TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Jorge Alberto Medina Galindo

GENERATION OF MALWARE BEHAVIORAL DATASETS IN A MEDIUM SCALE IOT NETWORKS

Master thesis

Supervisor: Hayretdin Bahsi
PhD

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Jorge Alberto Medina Galindo

22.04.2019

Abstract

Nowadays, all IoT devices are under the risk of being a part of botnets. In order to address this risk, various intrusion detection methods based on the datasets containing network behavior of the IoT malware were developed. However, these datasets are generated in small-scale environments and limited to specific malware types.

This research has focused on capturing the normal and malicious network traffic in a medium-scale IOT infrastructure. The normal traffic includes the usual activities such as device management and collection of sensed data whereas malicious traffic covers the network activities of the IoT devices under the execution and propagation of selected IoT malware within a botnet structure.

In this research, 80 different IoT emulated devices using containerization technology and 5 physical IoT devices were connected to a IoT management system in a controlled environment. In this network,, malware, Qbot, Mirai and Torii were installed and the botnet structure was established .
A separate network using monitoring tools was set up to collect network data into a centralized logging platform. The obtained network data was converted to a dataset that includes various statistical features of network traffic . This dataset can be used to induce machine learning based intrusion detection systems for discriminating malicious and benign behavior.

The data was generated in a medium-scale network which includes considerably more IoT devices when compared to similar publicly accessible datasets which were generated in small-scale networks. This scale extension enabled us to capture the patterns of malware spread better in a more realistic environment. Additionally, this dataset includes the behavior of the Torii botnet which has not been addressed in any other dataset.

This thesis is written in English and is 77 pages long, including 4 chapters, 20 figures, 1 equation and 14 tables.

List of abbreviations and terms

TUT	Tallinn University of Technology
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
C&C	Command and Control
IoT	Internet of things
IDS	Intrusion Detection Systems
SIEM	Security Information Event Management
DB	Data Base
SQLI	SQL injection

Table of contents

List of figures

List of Equations

List of tables

1 Introduction

Nowadays, the IoT devices are constantly under attack from different threats that could be found across the internet. Some of the most common threats within the IoT devices are the ones known as botnets.

Within the IoT cybersecurity area, the IDS based on behavior has been adopted in different infrastructures as a method of mitigation and detection of these types of threats.

Due to the extensive number of different types of threats that are presented today, this kind of solution does not have all the necessary data sets for the correct detection of threats yet. The current datasets doesn't consider the different types of behaviors that may exist for the creation of the datasets.

This can be a critical point for the detection models since datasets considering behavioral aspects based on a medium infrastructure have not been created yet.

For this reason, this thesis will focus on the next research questions:

1.How to create a medium infrastructure of IoT devices for malware spreading?

2.How to obtain information about the communication and infection methods of Botnets in a medium infrastructure?

3. How to simulate the expected behavior of IoT devices?

4.How to address the Torii botnet in a behavioral dataset?

The main objective of this thesis is to create a dataset that includes the normal expected behavior of some IoT devices and the behavior of malware Mirai, Qbot botnets. This will be achieved by use of the experimental methodology within a lab using a medium infrastructure of IoT devices and the emulation of the expected behavior of IoT devices. Additionally, Torii botnet will be spread to extend the list of current datasets.

This thesis includes the creation of the laboratory, emulation of behaviour, data collection and creation of the dataset that provides the information of the Qbot, Mirai and Torii botnet. Additionally, a preliminary analysis of the most relevant features is provided.

2 Background Information & Related Work

2.1 IoT

IoT or internet of things can be defined as an extension of the internet to different types of devices such as sensors or smart devices, objects, animals or people [30], [73]. This allows to achieve the extension of interconnectivity between internet and simple devices such as locks, spotlights, thermostats, etc.; These devices give the possibility and ability to transfer information without the need of human-computer interaction [14], [30].

The IoT devices were created to have an immense interconnectivity capability using different communication and network protocols. These types of devices were adapted to perform and analyze daily tasks. These tasks were performed originally in devices not thought to be connected to any network for their management or as a source of information for other systems [30].

These adaptation of devices can be observed in the last generation of smart devices that fall within the category of IoT. Thanks to connectivity capability of these devices, it is possible to operate and manage them in a remote way. [14], [30]. It is the case of smart thermostats or smart bulbs where it is possible to perform tasks like: turn them on and off, adjust the temperature and obtain information about their use. These tasks be done by any device with internet access or access to the internal network in where the IoT device is located [30].

Currently the area of the development of IoT devices has been growing in a faster way than expected. For this reason, it is possible to observe a whole new range of software that has been developed focused on IoT devices in terms of connectivity, management and operation [14], [30]. This software has been developed considering the optimal use of resources (processor, memory, etc.) for IoT devices.

2.1.1 IoT Device Architecture

All the different types of electronic devices that require some type of execution or process have the central processing unit CPU. These units have specific architectures depending on how the data is processed. These architectures define the method in which the processors behave to use the data in all the aspects. This characteristic in the IoT devices is applicable as well, because of the use of CPUs to perform tasks, collect and send data [13], [32].Within the IoT devices there can be found many different architectures. The type of architecture generally depends on who is the manufacturer of the device, purpose and capabilities needed.

This section will focus on the main architectures ARM and MIPS which are commonly found on IoT devices.

ARM

ARM stands for Advanced RISC Machine. This architecture is part of the family of the RISC architectures for computer processors. This is known for being well accepted in the creation of (SoC) system on chips. ARM itself has different sub architectures that are distributed throughout its catalog of processors. Each different version has adopted different profiles that can be applied to different cases depending on intended use of the device. This type of processors can be seen applied in devices such as the well-known Raspberry pi and multiple cell phones that are based on ARM processors [5].

Within this same architecture, 3 different profiles have been developed - profile A, R and M. These profiles were developed for specific purposes and specific devices. The profile A is mainly focused on application processing and execution of complex areas of computing where better performance is necessary. It is the case of cell phones and central units of cars. The profile R is focused on the implementation or execution of processes in real time and the profile M is mainly focused on the optimization of resources. It is characteristically used in the IoT devices based on this architecture [5].

MIPS

MIPS stands for Microprocessor without Interlocked Pipelined Stages. This as well as ARM is part of the family of RISC architectures. This architecture is known for being used in entertainment consoles, network and IoT devices. One of the main differences of MIPS compared to ARM is wider range of compatibility with other type of devices. This type of architecture has been applied either in low-resource devices or performance focused ones. This architecture provides a great processing performance to users. Additionally, it provides a great compatibility and portability with other architectures in terms of execution [69].

Architectures	**ARM**	**MIPS**
Devices	Raspberry Pi	Nintendo 64
	Cellphones	Tesla Model S
	Smartwatches	Sony PlayStation
	Smart thermostats,	MediaTek 7688

Table 1 IoT devices and Architectures

In the Table 1, it is possible to observe the main architectures and where they can be found or under what type of devices within the IoT area are presented. Due to the device capability and emulation options, the devices within this research will be based on the ARM architecture for virtual devices and in the MIPS architecture for physical ones.

2.1.2 Communication protocols

In the area of IoT development, it can be found some specific communication protocols based on the network protocols IEEE 802.11 and IEEE 802.15.4. The use of these kind protocols depends on the principal purpose and capabilities of IoT devices. There are large number of communication protocols for IoT devices. These protocols are mainly focus in the optimized use of resources such as memory, and network bandwidth. The use of each different protocol depends on the type of application that is intended to be used in the IoT solution and the devices capabilities [1], [23], [24], [34], [62].

In the area of Home Automation, where IoT devices and management systems can be found, it has been observed the use of certain application communication protocols. These protocols provide a certain number of characteristics that are beneficial for the automation and communication among IoT devices.[1], [23], [24].

The communication protocols generally used in the home automation area and IoT management are described in the following section.

MQTT

MQTT stands for MQ Telemetry Transport. This communication protocol is based on subscription and publication of messages. It is based on the concept of having different topics to which the different devices can subscribe to receive new messages or notifications from this. Here the device will receive a notification and will perform the task assigned if it is configured on it. At the same time the devices have the capability to generate new topics and send notifications to this one. This can be used as a notification process from the device to send his status among other applications. The basic infrastructure is the server client infrastructure [1], [22], [45]. Here a MQTT broker that is the server, stored and distributed the messages to the connected devices. The basic communication architecture for the MQTT protocol can be observed in the Figure 1.

MQTT protocol was selected to be used in the tests that were performed in this research. This protocol was chosen for its easy development and the possibility to emulate the protocol.

