 4.1. After we normalize all of the sample, we move into implementation stage. In this stage, we design our system considering the behavior of the malware that we mentioned in Chapter 3. We discuss the steps on how we design the system to analyze the malware in Section 4.2. Finally, to conclude this chapter, we write a conclusion remarks in Section 4.3.

4.1 Preparation stage

As proposed, this stage was develop with the aims to obtain the sample to be analyzed further. Thus, we have to solve how to collect malware samples (binary files) and run them, to get the traffic performed by the infected devices. After that, we need to standardize the network traffic in order to have the same duration.

4.1.1 Collect the sample

Collecting malware samples can be done by using online repositories. However, to collect the performed traffic it is necessary to run the malware. We have addressed this problem by doing two approaches, as follows:

- Run malware samples in our controlled environment and collect the performed traffic;
- Collect network traffic from sandbox endpoints located in Netherlands provided by researchers.

We have combined the both data to create our database to investigate the characteristics of IoT malware. Thus, it leads to the improvement of the accuracy in characterizing IoT malware.

A. First scenario: collect traffic from our Sandbox

Gathering network traffic (PCAP) is one of the main challenges in this research. We use **two different approaches** in collecting the sample. **The first one** is collecting the network behavior

(PCAP) from the malware which runs on our sandbox (see Fig. 4.2) infrastructure. We build our own infrastructure that runs the malware in a controlled environment and collect their network traffic. Our environment based on *Detux* open source sandbox that available on GitHub [39]. To acquire the network traffic from the malware, we have run the malware and set the sandbox in the correct way. Thus, made several modifications on the sandbox so it can run the malware properly.

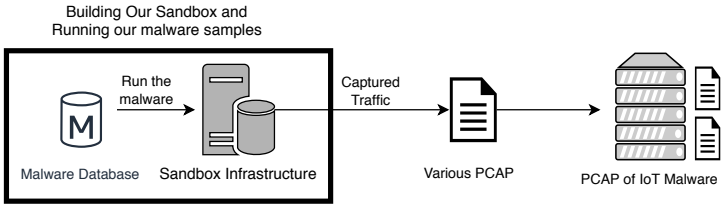


Figure 4.2: First scenario of collecting the network traffic from our sandbox infrastructure

Figure 4.3 illustrates the steps that we did in order to capture the malware traffic. First, we snapshot the sandbox that has been properly configured. This step is needed as a prevention to avoid the infection from executing the malware. After that, we run the network capturing on the sandbox while we execute the malware. During the process, we capture the traffic for 5 minutes. Finally, after all of the process have been done, we restore the sandbox into the initial stage, when the malware is not running. This process will continue until all the malware has been processed and we obtain all of their network traffic.

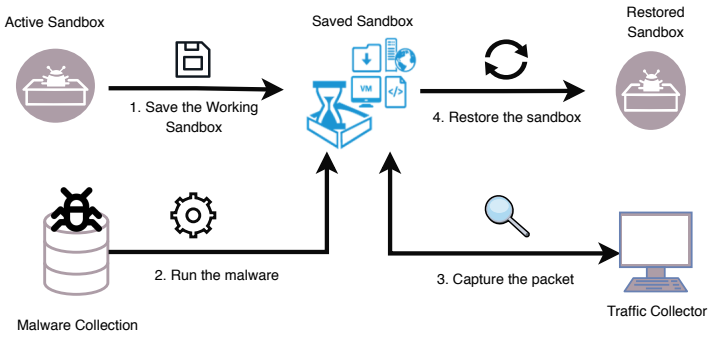


Figure 4.3: Sandbox implementation

Our sandbox infrastructure is based on the operation system Linux and, for capture the traffic, we have used the tool *Tshark* as software to capture the network traffic. Using this approaches, we run 36 different IoT malware variant on the controlled environment. We capture their communication for a certain range of times (in this experiment, we run it on 5 minutes) and collect their network traffic.

However, after we inspect their traffic, we realize there are two constraints from this scenario. First, we only have limited samples thus we cannot use it to do the analysis. The second, some malware samples cannot establish communication with their C&C server, due to the samples cannot get the reply from the contacted C&C server. Since we understand the limitation, we capture this traffic to get an insight into bot communication.

B. Second scenario: traffic provided by researcher

Due to the exposed limitation of our data-set, we complement this with the used of data donation from an external collaborator. This collaborator has a database of malware sample and network traffic from different families for the period of one year.

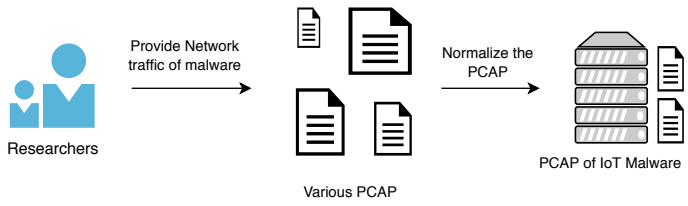


Figure 4.4: Second scenario of collecting the network traffic

We collect 1700 sample of network traffic from IoT malware. This sample was also collected by executing the malware on the controlled environment; however, the implemented network access policy allow them to communicate with their C&C server. The challenge from this scenario are some of the network traffic has been running for different duration of time. Hence, we opted to normalize the data to prevent bias.

4.1.2 Normalization of network traffic

Due to the exposed challenge, we have normalized the network traffics considering a windows time of 60 seconds, 180 seconds, 300 second, and 600 seconds. We realized during the process that in terms of behavior the normalization only take a small effect on the network behavior and there is no differences between the samples that run more than 300 seconds. Thus, we decided to used the sample with the duration of 300 seconds.

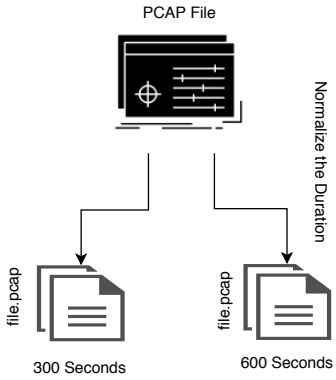


Figure 4.5: Normalization of network traffic

Number of samples	**Size**	**Duration**
1700 samples	14,2 GB %	300 seconds

Table 4.1: Total collected traffic samples

As summarized in Table 4.1, our resulting data-set is composed by 300 seconds of network traffic performed by 1700 unique malware samples. This data-set is not public yet, but we are considering to make it available as soon as we finished with this research.

4.2 Implementation stage

In this section we discuss what will we do after we collected the network traffic sample. As we mentioned, our classification divided the sample into the sample who has establish connection with C&C server and the sample with no establish connection. Following section will present how we investigate those two classification in our research.

4.2.1 Investigating malware with established C&C connection

We mentioned in Section 3.2 there are several steps that we do in order to investigate the malware with C&C connection. We start by determining whether the malware has communication with the outside environment (communication with public IP). This step is needed because IoT botnet needs

an establish connection from C&C to get their instructions. In each of the established connection, every C&C server has an address that try to identifies the host.

In order to investigate the connection to C&C server, we used the *Tshark* command to get all the endpoint's communication. Figure 4.6 show a fragment of the output of provide by *Tshark*.

```
================================================================================================
IPv4 Endpoints
Filter:<No Filter>
                  |  Packets  |   Bytes  | Tx Packets | Tx Bytes | Rx Packets |   Rx Bytes |
192.(Anonymous)        36723      2853611      36555       2705256       168         148355
192.(Anonymous)          285       157459        136        146089       149          11370
80.(Anonymous)            55         3774         27          1860        28           1914
67.(Anonymous)             4          292          1            70         3            222
31.(Anonymous)             3          222          0             0         3            222
32.(Anonymous)             3          222          0             0         3            222
33.(Anonymous)             3          222          0             0         3            222
34.(Anonymous)             3          222          0             0         3            222
35.(Anonymous)             3          222          0             0         3            222
36.(Anonymous)             3          222          0             0         3            222
37.(Anonymous)             3          222          0             0         3            222
38.(Anonymous)             3          222          0             0         3            222
================================================================================================
```

Figure 4.6: Command find the end point communication

There are few columns in the Figure 4.6. The first columns below the *filter* text refer to the IP address which communicate with the malware, second column is *Packets* the column which shows the total number of packets. The third column *Bytes*, refer to the size of the packets. And the *Tx and Rx* columns are the columns which refer to transmit and receive (packets or size) that the malware transmitted or received.

The next process is gathering the list of IP that has established communication. However, as already mentioned we take a note that what we want to find is the communication to the external environment. So here, in this example, we will discard the IP local IP (192.X.X.X)[1] including the domain name system (DNS) requisitions when we perform a finding.

As shown in figure 4.6, the sample has one IP public that has two established communication named 80.X.X.X and 67.X.X.X. We inspect those IP to find if there is an identification string in their communication. If we found the identification string then the malware has a connection with C&C server.

Figure 4.7 shows the result when we try to inspect the communication using *Tshark* command.

[1]This IP address has been anonymized as other IP(s) that will be presented in this section

This result was found when we try to inspect the communication with public IP address. There are several commands that we found in the inspection of sample. Based on the figure, there is a command called **BUILD** at the beginning of the instructions. We have realized that the string **BUILD** reveals the malware variation or family. Thus, we use this semantic to determine whether its has established a connection or not.

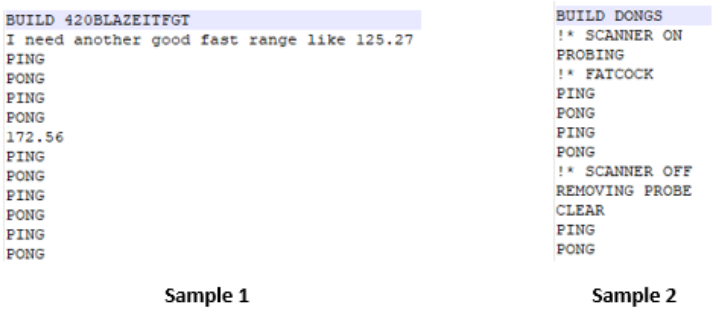


Figure 4.7: Example of command instruction from C&C server

After we determined whether the malware has an established connection or not, the next procedure is to categorize the malware. We list all of the family that appear in all of the data who has connection to C&C IP and classify the malware based on their family. By doing this approach we could categorize the malware. We have found distinct malware variation in our data-set, such as DONGS, ROUTER, RAZER, and GetWrecked.

Using the described approach, we were able to identify C&C communication in 580 samples while in contrast, there are 429 sample that we categorized as no establish connection to C&C. The rest of the data are 691 malware which we categorize as no C&C connection since we cannot find the communication with public IP. This result will elaborate further in Chapter 5.

4.2.2 Investigate the malware without established C&C connection

In order to analyze the malware without C&C established connection, we cannot use the same method as before. Thus, we used the machine learning approach in order to find the similarity and characteristic of the malware. Figure 4.8 elaborates our design to answer the problem.

As we mentioned, our research used two different approaches in investigating the malware. The result from the first approach on the malware with connection to the C&C was useful to train our

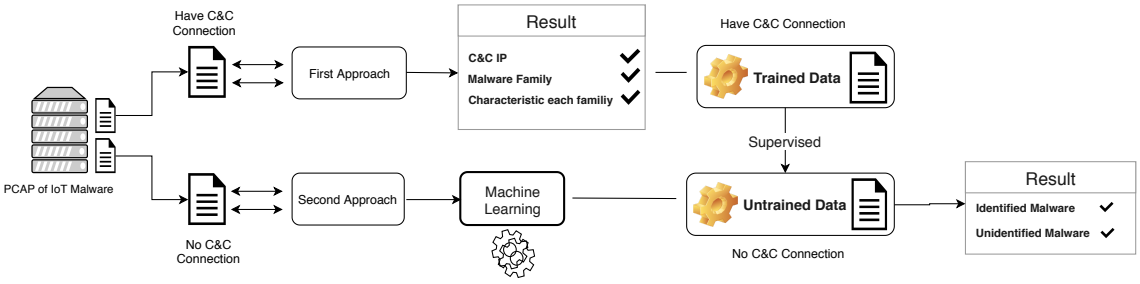


Figure 4.8: IoT malware classification when there is no establish connection

model to investigate the malware without connection to C&C IP. Thus, we build the knowledge data based on the result from first approach.

Knowledge data can be called as trained data in machine learning. Using machine learning, the trained data will be used to supervise the unlabelled data (the data without C&C connection and unknown data). However, there are questions that arise from this method;

1. What features of the network traffic sample are used in machine learning implementation?
2. What algorithm is used to classified the non establish sample?
3. How to find the C&C IP in non establish sample?

Those question will be answered further in the next part of this section.

A. Features of the network traffic sample

In our data (network traffic) there are several features which we used in collaboration with machine learning. These parameters will be used to determine the classification between unlabelled sample. Table 4.2 shows the list of features that we extracted from the network traffic of each sample.

Our labelled data-set has multiples malware families that we successfully classified. Hence, we used the most popular family **DONGS** to be a model to train the sample without C&C connection. After all of the data has been labelled, we will run several test in order to fully supervised untrained data. The next part will discuss how we implement the machine learning algorithm in order to supervised the untrained data (data without C&C connection).

Name	**Explanation**
C&C IP address	refers to the IP address of C&C server
SYN packets from malware	total amount of SYN packets from source
Total packets from malware	the total amount of packets originated from source
Data byte rate	total of data byte rate
Number of packet size	total of packet size
Number of connection	total of connection originated from source
Destination port of malware	refers to type of port
Number of unique IP	total of unique IP that was attemted to be contacted by the malware
Label	type of label (1 or 0)

Table 4.2: List of parameters

B. IoT traffic classification

To supervise the untrained data we proposed to used K-Nearest Neighbors (KNN) algorithm. The decision to use KNN has been done through the research. Before we decided to use the KNN, we did the comparison between three different algorithm. We compare Decision Three, Logistic Regression together with KNN algorithm to predict the accuracy of our prediction. Then we chose the highest algorithm to be used on our data-set. The result of can be seen on table 4.3.

Name	**Accuracy**
KNN Algorithm	92 %
Decision Tree	85 %
Logistic Regression	83 %

Table 4.3: Comparison of algorithm accuracy

KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labelled data-set consisting of training observations (x,y) and would like to capture the relationship between x and y. More formally, our goal is to learn a function h:X $\rightarrow$ Y so that given an unseen

observation x, h(x) can confidently predict the corresponding output y. Finally, after we determine the algorithm to be used, the next step is to implement KNN algorithm to analyzed to data-set.

C. Supervised the untrained data

To apply the KNN algorithm, we use the *Python 3* programming language that runs on the *Jupyter Notebook* application. We labelled the data as 1 (DONGS Family) and 0 (NOT DONGS).

The first thing we need to do is import several libraries then load the data set. The data set is in the CSV format without a header line so we will use pandas *read_csv* function.

```
# loading libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# loading training data
df = pd.read_csv("/analysis_thesis/ml/ml.csv", sep=";")
df.head()
```

Listing 4.1: Load CSV data

After we load the data, we should standardize all of variable so it can be used by KKN algorithm.

```
#Convert the scaled features to a dataframe and check the head of this dataframe to
    make sure the scaling worked.
df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()
```

Listing 4.2: Standardize the data

After the standardization process we will use specific *library* to train a KNN classifier and evaluate its performance on the data set using the 5 step modeling pattern:

1. Split the data

2. Import the learning algorithm
3. Instantiate the model
4. Learn the model
5. Predict the response

The *library* that we used, requires the design matrix X and target vector y be numpy arrays [38]. Furthermore, we need to split our data into training and test sets. We also have to determine how big the data test that we want to use. The *test_size* variable determine how much data we will use as a data test and the rest will goes to data train. The following code does just that.

```
from sklearn.model_selection import train_test_split
X = df_feat
y = df['TARGET CLASS']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state
    =8
```

Listing 4.3: Scikit-learn and split code

For more details, we provides the illustration on how the data being separated into four different type.

	X	X	Y	
	Feature 1	**Feature 2**	**Response**	
X_train	1	2	1	y_train
	3	4	0	
	5	6	0	
X_test	7	8	1	y_test
	9	10	0	

Figure 4.9: Example result of splitting the data

Figure 4.9 illustrates the result when we apply listing 4.3 code. The data will be split into X_train, X_test, y_train and y_test. *Train* represent the trained model on the training step, and *tested* represent the model on the testing set to evaluated how accurate our model.

The next step is implement the learning algorithm using KNN algorithm. The code below will

represent how we implement the KNN algorithm and which number we will chose as neighbors value in KNN.

```
#Using KNN
##Import KNeighborsClassifier from scikit learn.
from sklearn.neighbors import KNeighborsClassifier
#Create a KNN model instance with n_neighbors=1
knn = KNeighborsClassifier(n_neighbors = 5, p = 2, metric='euclidean')
#Fit this KNN model to the training data.
knn.fit(X_train, y_train)
```

Listing 4.4: Implement the algorithm

Finally, following the above modeling pattern, we will define our classifier, we need to fit KKN into our training data and evaluate its accuracy. We will use an arbitrary K and we will use cross validation to find its optimal value.

```
#Use the predict method to predict values using your KNN model and X_test.
pred = knn.predict(X_test)
#Create a confusion matrix and classification report.
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

Listing 4.5: Define classifer

The best K is the one that corresponds to the lowest test error rate, so we have to find which number is the best one. In order to find that, we write a code to do the validation and checking the error rate. One of the most popular method called **k-fold cross validation**.

```
#CHOSING K VALUE
#Create a for loop that trains various KNN models with different k values,
#then keep track of the error_rate for each of these models with a list. Refer to the
    lecture if you are confused on this step.
import numpy as np
```

```
error_rate = []
for i in range(1,50):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
    error_rate.append(np.mean(pred_i != y_test))
```

Listing 4.6: Cross validation

After we find the best neighbors to be implemented, we specify how many times we perform the validation. Thus, we have to find the optimal number of K. The next code will give as the result of evaluation.