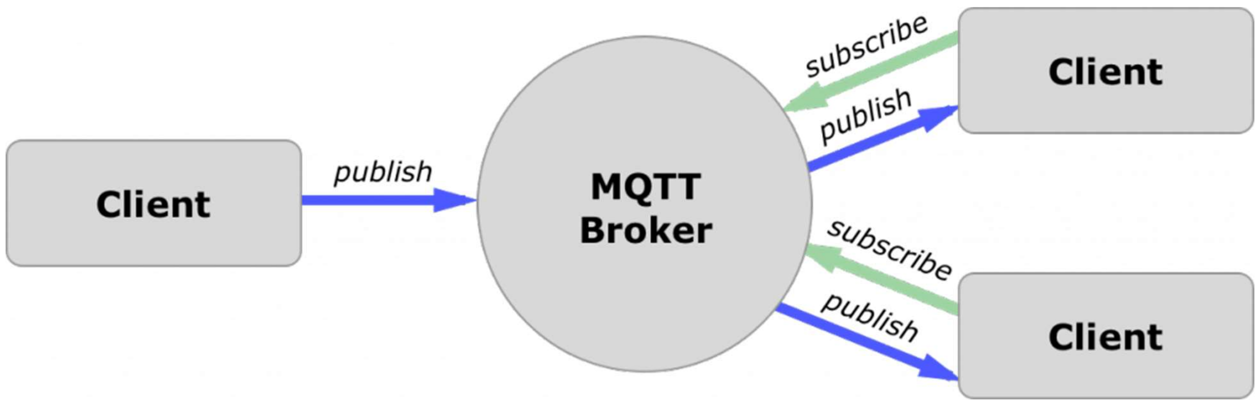

Figure 1 MQTT communication overview taken from 45

CoAP

The Constrained Application Protocol (CoAP) is another commonly used protocol for IoT communication. This communication protocol, unlike MQTT, does not need a broker that manages the messages that are sent. However, same as MQTT, this protocol is based on the communication type machine to machine that can be implemented using subscription and publication messages. CoAP also has the option to be used within the implementation of REST API model where the client can obtain access to resources through the methods get, put, post and delete [1], [11], [61].

2.1.3 IoT device management

The management of IoT devices focuses on a centralized device control system with the ability to retrieve and send information to the IoT devices. This allows the central control module to collect all the information from different devices, store it and manage the devices from a single point. Currently in the management for IoT devices it is possible to observe different types of solutions for a wide range of protocols and communication methods [12], [33], [68].

For the upcoming increase of use of IoT devices, different companies have started to be involved in the development of these kind of solutions. Currently it can be found open source and commercial solutions that allows the efficient management of these devices.

Below there are presented 2 main IoT device management solutions in the open source area.

OpenHAB

OpenHab stands for open Home Automation Bus. This open source solution is specifically designed to handle different IoT devices that can be found in the home infrastructures. This solution can be installed in devices such as the Raspberry Pi and conventional computers. OpenHAB allows interaction among IoT devices in an easy way and allows the development of modules for new IoT systems or for systems in development process. Also, it gives the option to link the content to a cloud platform and have access to devices through this. All these features are native to this software [18], [28].

Hassio

Hassio is a opensource assistant software based on the software “Home assistant”. This software was developed focusing directly on the integration and optimization of the system which run under a Raspberry Pi hardware. This software gives compatibility capabilities with Google home assistant and Alexa. Hassio also allows the integration of different network protocols which are used for IoT connectivity. Due to the raspberry pi hardware limitations, it is necessary to connect additional hardware in order to enable specific communication protocols or features. Hassio allows the automations of tasks for IoT devices through the use of yaml files and python scripts.[25], [26], [48].

Because of the hardware compatibility and the usability of the python language, all tests in the research were performed using the Hassio platform as an IoT device management system.

2.2 Emulation of IoT devices

Emulation of IoT devices involves implementation of different communication protocols and their execution under different architectures. Nowadays there are some commercial solutions that gives the possibility to perform this type of emulation. Those commercial solutions are focused on load tests and implementation of demos which are deployed before production systems. These types of emulated devices are not easily modified since they are built to give the advantage of choosing which characteristics are needed to run the software tests [6], [17], [19], [37].

In general, the emulation of IoT devices can be performed through 2 different approaches, the emulation based on firmware and the Emulation of capabilities.

The Emulation based on firmware is based on the compilation of the firmware used within the IoT device on a different device. In this method the full architecture of the device is emulated in order to compile and execute the firmware. With the firmware compiled and executed is possible to start to perform the functions of an IoT device.

The emulation of capabilities is based on the creation of the different functions in order to perform the tasks that belongs to the IoT device to emulate. Once the functions are created, it is possible to execute the processes of an IoT device. These processes will generate the expected outputs of a physical device [6].

The emulation of capabilities was used within this research in order to get the behavior of the devices. This emulation method was used considering the architecture of the device in where the emulation will be executed. For the implementation of the architecture, the inheritance capability of the containerization technology was used.

2.2.1 Containerization

Containerization is an operating system virtualization method used to launch and distribute applications without need to use a virtual machine for each of applications. In this method, fully isolated applications can be launched on the same host inheriting the main characteristics of the host in which this technology is used, like the architecture of the device [17], [72].

Container

A Container is a standardized software unit used in different containerization solutions. In this kind of solutions, all the dependencies necessary to execute an application are packaged and ready to be used. Container-based systems are identified as a lightweight virtualization method since they use some inherited resources from the host or from other containers besides the use of namespaces for isolation[17], [29], [71].

LXC

Linux Containers is a solution for creation of systems based on a single Linux kernel. This can be considered as a virtualization method for systems without the need to virtualize the host kernel. LXC allows to isolate different virtualized systems under a same host. These characteristics allow the use of containers for different purposes [17], [29].

Docker

Docker is an open source project originally based on the containerization method of LXC. However, Docker has several optimizations and special features which make it possible to be used for other solutions (for example the ability to run containers, facilitate its creation, manage image creation and versions) [17], [71], [82].

Docker offers an implementation model based on images. This allows to share an application or a set of services with all its dependencies in several environments. Docker also automates the implementation of the application (or combined sets of processes that constitute an application) in this container environment [40], [44].

Although, there are other methods of virtualization based on containers such as LXD or Kubernetes, this research will focus solely on Docker. Docker provides simplicity in the development of containers and thanks to the inheritance of the host architecture to containerization technology also bigger adaptability to different kind of hardware based on IoT architectures. This enable the virtualization of IoT systems based on a container infrastructure to be used as a medium virtualized infrastructure. In the Figure 2, it is shown the basic structure of the containers based on the Docker container engine.

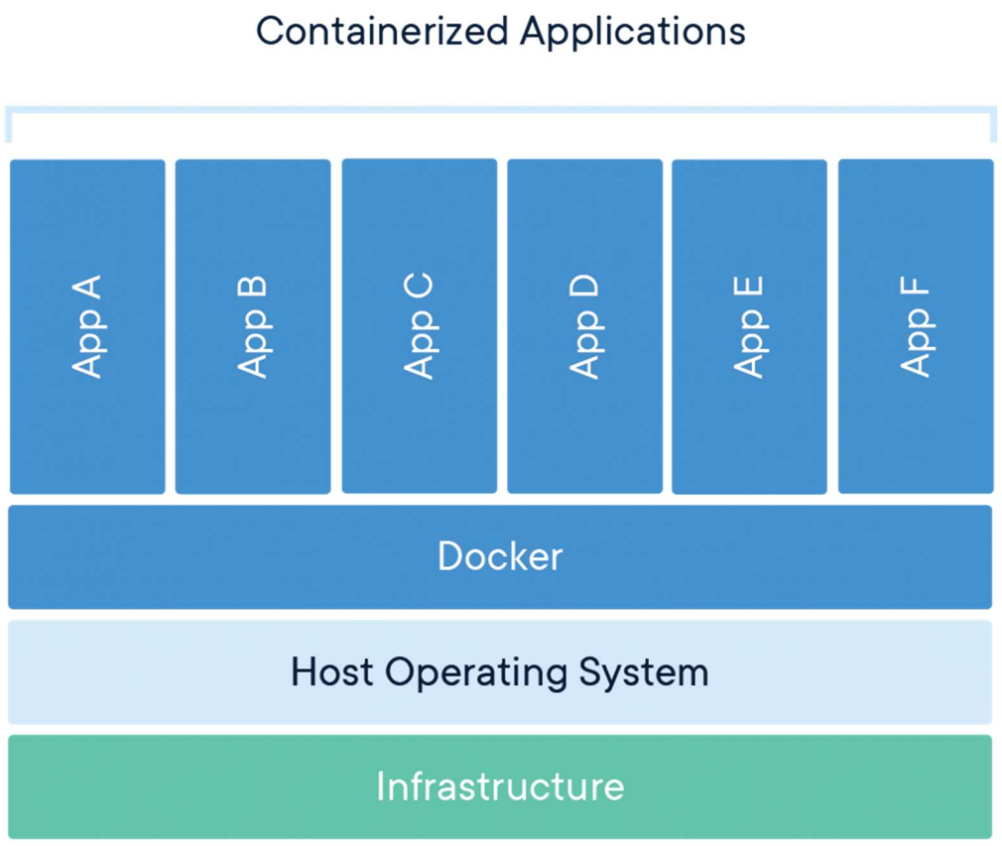

Figure 2 Containerized Application overview taken from [71]

2.3 Cybersecurity for IoT

The massive use and constant development of this type of devices has brought new challenges within the areas of technology, especially in the computer and information security area. These devices can be seen as an extended attack vector if they are not designed, programmed and managed considering the security of devices.

These vectors, creates new opportunities for the exploitation of vulnerabilities in this type of devices. The shared information among the IoT devices together with the lack of secure configuration management, create a risk that grows exponentially [4], [30], [52].

In the cybersecurity area it is possible to find guidelines needed to be followed in order to achieve certain level of security. However, these guidelines do not cover all the different security aspects that may be involved in the development, implementation and management of these devices. Additionally, various vulnerability identification methods have been developed in the area for some communication protocols and IoT devices [57].

Although these guidelines and identification methods have been made to address these types of risks, they are still not able to cover all latent risks and available attack vectors. Unfortunately, the constant development of new devices and technologies created a new and extended scope for risks associated with them [4], [52].

To understand this fact, it is necessary to verify the principal risks that are currently associated with the IoT devices. These risks are constantly exploited to be used as a principal infection method for the IoT devices. For this reason, the principal vulnerability categories that can be found within the IoT area are discussed below.

2.3.1 Vulnerabilities on IoT

Within the IoT protocols, software, hardware and management there can be found a great variety of vulnerabilities that can be exploited, depending on the situation or the use of the devices. Generally, these types of vulnerabilities are known and mitigation methods already exist. However, these vulnerabilities are still latent in many of the devices that are currently on the market [4], [52], [66].

Within this topic we will focus on the main categories of vulnerabilities that can be found in the IoT devices or their software.

Configuration management

In this category, there can be found all the problems that are related to some type of configuration of the device, such as default passwords. The default passwords can be used to compromise the devices using brute force attacks to have unauthorized access to the device. On the other side, vulnerabilities like insecure protocols can be attributed to this category by putting a bit of context on the use of this (for example, the implementation of telnet or ftp protocols that are commonly used within the IoT devices for purposes of debugging). In the case, that these protocol are not disabled in the devices before the release, the vulnerability will fall within the category of configuration management [4], [30], [66].

Input validation

The Input validation category refers to the vulnerabilities that can be exploited by the incorrect use of the inputs. This happens when the applications do not have any method of validation for the type of data inserted in the application. The data can be used to perform malicious activities like the execution of internal programs without the access to the system. Such is the case of the SQLI vulnerability that refers to the exploitation of a SQL DB. In here SQL queries are sent in a way that the system thinks that is being performed by an authorized user. This category can lead to the execution of malicious programs [4], [30], [66].

Another example is the command injection vulnerability. In this vulnerability a command is sent directly to the system. Once the commands are received, they are being executed without any other action needed [4].

These two vulnerabilities can be found in web interfaces of systems that were designed without considering these aspects of security.

The vulnerabilities and categories discussed in this previous section are commonly involved in the infection and propagation mechanisms of malware for botnets.

In this research, all efforts were concentrated in the configuration management category. The configuration management category is commonly used in techniques to spread Botnet malware within the IoT devices.

2.4 Botnets

A Botnet is a system based on the client server infrastructure. In this kind of systems the different Bots are connected to a main server. These servers within the Botnets area are known as a command and control server or C&C. These servers are responsible for sending the different tasks to be processed and executed by the Bots. These assignments can range from simplest tasks such as the execution of a command to more complicated executions such as established connection with another server or the silent installation of some programs or processes.

Originally the concept of Botnet was not associated with a malware. This kind of infrastructure can be seen in different non-malware related systems. The term Botnet could be defined as a set of interconnected devices, which are coordinated by a central management device. These systems are constantly seen on IRC bots [70].

Although the botnets were not originally created for malicious use, it is possible to observe this malicious use in a variety of malicious Botnets. These malicious Botnets are commonly built by the use of malware. Once the malware infects the device, it creates a connection and a process in a background mode which is waiting for instructions that are sent by the C&C server. These instructions may vary depending on how the malware was developed. Generally, the malwares uses unattended propagation and interconnectivity capabilities in order to collect bots. Once the malware is executed in the network, it will try to communicate with other devices to infect and add them to the botnet that is connected to the C&C server creating a complete interconnected infrastructure [70].

2.4.1 IoT botnets

The IoT Botnets are defined as multiple IoT devices interconnected with a C&C server. Since there is a large number of IoT devices and architectures, these types of Botnets are based on the use of specific malware for each different architecture used on IoT devices. This kind of approach generates a greater and better propagation method from the point of view of malicious activity [70], [75].

The main architectures used within IoT Botnet malware attacks are those previously described in this document: ARM, MIPS.

2.4.2 History of IoT botnet

The IoT botnets have had a great impact since 2014, when one of those that could be considered as a pioneer of the IoT Botnets was witnessed. This botnet was Bashlite or also known as QBOT. The botnet Bashlite is characterized by being used for the infection of IP cameras with the use of brute force attacks. It is speculated that many of its variations started to use Shodan as a principal method to obtain new vulnerable devices. Researchers speculated that this Botnet and its variations reached 96% of the vulnerable IoT devices [42].

2016 was the year in which one of the first major attack of a Botnet happened. This was performed by botnet Mirai. This botnet taking certain characteristics of its predecessor Bashlite and improving its propagation methods was able to infect a large range of devices around the globe. The Mirai botnet caused the highly known Dyn attack, where the DNS servers of the company Dyn were attacked. Mirai performed a DDoS on these servers (servers that were providing the domain names to several companies such as Spotify, Netflix, Facebook) which cause a down to the services provided. In addition, the Mirai source code went public at the end of the year and new variations began to appear on the internet [20], [42].

Some of the most well-known variations of Mirai are Persirai and Reaper. Although Reaper is not fully proved to be based on Mirai, it contains many similarities with it. These new variations of Mirai made a great impact in the IoT security area for the use of new methods of evasion and more advanced exploitation techniques [49][55].

Additionally, new types of Botnets started to appear such as Brikerbot or Torii. Brikerbot is based on the elimination of functionalities of the devices. This botnet makes changes in the system of devices making them useless and it is necessary to perform a re-flash of firmware to be able to use them again. On the other side Torii is a botnet characterized by the establishment of connectivity to its C&C through the Tor network. Due to the standby of Torii botnet, the specific functions of Torii are still unknown [49], [55].

2.5 Intrusion Detection Systems

An intrusion detection system (IDS), is a system focused on detection of threats or attacks that are currently inside of a selected infrastructure connected to the IDS[7], [16], [74]. In this type of systems different identification methods are used in order to detect the intrusions. These methods are generally based on different characteristics (files signatures, network latency, etc.) that are obtained through the analysis of the network packages or through the analysis of the devices in the infrastructure [31].

2.5.1 Intrusion Detection Methods

In the Intrusion Detection Systems different methodologies for detection can be found. However, there are 2 principals methods that are commonly used within the business and academic fields. These methods are based on behavior and anomalies or signatures [7], [16].

Anomaly-based intrusion detection technique.

This technique is based on behavior models. These models use an expected behavior of the devices or network to detect discrepancies. The detection of discrepancies is based on a network packets threshold limited by an expected range or events triggered by an anomalous behavior. These types of detection method is one of the most accurate in the IDS. However, there are currently several problems related to the obtention of valid behavior models. These problems are the lack of available datasets containing the different network behavior belonging to other threats and different kind of infrastructures. [7], [9].

If these models are created correctly, they allow to identify patterns in the behavior of the devices and network. This patterns can detect whether a device is compromised. This is possible for the fact that the device performs an action focused on the exploitation or installation of the malware and creates a flow of data different from those established in the behavior model [9]. However, the incorrect creation of the model can lead to allow malicious traffic to occur without any type of alert.

Signature-based intrusion detection techniques

The signature-based technique is one of the oldest and most used within the IDS devices. It is characterized by having a database of all the hashes or signatures that are specific to a type of malicious file, vulnerability or malware attack. Once these signatures are seen in the network, the IDS will perform the action that is configured (such as an alert or a drop of the network package).These techniques are based on the use of approximate pattern matching methods that allows the detection of malicious files and their variations[7], [15], [36].

2.6 SIEM

The acronym SIEM stands for Security Information and Event Management. This type of software provides capabilities of a log management software, an inter-event and correlation machine. This software provides the ability to analyze information in real time and the use of statistical methods for analysis and extraction of information. Additionally, it allows to create alerts focused on specific events or use cases [2], [56], [76].