```
#Now create the following plot using the information from your for loop.
plt.figure(figsize=(15,6))
plt.plot(range(1,50),error_rate,color='blue',linestyle='dashed',marker='o',
    markerfacecolor='red', markersize='10')
plt.xlabel('no. of K')
plt.ylabel('Error Rate')
# Print out confusion matrix
cmat = confusion_matrix(y_test, pred)
print(cmat)
print('TN - True Negative {}'.format(cmat[0,0]))
print('FP - False Positive {}'.format(cmat[0,1]))
print('FN - False Negative {}'.format(cmat[1,0]))
print('TP - True Positive {}'.format(cmat[1,1]))
```

Listing 4.7: Determining the accuracy

The result of the code will return the best K with the lowest error rate. By doing this machine learning we believe we can discover the untrained data (traffic without established C&C connection) by matching it characteristic with trained data. The final result of this procedure will show which data has characteristics that resemble trained data and which who does not. By doing this approach,

we hope we could detect the C&C IP address of the sample without established C&C connection.

4.3 Conclusion Remarks

Concluding this chapter, our main goal is to design a system that can determine the characteristic of IoT malware which can be useful to find the IP address of C&C server. Figure 4.10 present our detailed system and design. Our design consist of preparation stage and implementation stage.

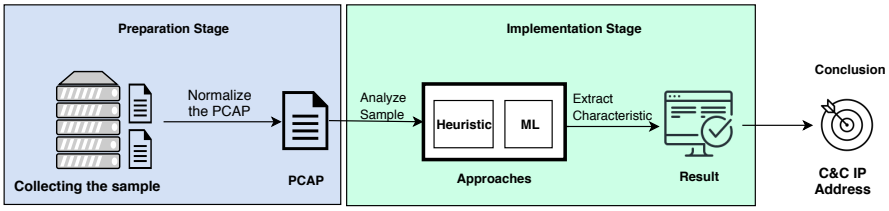


Figure 4.10: Summarizing of system & design

We start the preparation stage by collecting the sample from different sources and those sample has to be normalized in order to prevent the bias. After that, we begin the implementation stage by analyzing the sample to find the C&C IP address of IoT malware. We divided our approach into 2 steps. First is how we identify the network traffic with established C&C connection using heuristic approach. In relation with this approach, we explain how we find the C&C IP and how we determine the bot family. The second approach is related to machine learning implementation to identify the sample without C&C connection. We present how we build the knowledge based on the identified malware and explain how we combine the machine learning algorithm with our approach in order to discover the network traffic without C&C connection.

Chapter 5

Result

In this chapter we present the results from our investigation by describing the characterization of the bot family, the malware families classified by our methodology; and we also examine the C&C IP found. In relation to this result, we also present how many data that we used in this research.

In the first section described in Section 5.1. We present the proportion of the data which can be classified as has established connection with C&C and how much who does not. In relation to this result, we also expose the origin of this C&C IP address to find out from which origin this C&C IP comes from. In Section 5.2, we show the results of the categorization of bot family. We expose the result by displaying a list of family that appears in our data.

Section 5.3, will elaborate the characteristics of specific family who has C&C connection which we used as knowledge data on machine learning. Finally, Section 5.4, will present the result of our methodology in identifying the family and the C&C IP address of IoT malware. In relation to this result, we also present the accuracy of our methodology in classifying the sample.

5.1 Overview of samples

To identify the C&C IP address, we investigate the communication from the infected malware. We elaborate this approach in Section 4.2 in relation to the tools and how we used the result by using *tshark* command to get all the end point communication. Figure 5.1 shows the result of our implementation in classifying the sample.

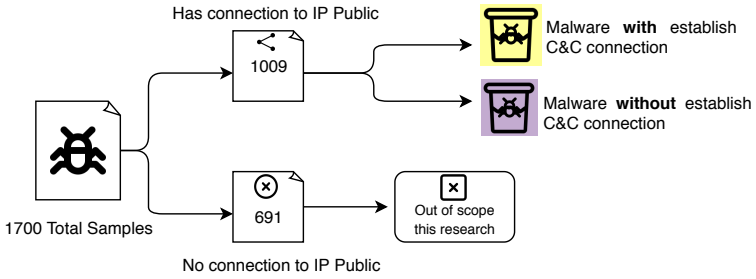
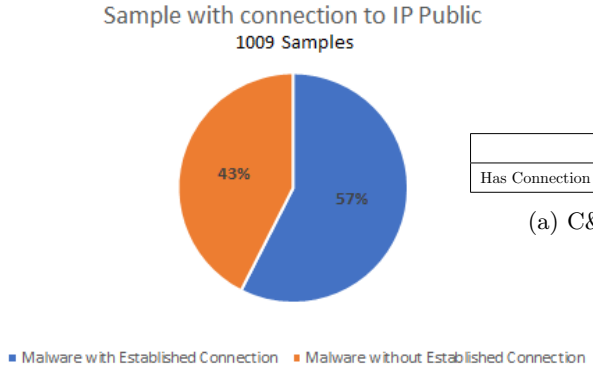


Figure 5.1: Overview of sample

As shown in figure 5.1, our data consist of 1700 network behavior (pcap) that we analyzed individually to find their C&C IP address. The result of this experiment shows that 1009 pcap have connection to the public IP, while 691 sample do not have connection to public IP. Note, samples that do not contact a public IP could be because the malware realize they were run in controlled environment or they do not have any intention to contact the C&C server. We have not investigate this behavior in our research, we only focus on the network behavior from malware that contacted public IP (1009 samples). Thus we only use the 1009 sample for our analysis.



	Established	No Established	Total Sample
Has Connection to IP Public	580 (57%)	429 (43%)	1009 (100%)

(a) C&C connection in number

Figure 5.3: Result of malware traffic classification

From 1009 samples that we analyzed, we can classified **580** samples as **sample with established**

connection to C&C while **429** samples as **non established connection to C&C**. Figure 5.3 summarize our findings.

By identifying the sample with the established connection, we found that there are 580 samples that has communication with their respective C&C IP address. We present in more details about the found C&C IP regarding its location and respective *Autonomous System Number*(ASN) in the next section.

5.1.1 Details of C&C IP from the sample with established connection

In this section we present the top ten list of C&C IP address that appear in sample with established connection. We gather this result by implementing the approach in Section 4.2.1.

No	IP	Counry Code	AS Number	Appears
1	128.X.X.44	US	104	24
2	113.X.X.147	CN	4816	8
3	198.X.X.230	US	36352	8
4	154.X.X.21	GB	61317	8
5	46.X.X.18	NL	43350	7
6	80.X.X.237	CZ	24806	7
7	193.X.X.179	UA	49588	6
8	185.X.X.25	IR	64434	5
9	173.X.X.176	DE	51167	5
10	185.X.X.76	IR	58267	5

Table 5.1: List of C&C IP

By doing this approach, we success to make a list of C&C IP address. This list can be useful to inspect network traffic for malicious activity from these hosts or to notify internet service provider or the system owners about malicious activity in their networks. We are open to share this findings in order to prevent the infection of IoT malware.

As shown in Table 5.1, the top ten list of IP address consist of IP address that comes from different country code. IP address of 128.X.X.44 is the most C&C IP address that appear in our data with

24 connections to different sample. From the list of top ten IP(s) address, we identify the list based on their country code, then we summarize the result to show from which country has the most connections to infected bots. Figure 5.4 shows the distribution of C&C in relation to their country code.

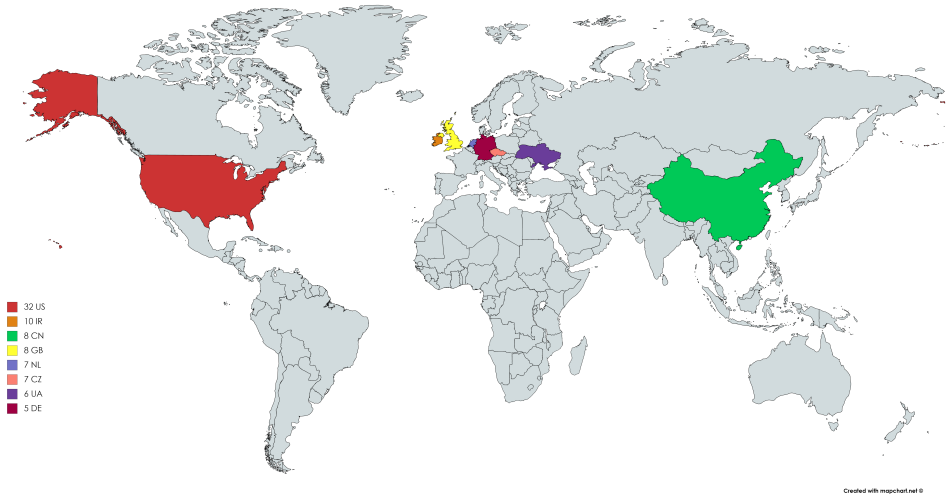


Figure 5.4: Distribution of C&C IP mapped by our solution

As shown on Figure 5.4, there are 8 country code that appear on the top ten list. Country Code US has the highest number for the most number of connection to infected bot with 32 connection. The next country code on the list is IR with 10 number of connection to sample, followed by GB and CN with 8 connections, NL with 7 connections, CZ with 7, UA with 6 connections, and DE with 5 connections.

Using this experiment we found there are **405 unique IP public** which have been used by C&C server. This also means that some of the C&C server can server more than 1 type of family since the total number of established connection is 580 samples. Our finding show that there are 46 C&C server that can serve more than 1 family.

5.2 Categorization of bot family

This section will provide an answer regarding of the categorization on botnet. As previous described, this classification has derived from inspecting the payload string on the network packet of the malware.

We categorize the botnet based on their *BUILD* command that appears on their payload. Since the payload is differ between the variant of the malware, we used it as a feature to categorize the botnet. Figure 5.5 presents our findings.

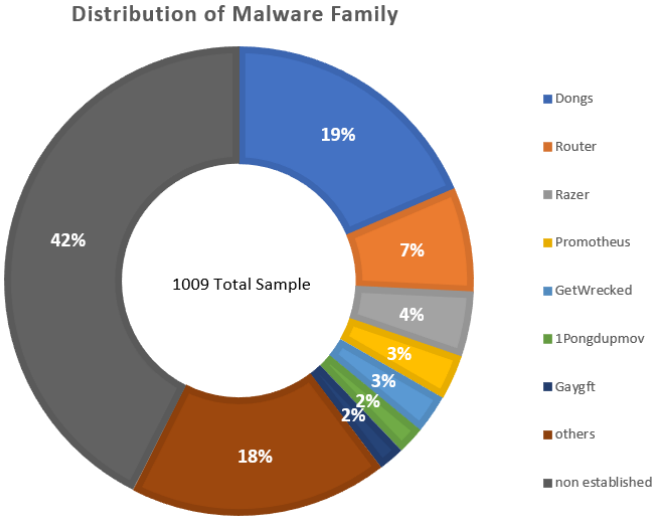


Figure 5.5: Distribution of data with C&C connection