In the area of information security, the SIEMs are widely used as a centralized system. SIEMs provides a possibility to find, correlate and obtain information from events that may define possible infections of devices, fraud attempts etc. Currently, there are a large number of SIEMs that have been developed by different companies such as McAfee, Symantec, AlienVault, etc. However, Splunk is currently one of the most used SIEM solutions in the industry. Splunk will be used as a method of information extraction and labeling in this research [2], [56], [59].

2.6.1 Splunk

Splunk is a SIEM software developed by the brand with the same name which specializes in the development of this product and its components. This software provides different features that allow capturing, indexation and correlation of events in real time. These features allow to create complex infrastructures (like cluster based or distributed ones) to gather data by using different collection methods such as Forwarders and agents. This software provides modules for machine learning, behavior analysis and the use of different types of programming languages. In addition, it has its own query language which is optimized for the use of memory and processing power. This query language allows the use of statistical functions and provides the ability to create private functions if it is necessary [2], [10], [77].

Currently, Splunk offers different types of licenses, which are categorized as free, commercial and dev or educational. For the purpose of this research an educational license from Splunk was provided. This license allows the indexation of 500 GB of information for 6 months and gives unlimited access to different tools that Splunk provides within its software like machine learning modules.

Splunk Data Analysis

Splunk enables the use of different commands for data analysis and creation of data-sets (these data-sets can be catalogued or tagged depending on the user's need). The data analysis and data-sets creation capabilities together with the option to execute complex machine learning methods make Splunk to be a suitable tool for the analysis of information on real time [2], [8], [10].

This type of analysis can be performed to obtain information regarding events that trigger specific use cases configured within Splunk. It also allows the extraction of information in different formats and the creation of reports.

2.7 Data Sets for IoT IDS anomaly based

Researches in the IoT IDS anomaly detection have been performed by different institutions like the University of California Irvine or the Czech technical university in Prague. These institutions in their researches focused on the obtention of datasets for IoT devices based on behavior of different botnets. These researches provide a wide range of datasets and achieve a greater scope in the detection of anomalies based on the behavior [43], [63].

Based on these researches, it is possible to create new datasets and also enlarge datasets available in previous researches. Thanks to this a larger database of expected behavior could be created.

Despite the current available datasets, there is still a lack of datasets. This lack is caused by the existence of different network behavior belonging to other threats and the behavior under different infrastructure sizes.

2.7.1 Related work

One of the most relevant studies in this research area has been done by the University of California Irvine. In this research a malware was spread into different IoT devices. During this malware spread, the network traffic was captured and analyzed in order to generate a dataset. In this dataset 115 features were generated from the network statistics. These features were generated by using mathematical functions (such as variance, covariance, mean, etc.) in different time lapses. Additionally, these features were used for the training of an IDS based on Deep auto-encoders to prove the detection capabilities of the dataset [43].

Within this research area there is also possible to find studies that are focused on different Botnets and approaches to generate datasets. Such is the case of the study carried out by the Czech Technical University in Prague where the Botnets Rbot, NSIS, Virut and Neris were propagated and analyzed [63].

This research was focused on different scenarios in where the different Botnets were executed. In each of those scenarios the bidirectional information of the network packets was captured. This information was used to create different datasets by the use of the tool Argus for each Botnet.

In the Table 2, it is presented a list of the most relevant research studies in this area and their characteristics.

Dataset Name	Botnets	No Features extracted	Location
detection_of_IoT_botnet_attacks_N_BaIoT Data Set	Mirai & Qbot	115	https://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT
ISOT Dataset Overview	Storm, Waledac	NA-pcap file	https://www.uvic.ca/engineering/ece/isot/assets/docs/isot-datase.pdf
CTU-13	Rbot, NSIS, Virut,Neris	NA-pcap file	https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html
ISCX-Bot-2014	Storm,Zeus	80	https://www.unb.ca/cic/datasets/botnet.html
IoT host-based datasets for intrusion detection research	Mirai	78	https://www.researchgate.net/publication/328531254_Providing_IoT_host-based_datasets_for_intrusion_detection_research
Android Botnet dataset	TigerBot, Wroba,etc.	NA-pcap file	https://www.unb.ca/cic/datasets/android-botnet.html

Table 2 Related work

2.7.2 medium infrastructure tests

Although previously described studies provided a large number of datasets, there are still some restrictions that may affect the malware detection capability, such as the dataset generation based on medium infrastructures. The majority of these studies have been carried out in smaller infrastructure laboratories (from 4 to 10 devices) [43], [63], [23].However, experiments in medium size infrastructures have not been carried out yet. The key point that will be addressed in this research is the creation of medium infrastructures with 85 IoT devices which will allow to obtain the data set focused on this size of infrastructure.

2.7.3 Emulation of behavior

The emulation of behavior can be performed in different ways that can range from the manual use of the devices imitating the desired environment, to the creation of scripts for the execution of specific functions. The emulation of behavior gives a great advantage for the creation of this type of datasets. If the emulation of behavior is not performed properly, it can generate problems in the performance of the datasets and IDS. This kind of problems are generally associated to the generation of network packets and the non-emulation of a real behavior of the IoT device(such is the case of the experiment performed in the study "IoT host-based datasets for intrusion detection research").[60], [81].

The imitation of an environment could be performed by the gather of the daily use of electricity. Following the information gather in [83] it is possible to estimate the daily hours per bulb in an office environment. In a normal Living room, the research showed a 1.7 hours usage per day in a time frame of 5 hours (being the average time spent in a living room). The information retrieved, it is possible to be used as a base for the emulation of benign behavior. In here the usage hours divided by average time spent in a living room and multiplied for 8 daily working hours provides an approximate usage time in working environments.

The malware behavior in the other side can be emulated by the execution of the different modules within the botnets. The execution of these models will generate network packages within the network, providing a real output of information expected from a botnet

3 Method

3.1 Lab Creation, Behavior and Dataset Generation

The main objective of this research is to generate an IoT behavior dataset for QBOT, Mirai and Torii malware in medium size infrastructures. In the following chapters the used infrastructure is explained.

3.1.1 Lab creation and infrastructure

Network diagram

In the Figure 3, it is possible to observe the diagram of the full laboratory infrastructure and how the different devices were connected in the network.