As shown in Figure 5.5, from 1009 data that we analyzed there are 42 % data that we categorize as non established (non-identified) family since we cannot categorize based on *BUILD* command, while the rest are the distribution of the family that we found.

Type of C&C	Classified	Number
Found C&C	Yes	580
Found C&C	No	429
Total data		1009

Table 5.2: Variety of data

By identifying the payload string of the malware, we can deepen the result of the research by finding what family that are most used by C&C server. We briefly mentioned in Section 4.2 that DONGS,

ROUTER, RAZER, PROMETHEUS and GetWrecked are the most families that appear in our result. Here we present the remaining result of the family that appear in our research.

Command	Count	Percentage
BUILD DONGS	187	19%
BUILD ROUTER	73	7%
BUILD RAZER	45	4%
BUILD PROMETHEUS	31	3%
BUILD GetWrecked	27	3%
BUILD 1PONGDUPMOVE	19	2%
BUILD GAYGFT	18	2%
Others	180	18%
Non Established	429	42%
TOTAL	1009	

Table 5.3: Distribution of malware family

Our result in Table 5.3 identify all the family based on their payload. The top 5 family that appear in our data set are *DONGS* with 187. Followed by *ROUTER* with 73, *RAZER* with 45, *PROMETHEUS* with 31 and *GetWrecked* with 27. We also categorize other family as shown in Table 5.3 such as *GAYGFT,1PONGDUPMOVE, etc.*

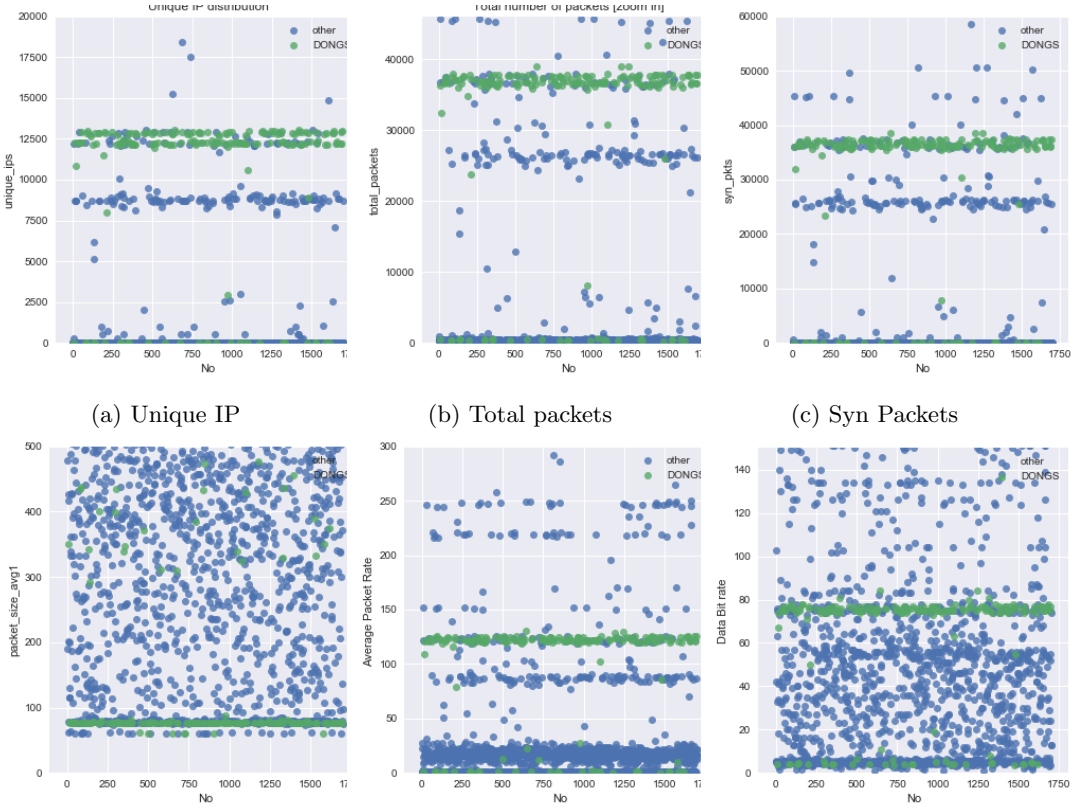
Despite the categorization based on their string command, there are 429 network behavior that we failed to identified. This data will be analyzed further using the machine learning method in Section 5.3.

5.3 The characteristics of specific family

In this section we describe the results from our solution to classify malware without established connection to C&C server. To do so, we used the machine learning as described in Section 4.2.2. We build the knowledge database from the malware that we have successfully classified. As a result, we used DONGS as the knowledge database since DONGS is the most popular family in our data-set. We also found some correlation that can differentiate the DONGS with the others family. Figure 5.6 and Figure 5.7 present our result.

We used several parameter that we mentioned in in Chapter 4. This parameter including total

number of packets, number of unique IP, number of SYN packets, total packets, average packet rate and data byte rate, and data bit rate. We measure the result of each parameter on DONGS class and compare it with other variant.



(a) Unique IP (b) Total packets (c) Syn Packets

(d) Average packets (e) Average packet rate (f) Bit rate

Figure 5.6: Dongs characteristics

Figure 5.6 present the result of our analysis from *DONGS* family. As shown in Figure 5.6, the characteristics of DONGS compare to others class can be seen in *green* colour. DONGS has a largest number of Unique IP that they tried to contacted. It also has relatively higher total packets compare to other variant. In terms of SYN Packets, DONGS relatively in the range of 30000 - 40000 packets, this result also correlate with the characteristics of total packets from DONGS family that reach 30000 - 40000 packets.

In terms of average packet size, DONGS has a relatively smaller packet size compare to other the malware. The average packet size of DONGS is below 100. However the packet rate is relatively higher compare to the others. Meanwhile, In the bit rate field, DONGS has a variation number,

some of the value in the range of 1 to 20 while the rest in the range of 60 - 80.

Another interesting characteristic from DONGS family is all the first IP public that they tried to contacted always correlated to the C&C IP. Thus characteristic makes it easier in determining the DONGS class.

By doing this research, we focus to classify variants from family DONGS. We believe this approach can also be implement in other variants of *BUILD* family. However, it is difficult to makes the proper characteristics from other family since the sample from other family is less compared to DONGS.

5.4 Analysis using machine learning

In the previous section we present behavioral pattern of the variant DONGS. In this section, we show how we can use machine learning to infer this pattern automatically and identify this type of malware among others in the network level.

The characteristics of DONGS family will be used as supervised data and the sample without C&C connection will use as untrained data.

Type of Variant	Function	Number	Label
DONGS Variant	Supervised Data	187	1
Unknown Variant	Untrained Data	429	0
	Total Data	616	

Table 5.4: Machine learning data-set

Table 5.4 shows the data which we used during the analysis. There are two types of data which we labelled as 1 and 0. When we use the machine learning to find our goal, we make a prediction whether some of the labelled 0 can be classified as labelled 1 or not.

5.4.1 Result of C&C IP address using the machine learning

In the process of implementing the KNN machine learning algorithm, we should take a note on several questions:

1. How big the test data size?
2. How we separate the data into train and test data?
3. How can we chose the best K-Value?

We mentioned in section 4.2.2 the size of the data depends on the value that set in the code. In our research, we set the value of the data test equal to 60% of the total data. It means 40% of the data is set to data train.

We split the sample as we described in Section 4.2.2 and find the best K-Value with the highest accuracy and the lowest error rate. The code that can be seen in Appendix A. Figure 5.7 present the result which value is the most suitable for this research.

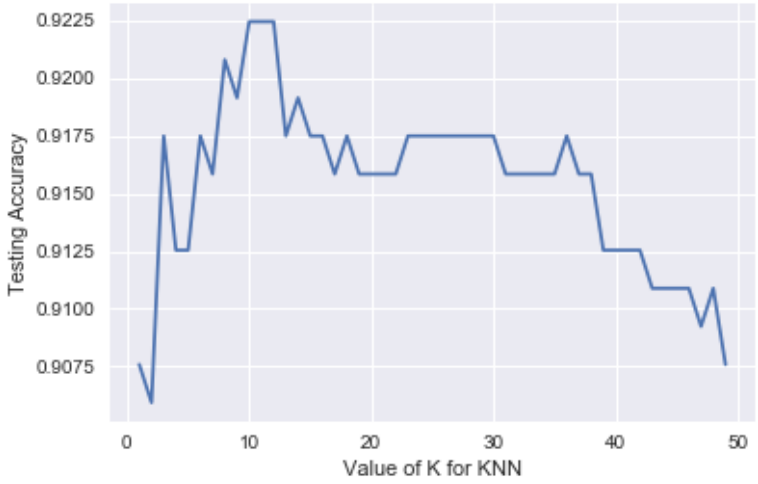


Figure 5.7: Accuracy of K value

Figure 5.7 shows that the best K value in terms of accuracy is 10. Value 10 can increase the accuration into 92,25%. It means our approach using machine learning can make a prediction with the maximum accuracy of 92,25%. Figure 5.8 present the error rate of the K-Value. It shows that the value equals 10 has the lowest error rate using with 7,75 %.

By knowing the accuracy and error rate of our system, we process the sample to find the characteristics that match the DONGS variant. **Using our system we predict there are 23 out of 429 non established samples that match the characteristics of DONGS variant. This means**

we can identify them as a DONGS Family.

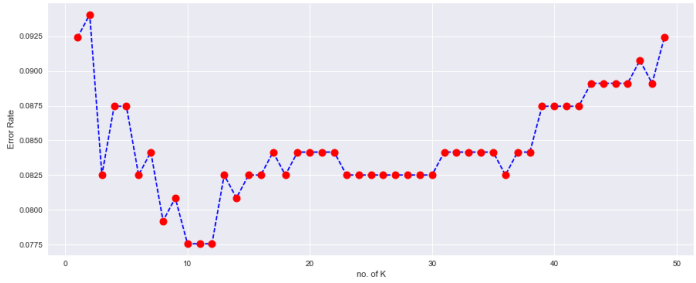


Figure 5.8: Error rate result

Using this assumption, we re-check the samples again and find the C&C IP address in the new classified sample by inspecting the first IP public address that the samples try to contacted (similar to DONGS characteristic). Using this result we conclude the result as can be seen in Figure 5.9.

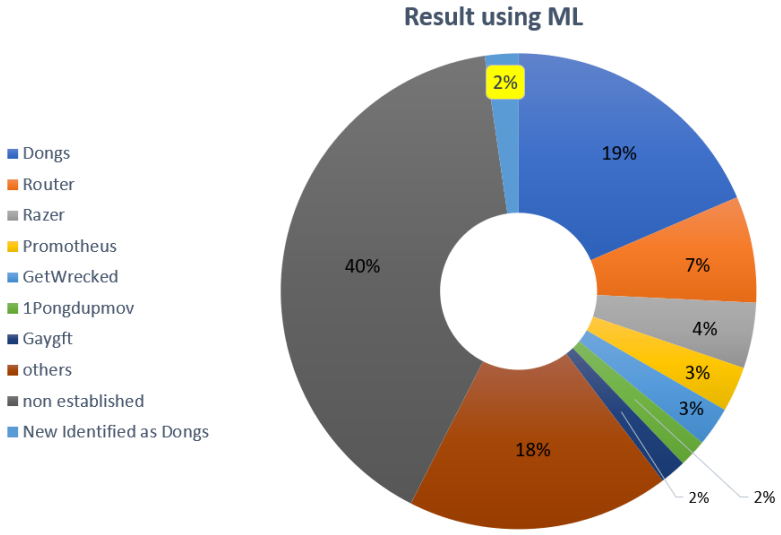


Figure 5.9: Distribution family after ML implementation

Our approach can identified the unknown variant from the non established sample. Using the prediction the number of identified DONGS family increase from 187 (19%) of the total samples into 210 (21%) of total samples. This result correlated with the number of C&C IP that we found from DONGS variant by checking the first IP public that the malware try to contact. Thus it makes the C&C IP that we found from the sample also increase from 580 connection into 603 connection

of samples however the number of unique IP is same with 405 unique IP.

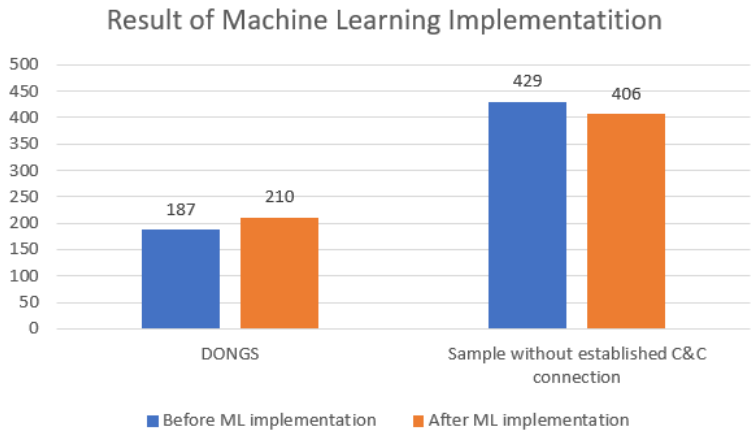


Figure 5.10: Comparison before and after ML implementation

To be more precise, we present the result with the comparison before and after machine learning implementation on Figure 5.10. As present in the result, we able to identify the sample without established connection to C&C server as DONGS family. Thus, it increases the DONGS family and reduces the number of sample without established connection to C&C server.

5.5 Conclusion remarks

During the exploration we manage to present the result of our experiment. This result including but not limited to the origin of C&C IP address, categorization of malware and also the characteristic of malware family. Our research conclude that our approach on the unknown data could be use with the accuracy of 92%