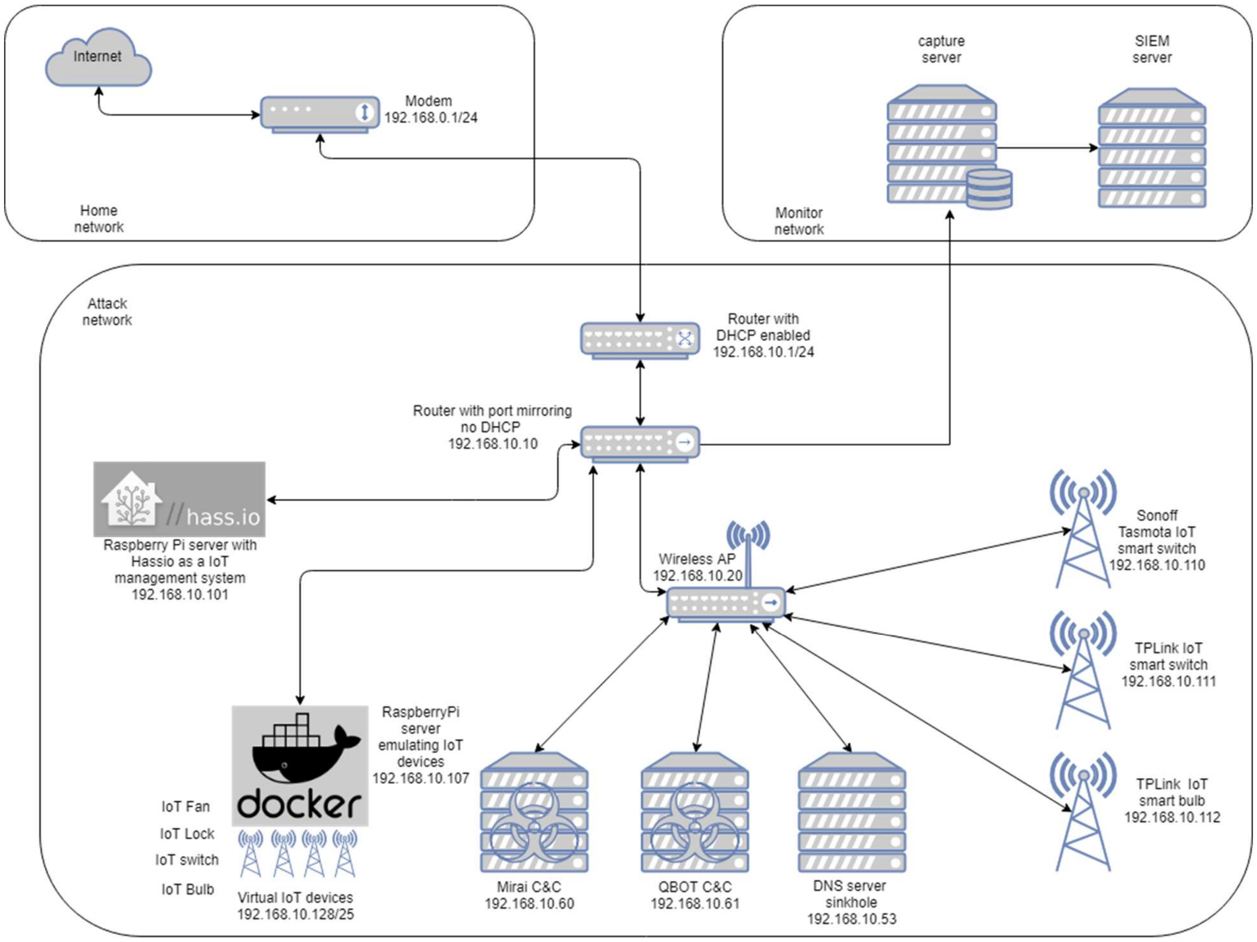

Figure 3 Lab Infrastructure Diagram

The creation of the infrastructure was focused on the recollection capability of network packages from all the endpoints and servers during the propagation of the Mirai, Qbot and Torii botnets. The infrastructure was designed considering 3 different networks: home network, attack network and monitor network.

The home network was directly connected to internet in order to provide internet access for the initial configuration of different devices. A different subnetwork mask to restrict the connectivity between the networks was configured within this network.

The attack network was created with the purpose of contention and spread of the malware . This infrastructure was composed by physical and virtual devices. The virtual devices were created with the use of containerization technology. This network included most of the devices and was used to generate the behavior of the devices during benign and malign traffic.

The devices belonging to this network and their functions are presented below.

Router with DHCP enabled:

This router is responsible for generating the new network segment with network mask "/24". This router using the new network segment enables to create an isolated network which allow communication only between internal devices. This router provides different IP addresses to the devices in the network by the use of DHCP.

Port mirroring Router:

This router is used to capture and transfer network packages among different devices by using port mirroring. Port mirroring is a technology available in some network devices that allows cloning and transferring of network packages. This type of technology is used for network monitoring in real time without affecting the network latency. For the configuration of this laboratory the information of all devices were captured and transferred to the monitor network.

Hassio Raspberry Pi Server

This device serves as an IoT management system of the infrastructure. This system was created by the use of the software Hassio over a Raspberry Pi which will simulate the same network behavior that can be seen on real implementations.

This device functions as a central management system for the IoT devices connected to the network.

Docker Raspberry Pi Server

This device allows to create virtual devices by the use of the containerization engine Docker. This server inherited its architecture (ARM) to the virtual devices by the use of containers. The inherited architecture allows the emulated devices to behave as an IoT device in order to execute the processes.

Wireless AP

A wireless access point was configured and connected directly to the router in order to allow the non-ethernet capable devices to connect to the network. The access point was configured to allow the main router to assign the IP addresses. This configuration will eliminate duplicated IP addresses on the network.

QBOT C&C

This server was the command and control server of the botnet Qbot. In this server FTP and WEB services were installed in order to allow the propagation of the malware. Additionally, the different binaries used for the infection of the devices were compiled in this server.

All the information related to the configuration and compilation of the botnet QBOT is described in detail in the section “Malware Test” under the subsection “QBOT configuration” of this document.

MIRAI C&C

This server was the command and control of the Botnet Mirai within the infrastructure. Same as the QBOT C&C server, this server had the FTP and WEB services installed. The binaries were modified and compiled in the same server.

All the information related to the configuration and compilation of the botnet Mirai is described in detail in the section “Malware Test” under the subsection “Mirai configuration” of this document.

DNS server sinkhole

The DNS server was responsible for the domain name resolution of the Mirai botnet. In addition, this server worked as a sinkhole for the domains to which Torii was constantly trying to establish connection. Thanks to this configuration it was possible to negate the connections between Torii and the domain belonging to his C&C.

Sonoff tasmota basic smart switch

This IoT device is known as one of the best low-cost IoT devices in the market. This IoT device have connectivity capability with different management systems by the use of different communications protocols. Additionally, this device is well known for being easily customizable to add extra features like cloud interaction via MQTT or CoAP.

TpLink smart switch

This device from the brand TpLink use MIPS architecture and MQTT, CoAP communication protocols and provides power consumption info, status, on/off capability and timer options. Additionally, “TpLink smart switch” includes native support to most of the IoT device management systems and firmware updates that provide the connection to external services.

TpLink smart bulb

This device same as the “TpLink smart switch” was developed by the brand TpLink and use MIPS architecture and MQTT, CoAP communication protocols. This device provides light intensity, status, and on/off capability. TpLink smart bulb has native support for the IoT device management system and firmware updates that enable connection to external services.

The monitor network was the last network created within the laboratory. This network contains only 2 devices that are used for storage and processing of the data received from the “Port mirroring Router”.

Capture server

This server was responsible for the recollection and storage of the network packages within the laboratory. This recollection was performed by the use of the tool Tcpdump. Additionally, this tool stored the recollected network packages in a pcap files. These pcap files operate as a source of information for the SIEM server.

SIEM server

This server was in charge of the functions for information indexing, filtering, analysis and dataset creation. These functions were performed by the use of the software Splunk. In this software, different filters were applied in order to differentiate malicious packets from benign ones. Additionally, statistics and machine learning tools were applied within Splunk to create the behavioral dataset.

The previous commented devices were composing the full physical infrastructure of the laboratory. However, 4 types of IoT devices were emulated within the Docker Raspberry Pi Server IoT. This emulation was performed in order to create a behavior dataset based on a medium size infrastructure.

In the following section the emulated IoT devices, emulation method and relevant information are described.

IoT devices

The full infrastructure was composited by 85 IoT devices where 80 out of 85 were emulated. These emulated devices were created by the use of the containerization engine Docker. All the emulated devices were developed considering the basic characteristics that are commonly found in the IoT devices such as the communication protocol and functions. These considerations were taken into account in order to emulate successfully the operation and behavior of IoT devices.

Creation of Containerized Virtual Devices

The emulated IoT devices were created using the python3 image based on Debian as the operating system. This low-resource operating system allows the execution of scripts in Python. Additionally, it gives the option of packages installation within the operating system by the use of the systems tools.

For the emulation of IoT functions, a Python script was developed to execute the functions and behavior of 4 different IoT device types.

These types of devices and their characteristics are described below:

Fan: This smart fan allows the selection of speed, oscillation state, current state fan, turn on and off capabilities.

Lock: This smart lock is characterized by providing the state of the lock, opening and closing capabilities.

Light bulb: It emulates the intensity of light, bulb status, turn on and off capabilities.

Switch: It is the most basic IoT device which gives the status and options to turn it on and off.

Protocol Used

The MQTT communication protocol was used within the Python script in order to provide a communication protocol that would be expected in real behavior. This protocol enabled the communication between the IoT emulated devices and the IoT device management system.

Container python IoT client emulation

The structured format of Dockerfile was used to create the IoT devices in an automated way. This Dockerfile allows to execute software by the declaration of the software as a variable in the file. This Dockerfile also allows the execution of commands and software installation at the moment of the creation of the devices.

All the devices were created with an internal vulnerability belonging to the configuration management vulnerability category (the embedded vulnerability was a weak password in the vulnerable telnet service). The telnet service was installed at the moment of the creation of each device. Additionally, a user with root privileges and a password easy to guess was created on each device. The Telnet service with this vulnerability provided the main botnet infection method to the devices.

Architecture

Thanks to the architecture inheritance capability of the docker containerization engine it was possible to use the ARM architecture within the emulated devices. This architecture in specific can be found in large number of IoT devices. The use of ARM provided the architecture needed to emulate the expected execution behavior of the devices[23].

In the Table 3 the different IoT devices within the infrastructure in the laboratory are described.

Architecture	Type of device	Protocol used	Type of device	States	No of devices created
ARM	Virtual	MQTT	Switch	On/ Off	20
ARM	Virtual	MQTT	Fan	On/ Off/ Oscillation/ Speed	20
ARM	Virtual	MQTT	Light bulb	On/ Off/ Intensity	20
ARM	Virtual	MQTT	Lock	Lock/ Unlock	20
MIPS	Physical	MQTT	Switch	On/ Off	2
MIPS	Physical	MQTT	Light bulb	On/ Off/ Intensity	1

Table 3 IoT infrastructure

3.1.2 Emulation of legitimate behavior

An automated method of execution was created for the emulation of the benign behavior. This method was created by the use of python script using the MQTT protocol and the inherited architecture.

The emulation of behavior was based on the different states listed in the Table 3.

These states can be assigned to the different IoT devices in certain hours of the day by the use of the management system Hassio. The Hassio software provides a native support for automation and scripting.

These automation and scripting capabilities allow to specify a sequence of actions to be executed in the devices connected to Hassio. Once the script was configured, it allowed to assign actions to the specific devices. These actions were assigned depending on a trigger action configured within the script.

Within the laboratory the following triggers were configured in order to obtain the expected behavior:

- All devices were turned on at 8:00 AM.
- Each time a state of a device changed, Hassio started a countdown until the next change of state.
- The countdown was created in a randomized way taking as limit 20 state changes and 3 hours of state "on" per day. This is described in the section "Emulation of behavior".
- All devices were turned off at 7:00 PM.
- Additionally, the triggers were limited to being executed from Monday to Friday to follow a working environment behavior.

These triggers were configured taking in consideration the expected hours in where a light bulb is used per day as explained in the section "Emulation of behavior" This information was obtained by the use of the data provided in [82] as it is described in the section "Emulation of behavior".

By the use of these triggers, network packages were generated across the network in order to send and receive the information. All these network packages were captured and stored. The captured network packages provided the next information:

Time, protocol used, tcp stream, tcp stream size, source IP , destination IP, mac addresses, tcp raw message and response code.

3.2 Malicious behavior generation

The different characteristics that were used for the Botnet propagation tests to generate the malicious behavior are described below. These characteristics are presented together with the dates in which each malware was deployed, used and their compilation process.

3.2.1 Botnets used

Different tests were carried out for the Qbot, Mirai and Torii Botnet within the laboratory.

The botnets Qbot and Mirai were deployed and controlled by the researcher.

To control the botnets, a C&C server was created for each one and the source code of the bots was modified to connect to the C&C.

The creation of these servers and the modification of the source code were performed in order to allow the execution in the reduce scope. Additionally, the Torii malware was deployed within the environment. The source code of the Torii botnet is not currently available on the internet. For this reason, the obtained samples were extracted from an archive belonging to hybrid analysis.

The Torii botnet propagation methods were contained using different methods. The containment of Torii was possible by the use of a sinkhole created in the DNS server and firewall rules.

The connection attempts from Torii to the C&C were redirected and denied. This was performed in order to eliminate the risk of an improper use of the devices. The principal configurations made within the sinkhole and the firewall are presented in the section “Botnet contention method”.

Additionally, the configurations of the Qbot and Mirai botnets which provide a scope reduction and containment within the laboratory are presented below.

QBOT Configuration

The files for the Yakuza variation of QBOT are 3: the C&C source code, the Binary source code to infect devices and the script for cross compilation of different architectures.

The basic configuration of the malware and its C&C required the installation of some specific dependencies. These dependencies are gcc, httpd, Vsftpd, python and Perl.

Once the dependencies are installed, it is possible to proceed with the configuration and build of C&C. The build process of the C&C was performed using the command gcc over the file server.c with the flag pthread as it is showed below.

```
gcc server.c -o server -pthread
```

Once this command runs, it generates the executable binary of the server. The binary needs to be executed to start the process of the C&C server that will be listening for connections. However, it is necessary to specify a user and password before the execution of the binary. The user and password can be specified by a normal text file with the name “login.txt” with the format <user> <password> .This file needs to be located in the same folder as the C&C server binary in order to be accessed. Once the file is in place, it is possible to execute the C&C by adding the bot port (bot port is used for the bots connection), the number of threads and the access port for management as it is showed below.

```
./server <bot> <threads> <access>
```

```
./server 23 100 666
```

This previous command will create a new process listening in the port 23 for bots and in the port 666 for management connections. To connect to the management interface it is needed to execute the telnet command to the localhost in the port 666 as below.

```
telnet localhost 666
```

To configure the infectious malware, it is necessary to modify the connection string of the C&C before its compilation. In the file yakuza.c, it is needed to change the IP address already presented in the file for the one belonging to the C&C.

This change of IP addresses is needed to be performed in order to avoid real malicious activity. Once the information has been modified, it is possible to proceed with the compilation. For the compilation process, it is provided a script within Yakuza that will compile the source code in different architectures allowing the creation of a single distribution point.

The script can be found under the name yakuza.py.

./yakuza.py

Once the script runs, it will compile the source code and place the binaries in the folders "www/html/" and "/var/ftp/". Additionally, the python script will create a new shell script that will work as a principal dropper called "bin.sh".

When this script is executed, it will attempt a connection to the C&C to retrieve the corresponding binary based on the victim's architecture. After the binary is successfully retrieved, the binary is executed and the bot daemon is started. For the propagation of the binaries, Qbot needs to be executed with another program to attempt the brute force or dictionary attack to the devices.

In this experiment, a python script was developed to perform a brute force attack to the telnet services in the victim devices. Additionally the script works as main infection method by retrieving the file bins.sh using the command wget.

QBOT spread method

For the propagation mechanism, QBOT generates a special URL that must be accessed from the command line in the compromised device. This URL access can be achieved by using different script languages such as Python or Perl.

In these scripts it is possible to automate the access to the vulnerable devices and execute the malicious URL given by QBOT. This URL works as the dropper of the malware that analyzes the architecture of the device and verifies the privileges of the user.

Once the dropper is executed, it creates a new request to the C&C server which specifies the architecture and obtains the executable binary necessary to infect the device.

By the execution of the binary, the malware infection mechanism is completed and the same process repeats to the next victim to attempt the infection.

Mirai Configuration

The Mirai source code is composed by 3 different directories (dlr, loader and Mirai) containing the files needed for its compilation.

Inside the "Mirai" directory 3 different sub-directories can be found. These subdirectories have the source code of the C&C, the malware and the source code of some tools needed to compile the malware.

To configure the Mirai botnet, it is necessary to obfuscate the data of the C&C server.

Mirai uses a basic authentication mechanism based on a MySQL database. This database is needed to be configured with a specific name and tables in order to allow the authentication. For this it is necessary to name the database Mirai and create 3 tables: history, users and whitelist. The full structure of the database can be accessed from [85].

Additionally, it is necessary to have the cross compilation files that can be obtained from [85] and install the following dependencies : gcc, golang, electric-fence, apache2 and vsftpd.

Once the necessary dependencies have been installed and the database has been configured, it is necessary to modify the main.c file in the *mirai/cnc/* directory. In this file, it is necessary to change the IP address, username and password of the database for the own ones.

To start with the configuration, it is needed to perform a compilation attempt in order to get the tools that will allow to continue with the process.

This compilation attempt can be performed by the use of the following command inside the Mirai directory:

```
./build.sh <debug:release> <telnet:ssh>
```

```
./build.sh debug telnet
```

This command will create the C&C binary and the tools needed to compile the bot.

Once the compilation attempt is finalized, it is possible to verify the proper operation of C&C.

To verify the operation of the C&C, it is firstly needed to insert a new user into the database to connect to the C&C server. This insertion of the new user can be performed by selecting the mirai database and using the following command inside the SQL engine:

INSERT INTO users VALUES (NULL, 'username', 'password', 0, 0, 0, 0, -1, 1, 30, '');

Once the user has been created, it is necessary to copy the file prompt.txt located in the folder '*/mirai"* to the location "*/mirai/debug"* and "*/mirai/release"*.

Once the file is located in the folder, the configuration of the C&C is completed and it is possible to test it.

To test the C&C it is necessary to execute the binary *./cnc* that was generated inside the corresponding folder */debug* or */release*.

This will establish the connection with the database and open the port 23 listening for the connection. Once the binary is run, a connection to the C&C can be established by the use of the command "*telnet localhost"* and the username and password inserted in the database.

Once the C&C is configured and executed, it is necessary to perform the following steps to configure and compile the bot that will be used to infect the IoT devices.

First, it is necessary to use the tool that was compiled during the first compilation attempt. This tool is localized under the directory */debug* and it is called *enc*. The tool "enc" performs obfuscations to different types of data. This tool will be used to obfuscate the domain name of the C&C server. Once the domain name is obfuscated, it is needed to be added into the malware configuration. In this research a local DNS server was created to obtain a domain name.

To obtain this obfuscation it is necessary to run the following command:

../debug/enc string {your.domain.com}.

In this command the type of data to obfuscate together with the data itself are needed to be specified.

Once this tool is executed, the output needs to be added to the file "/mirai/bot/table.c" in the line with the comment "server name".

Once these steps are completed it is necessary to re-compile the files. This will create the binaries of the different architectures. These files will be the ones infecting the different devices within the network.