During this experiment we also find few interesting points that helpful in our research such as;

1. There are 46 of C&C server which can serve more than 1 variants;
2. The number of unique IP address of C&C server is 405 IP(s);
3. Some of malware will keep trying to connect to different IP C&C if they cannot establish communication while the others just keep trying in one address;

4. Generally C&C IP is the top talker with the more number of connection destined to endpoints communication;

5. Aside from DONGS family, it is difficult to find the pattern of other family due to the small amount of result and no clear pattern;

Chapter 6

Conclusion

The increasing popularity of the Internet of Things (IoT) have made IoT devices a powerful amplifying platform for hackers [14]. Hackers target the IoT device by build the malicious software called malware [14]. In recent years the massive growth of IoT malware has been in line with the number of attacks to a system [40]. Malware that impacts IoT systems, however, operates a bit differently than traditional malware. Furthermore, to avoid detection IoT malware changes behavior or feature set very frequently. Considering this behavior, technique that have been used to in the past few years to detect IoT malware still facing obstacle of detecting these new forms of malware. Therefore, there is a necessity to develop an approach to identify the IoT malware that able to prevent the attack of IoT malware.

In this thesis we propose a hybrid approach to understand IoT malware behavior in order to identify the communication channel of the botnet. With this, we can identify the address of the malware C&C server, find the characteristic of the malware and classify their family. This is important because by knowing the address of C&C server and the malware behaviors. it can help us in preventing the attack of IoT malware. The first part of this thesis addressed the state-of-the-art of IoT malware and their classification technique (Chapter 2). That part described the types of IoT malware, the overview of attacks they offer, and their characteristics. Moreover, several techniques in analyzing the malware were explained: the static analysis, the dynamic analysis and the hybrid

analysis. The main finding was we can combine those technique with machine learning algorithm to make an analysis of the malware. However, several issues were found in these works such as high machine learning cost, high cost computing, the lack of testing in real devices and lack of efficiency on detecting new malware. For this reason, we proposed an efficient approach for detecting C&C controller of IoT malware that does not need high computing cost but can offer a high accuracy.

In order to classify the malware family, we used an approach to find a specific protocol messages in their network communication that could differentiate each of the family. We used those message to cluster patterns of communication that leads to malware family. We supplement this approach by adding machine learning algorithm to increase the classification approach that we apply. Through this term, we able to classify the family of 603 sample of a total 1009.

To identify the characteristic of IoT malware, we develop a database consist of the network traffic behaviour the malware. The database which we build compose of 1009 sample of IoT malware, from different malware families collected from 1 year daily basis. We classified the malware based on their family and cluster patterns that differentiate them between each other.

Finally, to address the main goal which is to prevent the malware by identifying their C&C address, we develop a set of hybrid approaches to inspect the malware communication and identify the malware family to find a relation to C&C address. The methodology that we proposed are combination between the heuristic analysis and the machine learning implementation. Based on our approach we were able to find patterns that can successfully find malware families. Our approach shows that the system reach an accuracy of 92% in classifying the malware and was able to identify the C&C server of 603 sample.

In general, this thesis can achieve the research objective and analysis of the data gathered from different sources and it can help to answer the research questions. Considering our main goal in identifying the IoT botnet behavior to prevent its activity. We propose the methodology that we used in identifying the IoT malware and present the results including but not limited to the origin of C&C IP address, categorization of malware and also the characteristic of malware family. Moreover by looking into our findings, we can provide the information about C&C server address to the researcher and network administrator in order to prevent the infection of IoT malware.

6.1 Limitations & future works

In this research, the goal of identifying the C&C IP was achieved with several solutions which have their own flaws. The limitations can be coming from gathering the necessary sample, and classifying the types of sample that appear in our research. The current work was a first attempt towards an approach on identifying the IoT malware using two combinations approach (heuristic and machine learning). This leaves a space for improvement in the detecting and the classification method. In this research, we used a specific subset of messages and protocol to identify messages. If the message was encrypted or we cannot find the message then we cannot differentiate the malware. Thus, it makes the development of knowledge database will likely be failed.

For the future work, the signatures should not only be limited to the specific subset of messages. The research can be expanded by looking into other strings, protocol or even other behavior that can distinguish the malware. Another suggestion for the next research is to automate a system of classification of malware. Currently, some of the approaches in finding the C&C IP was done manually by inspecting each of the sample packet. Thus, there is a need to build an automation code to faster the research.

Reflections and acknowledgments

This thesis is a result of my curiosity regarding IoT malware. During working on master thesis assignment, I realized it was definitely most energy and time-consuming tasks I faced lately, and I am glad it was possible to bring it to the end. Due to several problems, concentration, emotions and time management are my biggest hindrances. During my research, I faced many walls. I had problems with collecting the data, sometimes I was looking into some clues that I never got and became frustrated. But then I realized this is a part of progress as well. And then, I managed to learn how to handle those things in order to complete the master assignment.

Several people and institutions have supported this thesis. I feel extremely thankful for all the encouragement, help and support I have received from each one of them. One of the most important institutions is the government of Indonesia represented by the Indonesia Endowment Fund for Education (LPDP), as it provided me with financial support for my thesis and my study at University of Twente. I also wish to thank my daily thesis supervisor at the University of Twente, Joao Ceron for his continuous support, suggestions, and encouragement, as well as his appreciation of my work during the thesis process. I also want to thank Anna Sperotto for her support during my lowest time, and Carlos Ganan from Delft University for his help in the process of gathering the data.

I feel exceedingly grateful because I have had the opportunity to work with an expert on these studies. Last, I am indebted to my parents, my family, my close friends and my colleagues for their constant love and support during my thesis. Especially for my parents and my family, I will work hard for their healthiness in the future.

Appendices

Appendix A: Sample of malware PCAP