Mirai spread method

Once the malware infects a penetrated device, it automatically enables its internal scanner function. The scanner function selects different IP addresses for scanning in a semi-random way. This function will search for easy to guess accounts in the telnet and ssh services. This semi-random selection is based on a list of IP addresses to be ignored (black list). This list is composed by IP addresses related to the Department of Defense, corporations (HP, GE, etc.) and home environments. This black listing approach provides a manageable scope in Mirai by eliminating the risk factors involved in the spreading of the malware from the malicious point of view. These risks factors are associated with the obtention of vulnerable devices from unwanted IP addresses that can lead to the detection of the infection.

After the vulnerable devices are located, the malware sends the information to the C&C server. This information is processed by a module called Scanlistening inside the C&C server.

This module is in charge of the recollection of all information of the vulnerable devices that are found by the internal scanner. Once the information is recollected and proceeded, the module sends the information to a sub-module of Mirai called "loader". This module creates an auto removable dropper and sends the infectious binary to the device using the information previously collected by the scanner module.

Torii spread method

In addition, the Torii infectious binary was propagated within the laboratory in a manual way. The propagation was performed in a manual way for the lack of current information, source code and the propagation method used by Torii. A sample of the malware was obtained from an archive belonging to "hybrid analysis". The obtained sample was focused on the specific architecture ARM of the IoT devices. By the use of this sample, it was possible to spread the malware through the IoT devices. The malware was deployed by the use of command line with root privileges inside of the different devices intended to be infected.

3.2.2 Botnets contention methods

One of the principal risks within this experiment was the abuse of the devices within the laboratory.

The current information of the functions and protocols used by Torii are still unknow. This unknow information are a risk that is needed to be addressed. This risk created the possibility of an unauthorized used of the devices in order to execute attacks or recollect information.

In this laboratory 2 principal risks that are needed to be addressed were found.

These risks are the following ones:

The possible existence of hidden code in the internal propagation method of Mirai to connect to its real C&C.

Unknow spread methods and functions of Torii (ports, protocols, Domains)

For these risks associated within the experiment, it was needed the creation of a method to ensure the non-malicious use of the devices. In the following section the contention methods created in order to achieve the non-malicious use of the devices are described.

The botnets contention method were created considering the propagation methods of Mirai and Torii. Although, the Mirai propagation methods are well known and it is possible to address this risk by the proper configuration of the malware, extra measures were taken in order to ensure the proper contention. These extra measures were applied in order to contain the Mirai and torii botnets.

These measures were the creation of a sinkhole and firewall rules to deny the possible connections to the reals C&Cs. The relevant information and configuration of each of these measures is described below:

DNS Sinkhole

The DNS sinkhole is used to redirect the DNS records of the real botnet C&C to localhost , non-existing or unused IP address within the network. A valid IP is not returned for the host resolution and the infected host never connects to the botnet C&C server.

The general overview of how the DSN sinkhole was working can be seen in the Figure 4.

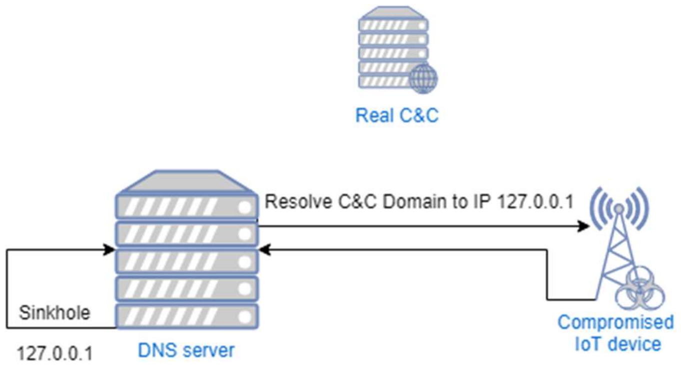

Figure 4 DNS Sinkhole

Additionally, the configuration made in order to redirect all attempts to resolve the DNS records are presented in the Table 4.

Botnet	Real C&C Domain	Redirected to
Mirai	xf0.pw	127.0.0.1
Mirai	santasbigcandycane.cx	
Mirai	disabled.racing	
Mirai	swinginwithme.ru	
Mirai	imscaredaf.xyz	
Mirai	queryhost.xyz	
Mirai	icmp.online	
Torii	haletteompson.com	
Torii	tillywirtz.com	
Torii	andrewabendroth.com	
Torii	akotae.com	
Torii	haletteompson.com	
Torii	dushe.cc	
Torii	psoriasiafreelife.win	

Table 4 *DNS Sinkhole Domains*

Firewall rules

The firewall rules were applied in the main router in order to only allow the outgoing traffic from the devices in the range 192.168.10.128/25 for the ports used in HTTP, DNS (APT needed ports) . Additionally, an allow rule was applied to outgoing traffic from all the other devices except the physical IoT devices.

The configured rules are described below in the Table 5.

Source	Port	Destination	Port	Protocol
192.168.10.128/25	80,53	Any	Allow	TDP/UDP
192.168.10.60	80,53	Any	Allow	TDP/UDP
192.168.10.61	80,53	Any	Allow	TDP/UDP
192.168.10.53	80,53	Any	Allow	TDP/UDP
192.168.10.101	80,53	Any	Allow	TDP/UDP
192.168.10.107	80,53	Any	Allow	TDP/UDP
192.168.10.10	Any	Any	Allow	TDP/UDP
192.168.10.20	Any	Any	Allow	TDP/UDP
Any	Any	Any	Deny	TDP/UDP

Table 5 Firewall Rules

3.2.3 Date Range

The different malwares propagated within the laboratory were deployed at different times. These times were used to obtain detailed information of each Botnet and eliminate overlaps of information. Besides, the Mirai malware has a specific function that verifies whether another malware is running and eliminates it in order to have control. The times used for each of the malware and the original point of infection are shown in the Table 6.

Malware	Date – start	Date – end	Origin endpoint of infection
QBOT	March-09-2019:18:56:07	March-11-2019:	Ip 192.168.10.128
MIRAI	March-12-2019:19:13:34	March-14-2019:	Ip 192.168.10.155
TORII	March-16-2019:20:06:22	March-17-2019:	Ip 192.168.10.135

Table 6 Malware propagation times and point of infection

3.2.4 Infected devices

In order to obtain an expected behavior of the malign and benign activity, limited number of the devices were infected. QBOT infected 40 different devices chosen in a pseudo randomized way. This was performed by the python script specifying a scope of devices to be scanned and infected.

On the other side, the Mirai botnet infects 25 devices. This was performed by the change of the configuration of the internal scanner mechanism. This configuration was changed in order to avoid the malware spreading to external IP ranges.

The malware of the Torii botnet was deployed in a manual way in 12 devices. This method was selected in order to minimize the possible unwanted propagation of Torii. All devices under the scope of Torii were under the DNS sinkhole scope as commented in the section "Botnets contention methods" of this document.

The scope for infection was reduced within the laboratory. This reduction was performed in the network by changing the sub-mask /24 to /25. The beginning of the new network segment was the IP 192.168.0.128. This network segment provided a range of more than 100 IP addresses that were susceptible to be infected.

The different infected IP addresses and controlling botnets are showed in the Table 7.

IPs	Botnet
192.168.10.128,178,192,191,152,156, 155,179,182,130,149,198,201-207,136-146,162-173.	Qbot
192.168.10.128,189,145,156,132,177, 190,134,165,182,149,145,199,136,133, 201,206,144,179,158,139,166,158,181,172	Mirai
192.168.10.135,137,142,147,152,157,162, 167,172,177,182,187	Torii

Table 7 Infected IPs

3.2.5 Malicious behavior

The configurations showed in the sections “Botnets used”, “Botnet contention methods”, “Data range” and “Infected devices” were used in order to generate the expected malign behavior. In these configurations was took into account the propagation method, times, number of devices and internal configuration of the Botnets.

Within the laboratory the 42% of the network traffic captured was belonging to malicious traffic. The 62% of the devices were infected through the spread of the different malwares during 6 of the 11 days of the laboratory execution.

A more in deep inspection of the network traffic and malware spread rating can be observed in the section “Traffic profile” of this document

4 Results & Discussions

The network packets of the internal connections were collected and sent to the monitor network using the port mirroring router.

Once the information was received in the monitor network, it was captured and dumped by the use of the tool Tcpdump in the capture server. This information was sent for analysis to the SIEM server in which Splunk was installed.

The total number of packets captured and sent within the experiment are shown in the table 8.

Number of packets	Type of traffic	Number of devices
4,661,067	Malware Qbot	40
2,457,683	Malware Mirai	25
1,783,973	Malware Torii	12
13,437,636	Benign Traffic	85

Table 8 Number of Packets

The SIEM Splunk has embedded methods for the identification and labeling of the information. Additionally, Splunk has an optimized indexing method which allows the obtention and labeling of information from different sources. The indexing method is used to extract information by the detection of the data source used. Once the information was obtained and indexed by Splunk, the Splunk query language was used for the data categorization and labeling.