This section contains sensitive data and thus the content is not publicly available. To request the completed version of the content, please contact the author of the thesis.

ID of Pcap	Num of Dst Port	Total packet number	Num of Unique Public IP	Data byte rate	Data Bit rate	Avg Packet Size	Avg Packet Rate	Num of SYN Packet	Mirai Signature	Build Command	Durations
1		443	0	5256 bytes/s	42 kbps	189.98 bytes	27 packets/s	0	FALSE	-	16
2	1	381	2	12 kBps	103 kbps	552.86 bytes	23 packets/s	1	FALSE	Cinex	16,3
3	1	364	1	607 bytes/s	4856 bits/s	478.96 bytes	1 packets/s	2	FALSE	-	287,2
4	1	365	1	677 bytes/s	5423 bits/s	540.99 bytes	1 packets/s	1	FALSE	-	291,3
5	289	902	289	605 bytes/s	4844 bits/s	201.17 bytes	3 packets/s	577	FALSE	GetWrecked	299,6
6	12174	36729	12173	9510 bytes/s	76 kbps	77.64 bytes	122 packets/s	36118	FALSE	GOOGLE	299,8
7	2	424	1	497 bytes/s	3983 bits/s	350.84 bytes	1 packets/s	62	FALSE	DONGS	298,7
8		243	0	7883 bytes/s	63 kbps	519.50 bytes	15 packets/s	0	FALSE	-	16
9	40753	45612	45282	9411 bytes/s	75 kbps	61.84 bytes	152 packets/s	45359	TRUE	-	299,7
10		156	0	4161 bytes/s	33 kbps	427.07 bytes	9 packets/s	0	FALSE	-	16
11	2	406	1	590 bytes/s	4724 bits/s	417.23 bytes	1 packets/s	1	FALSE	RAZER	286,9
12	12249	36269	12248	9351 bytes/s	74 kbps	77.26 bytes	121 packets/s	35944	FALSE	DONGS	299,7
13		285	0	2328 bytes/s	18 kbps	130.78 bytes	17 packets/s	0	FALSE	-	16
14	8699	1150640	8700	445 kBps	3564 kbps	116.18 bytes	3835 packets/s	25440	FALSE	-	300
15		164	0	5245 bytes/s	41 kbps	512.04 bytes	10 packets/s	0	FALSE	-	16
16		348	0	4339 bytes/s	34 kbps	199.63 bytes	21 packets/s	0	FALSE	-	16
17	8698	1032032	8700	399 kBps	3194 kbps	116.09 bytes	3440 packets/s	25653	FALSE	-	300
18	10837	32357	10836	8490 bytes/s	67 kbps	77.64 bytes	109 packets/s	31943	FALSE	DONGS	295,9
19		252	0	5344 bytes/s	42 kbps	339.55 bytes	15 packets/s	0	FALSE	-	16
20		300	0	9145 bytes/s	73 kbps	488.08 bytes	18 packets/s	0	FALSE	-	16
21	8704	847485	8708	177 kBps	1423 kbps	62.99 bytes	2825 packets/s	815129	FALSE	-	300
22	1	387	1	17 kBps	140 kbps	737.83 bytes	23 packets/s	1	FALSE	-	16,3
23	1	443	1	596 bytes/s	4774 bits/s	332.07 bytes	1 packets/s	1	FALSE	GetWrecked	246,5
24	1	479	5	583 bytes/s	4667 bits/s	362.18 bytes	1 packets/s	1	FALSE	GAYGFT	297,3
25	1	348	1	512 bytes/s	4096 bits/s	426.06 bytes	1 packets/s	1	FALSE	GEMINI	289,5
26	1	507	1	16 kBps	130 kbps	518.68 bytes	31 packets/s	1	FALSE	-	16,1
27		282	0	11 kBps	89 kbps	633.61 bytes	17 packets/s	0	FALSE	-	16
28	37151	73959	37150	18 kBps	149 kbps	75.80 bytes	246 packets/s	73650	FALSE	-	299,9
29		364	0	4229 bytes/s	33 kbps	186.05 bytes	22 packets/s	0	FALSE	-	16
30	1	325	1	552 bytes/s	4419 bits/s	504.01 bytes	1 packets/s	61	FALSE	DANKv1	296,5
31	1	329	1	535 bytes/s	4285 bits/s	429.10 bytes	1 packets/s	3	FALSE	-	263,5
32		418	0	10 kBps	81 kbps	390.41 bytes	26 packets/s	0	FALSE	-	16

33	12210	36705	12209	9508 bytes/s	76 kbps	77.68 bytes	122 packets/s	36276	FALSE	GAYGFT	299,9
34		347	0	7224 bytes/s	57 kbps	333.34 bytes	21 packets/s	0	FALSE	-	16
35	2	2075729	2	779 kBps	6236 kbps	107.06 bytes	7281 packets/s	2	FALSE	-	285,1
36	12291	37018	12291	9544 bytes/s	76 kbps	77.17 bytes	123 packets/s	36560	FALSE	DONGS	299,3
37		288	0	4004 bytes/s	32 kbps	222.64 bytes	17 packets/s	0	FALSE	-	16
38		276	0	5504 bytes/s	44 kbps	319.31 bytes	17 packets/s	0	FALSE	-	16
39		223	0	5049 bytes/s	40 kbps	362.54 bytes	13 packets/s	0	FALSE	-	16
40	12266	36563	12265	9413 bytes/s	75 kbps	77.24 bytes	121 packets/s	36235	FALSE	DONGS	300
41	1	346	1	639 bytes/s	5119 bits/s	546.71 bytes	1 packets/s	1	FALSE	ROUTER	295,6
42	12908	37467	12911	9700 bytes/s	77 kbps	77.67 bytes	124 packets/s	36931	FALSE	ART	300
43	2	437	1	691 bytes/s	5528 bits/s	463.21 bytes	1 packets/s	1	FALSE	SCREAMS	292,9
44		200	0	4958 bytes/s	39 kbps	396.94 bytes	12 packets/s	0	FALSE	-	16
45	12295	36672	12298	9447 bytes/s	75 kbps	77.28 bytes	122 packets/s	36160	FALSE	DONGS	300
46	12256	36850	12259	9500 bytes/s	76 kbps	77.28 bytes	122 packets/s	36389	FALSE	DONGS	299,7
47		192	0	3611 bytes/s	28 kbps	301.11 bytes	11 packets/s	0	FALSE	-	16
48	1	339	1	586 bytes/s	4690 bits/s	478.96 bytes	1 packets/s	1	FALSE	-	276,9
49	31232	2119616	31232	427 kBps	3421 kbps	60.52 bytes	7065 packets/s	65641	FALSE	ROUTERZ	300
50		207	0	2094 bytes/s	16 kbps	161.96 bytes	12 packets/s	0	FALSE	-	16
51	1	262	1	551 bytes/s	4409 bits/s	583.76 bytes	0 packets/s	1	FALSE	-	277,5
52	12892	37426	12891	9625 bytes/s	77 kbps	77.15 bytes	124 packets/s	37033	FALSE	DONGS	300
53		327	0	2939 bytes/s	23 kbps	143.93 bytes	20 packets/s	0	FALSE	-	16
54	1	356	1	644 bytes/s	5158 bits/s	540.12 bytes	1 packets/s	62	FALSE	ROUTER	298,2
55		255	0	4362 bytes/s	34 kbps	273.91 bytes	15 packets/s	0	FALSE	-	16
56		213	0	6671 bytes/s	53 kbps	501.53 bytes	13 packets/s	0	FALSE	-	16
57	2	465	1	697 bytes/s	5583 bits/s	445.34 bytes	1 packets/s	62	FALSE	ROUTER	296,7
58	1	358	1	688 bytes/s	5506 bits/s	564.44 bytes	1 packets/s	1	FALSE	UZY	293,6
59		266	0	4439 bytes/s	35 kbps	267.17 bytes	16 packets/s	0	FALSE	-	16
60		165	0	5851 bytes/s	46 kbps	567.78 bytes	10 packets/s	0	FALSE	-	16
61	1	331	1	649 bytes/s	5197 bits/s	560.60 bytes	1 packets/s	1	FALSE	PROMETHEUS	285,6
62	9035	27159	9034	7169 bytes/s	57 kbps	79.19 bytes	90 packets/s	26565	FALSE	RAZER	300
63	12277	36599	12276	9421 bytes/s	75 kbps	77.23 bytes	121 packets/s	36230	FALSE	DONGS	300
64	1	329	1	625 bytes/s	5000 bits/s	544.33 bytes	1 packets/s	1	FALSE	1PONGDUPMOVE	286,5
65		222	0	2595 bytes/s	20 kbps	187.19 bytes	13 packets/s	0	FALSE	-	16

66		259	0	6319 bytes/s	50 kbps	390.64 bytes	16 packets/s	0	FALSE	-	16
67		432	0	10 kBps	87 kbps	404.07 bytes	26 packets/s	0	FALSE	-	16
68	33236	66548	33236	17 kBps	136 kbps	76.73 bytes	221 packets/s	66079	FALSE	HERAV2	300
69		196	0	5288 bytes/s	42 kbps	432.01 bytes	12 packets/s	0	FALSE	-	16
70	1	338	1	678 bytes/s	5424 bits/s	578.16 bytes	1 packets/s	10	FALSE	-	288,2
71		228	0	5508 bytes/s	44 kbps	386.79 bytes	14 packets/s	0	FALSE	-	16
72	2	372	1	484 bytes/s	3879 bits/s	381.15 bytes	1 packets/s	1	FALSE	VOID	292,4
73		301	0	3532 bytes/s	28 kbps	187.89 bytes	18 packets/s	0	FALSE	-	16
74	32380	64984	32383	16 kBps	133 kbps	76.95 bytes	216 packets/s	64486	FALSE	SCREAMS	299,7
75		262	0	1987 bytes/s	15 kbps	121.45 bytes	16 packets/s	0	FALSE	-	16
76	1	411	1	481 bytes/s	3850 bits/s	332.61 bytes	1 packets/s	1	FALSE	-	284
77	32491	65547	32495	16 kBps	133 kbps	76.39 bytes	218 packets/s	65015	FALSE	1PONGDUPMOVE	300
78		243	0	14 kBps	115 kbps	953.35 bytes	15 packets/s	0	FALSE	-	16
79	12244	36733	12243	9588 bytes/s	76 kbps	77.70 bytes	123 packets/s	36379	FALSE	GAYGFT	297,7
80	2	329	1	516 bytes/s	4131 bits/s	434.12 bytes	1 packets/s	1	FALSE	DONGS	276,5
81		201	0	2594 bytes/s	20 kbps	206.68 bytes	12 packets/s	0	FALSE	-	16
82	2	244	1	9419 bytes/s	75 kbps	618.01 bytes	15 packets/s	2	FALSE	-	16
83		290	0	5718 bytes/s	45 kbps	315.74 bytes	18 packets/s	0	FALSE	-	16
84	2	325	1	515 bytes/s	4122 bits/s	438.66 bytes	1 packets/s	1	FALSE	DONGS	276,6
85		290	0	3486 bytes/s	27 kbps	192.51 bytes	18 packets/s	0	FALSE	-	16
86	1	1033	1	908 bytes/s	7265 bits/s	263.59 bytes	3 packets/s	1	FALSE	PROMETHEUS	299,8
87	40609	45412	45124	9223 bytes/s	73 kbps	60.90 bytes	151 packets/s	45120	TRUE	-	299,8
88	8457	25253	8457	6804 bytes/s	54 kbps	80.83 bytes	84 packets/s	24867	FALSE	-	300
89	12840	37489	12840	9649 bytes/s	77 kbps	77.15 bytes	125 packets/s	37084	FALSE	DONGS	299,7
90		362	0	9476 bytes/s	75 kbps	419.12 bytes	22 packets/s	0	FALSE	-	16
91	1	457731	3	96 kBps	774 kbps	60.36 bytes	1603 packets/s	1	FALSE	-	285,4
92		440	0	10 kBps	80 kbps	366.08 bytes	27 packets/s	0	FALSE	-	16
93	12998	37942	12997	9756 bytes/s	78 kbps	77.08 bytes	126 packets/s	37444	FALSE	DONGS	299,8
94	2	391	1	709 bytes/s	5678 bits/s	516.81 bytes	1 packets/s	1	FALSE	ROUTER	284,7
95	37353	468257	37353	97 kBps	780 kbps	62.52 bytes	1561 packets/s	74467	FALSE	-	300
96	36458	73257	36458	18 kBps	149 kbps	76.54 bytes	244 packets/s	72928	FALSE	PROMETHEUS	299,7
97	1	408	1	831 bytes/s	6651 bits/s	590.54 bytes	1 packets/s	1	FALSE	PENIS	289,8
98	2	658	5	590 bytes/s	4721 bits/s	268.54 bytes	2 packets/s	62	FALSE	GAYGFT	299,4

99		288	0	4880 bytes/s	39 kbps	271.32 bytes	17 packets/s	0	FALSE	-	16

Appendix B: Source code of machnine learning analysis

```
# coding: utf-8
# # Applying Machine Learning on the dataset

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().magic(u'matplotlib inline')

df = pd.read_csv("C:/Users/dzulq/analysis_thesis/ml/ml_fix_data.csv", sep=";")

df.head(10)

sns.pairplot(df, hue='TARGET CLASS')

# ### Standardize the Variables

from sklearn.preprocessing import StandardScaler
#Create a StandardScaler() object called scaler.
scaler = StandardScaler()


#Fit scaler to the features.
scaler.fit(df.drop('TARGET CLASS', axis=1))

df.dtypes

#Use the .transform() method to transform the features to a scaled version.
scaled_features = scaler.transform(df.drop('TARGET CLASS', axis=1))

```

```
#Convert the scaled features to a dataframe and check the head of this dataframe to
    make sure the scaling worked.
df_feat = pd.DataFrame(scaled_features, columns = df.columns[:-1])
df_feat.head()

# ## Train Test Split
# ##Use train_test_split to split your data into a training set and a testing set.

from sklearn.model_selection import train_test_split
X = df_feat
y = df['TARGET CLASS']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.6, random_state
    =0)

print X.shape
print y.shape

print X_train.shape
print X_test.shape

print y_train.shape
print y_test.shape

print X
# ## USING LOGISTIC REGRESSION Algorithm

# import the class
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

```

```
59 logreg = LogisticRegression()
60 logreg.fit(X_train, y_train)
61
62 y_pred = logreg.predict(X_test)
63
64 # compare actual response values (y_test) with predicted response values (y_pred)
65 print(metrics.accuracy_score(y_test, y_pred))
66
67 # ## Using KNN Algorithm
68
69 ##Import KNeighborsClassifier from scikit learn.
70 from sklearn.neighbors import KNeighborsClassifier
71 from sklearn.metrics import classification_report, confusion_matrix
72
73 #Create a KNN model instance with n_neighbors=1
74 #knn = KNeighborsClassifier(n_neighbors = 5, p = 2, metric='euclidean')
75
76 knn = KNeighborsClassifier(n_neighbors=15, metric='euclidean')
77 knn.fit(X_train, y_train)
78 y_pred = knn.predict(X_test)
79 print(metrics.accuracy_score(y_test, y_pred))
80
81 print(confusion_matrix(y_test, y_pred))
82 print(classification_report(y_test, y_pred))
83
84 print X_test
85
86 predicted= knn.predict(X_test) # 0:unknown, 1: Dongs
87 print(predicted)
88
89 # ### Try with different K value
```

```
90
91 knn = KNeighborsClassifier(n_neighbors=15, metric='euclidean')
92 knn.fit(X_train, y_train)
93 y_pred = knn.predict(X_test)
94 print(metrics.accuracy_score(y_test, y_pred))
95
96 #Fit this KNN model to the training data.
97 knn.fit(X_train, y_train)
98
99 #Use the predict method to predict values using your KNN model and X_test.
100 pred = knn.predict(X_test)
101
102 #Create a confusion matrix and classification report.
103 from sklearn.metrics import classification_report, confusion_matrix
104
105 print(confusion_matrix(y_test, pred))
106
107 print(classification_report(y_test, pred))
108
109 #CHOSING K VALUE
110 #Create a for loop that trains various KNN models with different k values,
111 #then keep track of the error_rate for each of these models with a list. Refer to the
     lecture if you are confused on this step.
112 import numpy as np
113 error_rate = []
114 for i in range(1,50):
115     knn = KNeighborsClassifier(n_neighbors = i)
116     knn.fit(X_train, y_train)
117     pred_i = knn.predict(X_test)
118     error_rate.append(np.mean(pred_i != y_test))
119
```

```
120 #Now create the following plot using the information from your for loop.
121 plt.figure(figsize=(15,6))
122 plt.plot(range(1,50),error_rate,color='blue',linestyle='dashed',marker='o',
        markerfacecolor='red', markersize='10')
123 plt.xlabel('no. of K')
124 plt.ylabel('Error Rate')
125 plt.savefig('error_rate_dst.png')
126
127 knn = KNeighborsClassifier(n_neighbors = 10)
128 knn.fit(X_train, y_train)
129 pred = knn.predict(X_test)
130 print(confusion_matrix(pred, y_test))
131 print(classification_report(y_test, pred))
132
133 # Print out confusion matrix
134 cmat = confusion_matrix(y_test, pred)
135 print(cmat)
136 print('TN - True Negative {}'.format(cmat[0,0]))
137 print('FP - False Positive {}'.format(cmat[0,1]))
138 print('FN - False Negative {}'.format(cmat[1,0]))
139 print('TP - True Positive {}'.format(cmat[1,1]))
140 #print('Accuracy Rate: {}'.format(np.divide(np.sum([cmat[0,0],cmat[1,1]]),np.sum(cmat
        ))))
141 #print('Misclassification Rate: {}'.format(np.divide(np.sum([cmat[0,1],cmat[1,0]]),np
        .sum(cmat))))
142
143 # try K=1 through K=25 and record testing accuracy
144 k_range = list(range(1, 50))
145 scores = []
146 for k in k_range:
147     knn = KNeighborsClassifier(n_neighbors=k)
```

```
148     knn.fit(X_train, y_train)
149     y_pred = knn.predict(X_test)
150     scores.append(metrics.accuracy_score(y_test, y_pred))
151
152 # import Matplotlib (scientific plotting library)
153 import matplotlib.pyplot as plt
154
155 # allow plots to appear within the notebook
156 get_ipython().magic(u'matplotlib inline')
157
158 # plot the relationship between K and testing accuracy
159 plt.plot(k_range, scores)
160 plt.xlabel('Value of K for KNN')
161 plt.ylabel('Testing Accuracy')
162 plt.savefig('testing_accuracy.png')
```

Listing 6.1: Machine learning analysis

Bibliography

[1] Mohamed Abomhara and Geir M Køien. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security*, 4(1):65–88, 2015.

[2] Mohammed Ali Al-Garadi, Amr Mohamed, Abdulla Al-Ali, Xiaojiang Du, and Mohsen Guizani. A survey of machine and deep learning methods for internet of things (iot) security. *arXiv preprint arXiv:1807.11023*, 2018.

[3] Mohannad Alhanahnah, Qicheng Lin, Qiben Yan, Ning Zhang, and Zhenxiang Chen. Efficient signature generation for classifying cross-architecture iot malware. In *2018 IEEE Conference on Communications and Network Security (CNS)*, pages 1–9. IEEE, 2018.

[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1093–1110, 2017.

[5] Vitor Hugo Bezerra, Victor G Turrisi da Costa, Sylvio Barbon Junior, Rodrigo Sanches Miani, and Bruno Bogaz Zarpelao. One-class classification to detect botnets in iot devices. In *SBSeg 2018*, pages 43–56. SBC, 2018.

[6] João Marcelo Ceron, Klaus Steding-Jessen, Cristine Hoepers, Lisandro Zambenedetti Granville, and Cíntia Borges Margi. Improving iot botnet investigation using an adaptive network layer. *Sensors*, 19(3):727, 2019.

[7] Andrei Costin and Jonas Zaddach. Iot malware: Comprehensive survey, analysis framework

and case studies. *BlackHat USA*, 2018.

[8] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. Understanding linux malware. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 161–175. IEEE, 2018.

[9] Anusha Damodaran, Fabio Di Troia, Corrado Aaron Visaggio, Thomas H Austin, and Mark Stamp. A comparison of static, dynamic, and hybrid analysis for malware detection. *Journal of Computer Virology and Hacking Techniques*, 13(1):1–12, 2017.

[10] Michele De Donno, Nicola Dragoni, Alberto Giaretta, and Angelo Spognardi. Ddos-capable iot malwares: Comparative analysis and mirai investigation. *Security and Communication Networks*, 2018, 2018.

[11] Neil DuPaul. Common malware types: Cybersecurity 101. https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101, 2012. Accessed: 2019-01-28.

[12] Sam Edwards and Ioannis Profetis. Hajime: Analysis of a decentralized internet worm for iot devices. *Rapidity Networks*, 16, 2016.

[13] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM computing surveys (CSUR)*, 44(2):6, 2012.

[14] Ekta Gandotra, Divya Bansal, and Sanjeev Sofat. Malware analysis and classification: A survey. *Journal of Information Security*, 5(02):56, 2014.

[15] Dan Gooding. Hackers infect 500,000 consumer routers all over the world with malware. https://arstechnica.com/information-technology/2018/05/hackers-infect-500000-consumer-routers-all-over-the-world-with-malware/, 2018. Accessed: 2019-01-28.

[16] Nazrul Hoque, Dhruba K Bhattacharyya, and Jugal K Kalita. Botnet in ddos attacks: Trends and challenges. *IEEE Communications Surveys and Tutorials*, 17(4):2242–2270, 2015.

[17] Luis Eduardo Suástegui Jaramillo. Detecting malware capabilities with foss: Lessons learned through a real-life incident. In *2018 13th Iberian Conference on Information Systems and*

Technologies (CISTI), pages 1–6. IEEE, 2018.

[18] Mwangi Karanja, Shedden Masupe, and Mandu Jeffrey. Internet of things malware : A survey. *International Journal of Computer Science and Engineering Survey*, 8, 07 2017.

[19] Ahmad Karim, Rosli Bin Salleh, Muhammad Shiraz, Syed Adeel Ali Shah, Irfan Awan, and Nor Badrul Anuar. Botnet detection techniques: review, future trends, and issues. *Journal of Zhejiang University SCIENCE C*, 15(11):943–983, 2014.

[20] Ehsan Khoshhalpour and Hamid Reza Shahriari. Botrevealer: Behavioral detection of botnets based on botnet life-cycle. *The ISC International Journal of Information Security*, 10(1):55–61, 2018.

[21] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[22] Ayush Kumar and Teng Joon Lim. Edima: Early detection of iot malware network activity using machine learning techniques. *arXiv preprint arXiv:1906.09715*, 2019.

[23] Kaspersky Lab. Machine learning for malware detection. https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf, 2017. Accessed: 2018-12-23.

[24] Kirti Mathur and Saroj Hiranwal. A survey on techniques in detection and analyzing malware executables. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 2013.

[25] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiotnetwork-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[26] Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of static analysis for malware detection. In *Computer security applications conference, 2007. ACSAC 2007. Twenty-third annual*, pages 421–430. IEEE, 2007.

[27] P Munchaster. Mirai-busting hajime worm could be work of white hat. https://www.infosecurity-magazine.com/news/mirai-busting-hajime-worm-could/, 2017.

Accessed: 2019-01-28.

[28] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov. A method to detect internet of things botnets. In *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 105–108, Jan 2018.

[29] Radware. Brickerbot results in pdos attack. https://security.radware.com/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/. Accessed: 2019-01-28.

[30] Rafael A Rodríguez-Gómez, Gabriel Maciá-Fernández, and Pedro García-Teodoro. Survey and taxonomy of botnet research through life-cycle. *ACM Computing Surveys (CSUR)*, 45(4):45, 2013.

[31] James A Sherer, Melinda L McLellan, Emily R Fedeles, and Nichole L Sterling. Ransonware-practical and legal considerations for confronting the new economic engine of the dark web. *Rich. JL & Tech.*, 23:1, 2016.

[32] Jiawei Su, Danilo Vasconcellos Vargas, Sanjiva Prasad, Daniele Sgandurra, Yaokai Feng, and Kouichi Sakurai. Lightweight classification of iot malware based on image recognition. *arXiv preprint arXiv:1802.03714*, 2018.

[33] Peter Szor. *The Art of Computer Virus Research and Defense: ART COMP VIRUS RES DEFENSE _p1*. Pearson Education, 2005.

[34] Symantec Security Response Security Response Team. Vpnfilter: New router malware with destructive capabilities. https://www.symantec.com/blogs/threat-intelligence/vpnfilter-iot-malware, 2018. Accessed: 2019-01-28.

[35] Sadegh Torabi, Elias Bou-Harb, Chadi Assi, Mario Galluscio, Amine Boukhtouta, and Mourad Debbabi. Inferring, characterizing, and investigating internet-scale malicious iot device activities: A network telescope perspective. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 562–573. IEEE, 2018.

[36] unixfreaxjp. Mmd-0059-2016 - linux/irctelnet (new aidra) - a ddos botnet aims iot w/ ipv6 ready. http://blog.malwaremustdie.org/2016/10/mmd-0059-2016-linuxirctelnet-new-ddos.html, 2016. Accessed: 2019-01-28.

[37] Dolly Uppal, Vishakha Mehra, and Vinod Verma. Basic survey on malware analysis, tools and techniques. *International Journal on Computational Sciences & Applications (IJCSA)*, 4(1):103, 2014.

[38] Jake Vanderplas. Nearest neighbors. https://scikit-learn.org/stable/modules/neighbors.html, 2007. Accessed: 2019-04-28.

[39] Rahul Binjve Amey Gat Vikas Iyengar, Muslim Koser. Detux sandbox. https://github.com/detuxsandbox/detux, 2018. Accessed: 2018-10-28.

[40] Aohui Wang, Ruigang Liang, Xiaokang Liu, Yingjun Zhang, Kai Chen, and Jin Li. An inside look at iot malware. In Fulong Chen and Yonglong Luo, editors, *Industrial IoT Technologies and Applications*, pages 176–186, Cham, 2017. Springer International Publishing.

[41] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A taxonomy of computer worms. In *Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 11–18. ACM, 2003.