4.1 Traffic profile

Within the laboratory 23,340,359 network packets were capture belonging to malicious and non-malicious traffic.

In the Figure 5 it is possible to observe the non-malicious traffic generated across the time. The principal information that can be extracted from this Figure is the low but constant traffic generated across the time in where 32% of the packages belongs to system updates , 53% to the communication of IoT devices (MQTT) protocols and the missing 15% to other network data (TLS errors, pings, etc.). In the traffic generated it is possible to observe a bigger concentration of traffic during the days of the week. However, it is showed a peak of information during the

Saturday. This peak could be cause by the update of the IoT devices that was configured to don't use bandwidth during the weekdays. After this peak the network packages went down during the Sunday until Monday morning.

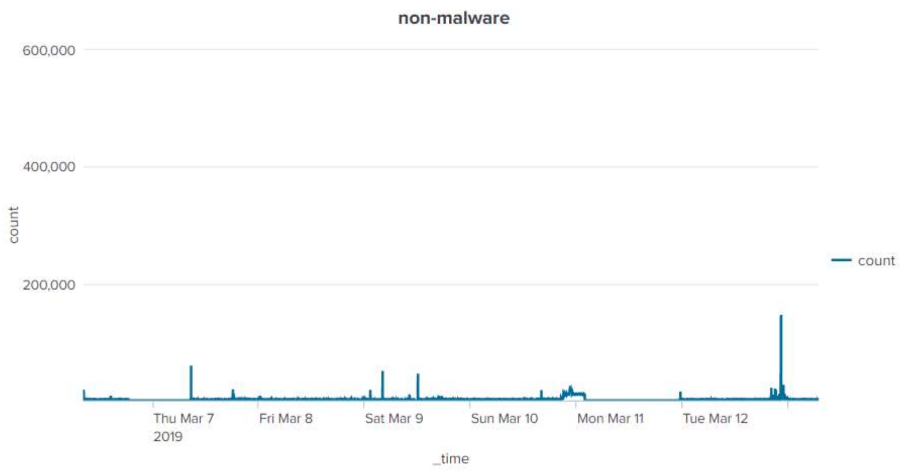

Figure 5 Non-malware traffic across time

In the Figure 6 the network packages captured across the time belonging to the malware traffic are displayed. In this Figure it is possible to observe the main accumulation of data during the spread of Qbot, on Saturday 9. The peak presented on Saturday night could be the constant communication between the C&C and the bots.

Additionally, it is possible to observe the peak presented on Tuesday 12 when the spread of mirai started to occur.

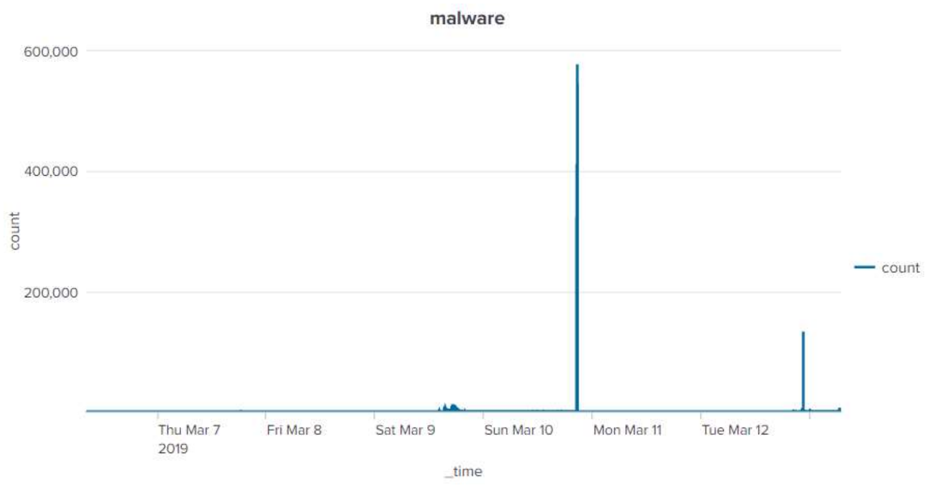

Figure 6 Malware traffic across time

The packets generated from Torii at the end of the laboratory can be seen , however as the number of devices was limited to 12, the information captured was lower than Qbot and Mirai.

Within the different network packages, the main data captured was belonging to the spread of the malware across the different devices in where 68% of the data belongs to the spread of the different malware and 32% to the communication between the C&C and the bots.

4.2 Labeling of data

The labeling of the information was performed within the SIEM Splunk. The labeling was performed by the use of the Splunk query language "SPL". By the use of this language, it was possible to label the data as benign or malign traffic using the IP addresses of the different devices in the laboratory. The benign traffic was extracted by the depreciation of the IP addresses used in the bot configuration. Additionally, the protocol used within the network packets was considered.

The next command is a sample of the Splunk query language used within Splunk to extract the benign network traffic and label it:

"index=main dst_port!=23 src_port!=23 dst_ip!="192.168.10.61" | where NOT cidrmatch("192.168.10.128/25", src_ip) AND NOT cidrmatch("192.168.10.128/25", dst_ip)"

The internal communication among the IoT devices was disregarded by the use of the previous query. This communication was disregarded in view of the fact that this type of communication does not exist in a normal benign behavior. This provided an opportunity to label the information that will be used for the creation of the datasets based on expected behavior of the network traffic.

The malign traffic was labeled by the use of similar methods with some extra considerations applied. In the malign traffic labeling process only incoming connections with the C&C and communication among the IoT devices were considered. This communication as it was commented before is expected to be established only in the malign traffic. For example, this kind of communication will be generated when an IoT device uses the Mirai scan module to infect new vulnerable devices.

The next query is a sample of the extract method used within Splunk to get the malign network traffic data.

"index=main | where cidrmatch("192.168.10.128/25", src_ip) AND dst_ip="192.168.10.61" OR cidrmatch("192.168.10.128/25", dst_ip) AND src_ip="192.168.10.61" OR cidrmatch("192.168.10.128/25", src_ip) AND cidrmatch("192.168.10.128/25", dst_ip)"

4.3 Feature Extraction with Splunk

Once the labeled files were obtained, the feature extraction was performed. This extraction resulted in the creation of the required dataset for the future behavior-based training of an IDS.

The methods followed to extract the features are described below.

After the labeling of the data, the dataset described in the research [43] was taken as a base structure of the dataset. Once the base structure was prepared, the features were extracted from the files by the use of the Splunk query language (SPL). The features generated were stored in a csv file in order to allow the future use of this data in IDS trainings [8].

In the study [43], a set of 23 different features were extracted from the network packages.

For this research the same 23 features were extracted in order to create the datasets for training of IDS devices.

The feature extraction method used within the laboratory is described below, as well as the Splunk queries used[8].

The feature extraction was based on 5 time frames: 100ms, 500ms, 1.5s, 10s and 1m [43].

The features obtained were based on the size of bytes transmitted per network packet, number of hosts observed and the mean of the packets transmitted on a time frame.

In order to aggregate the information, it was necessary to consider the following characteristics:

- Source IP,
- Source Mac address,
- Source IP and destination IP,
- Jit between the packet sent from source IP to destination IP (the jit is the difference of time between transactions with the same IP values),
- Source IP and source port together with destination IP and destination port [43].

Additionally, Splunk provides functions allowing the obtention of this data types.

The functions used for the extraction of features are described in the Table 9.

Function	Description
bin	This function allows to specify the time windows to be used for the analysis of the information. It is necessary to specify the field in where the time window will be applied and the time: *bin _time span = 100ms.*
Eval	This function allows the evaluation and concatenation of functions in the same query: | eval *M=len(_raw),* stdev(M).
Len	This function is used to calculate the length on bytes of the specified variable, it needs to be used together with the eval function to obtain the length of the sent packets: *eval M=len(_raw).*
Mean	This function calculates the mean of a variable: mean (M)
Stdev	This function calculates the standard variation of a variable: stdev(M).
Transaction	This function calculates the time among different events by specifying the number of events: transaction src_ip dst_ip maxevents=2.
By	This statistical function allows the obtention of statistical data considering other fields: By _time src_ip.
correl	This function calculates the correlation between variables: correl(X,Y).
magn	This function calculates the magnitude between variables: correl(X,Y).
covar	This function calculates the covariance between variable: correl(X,Y).

Table 9 Splunk SPL functions

The complete queries that allowed the obtention of the features by the use of Splunk are available in the following link:

https://github.com/saranted/IoT-behavior-Dataset/

The principal information previously described can be observed in the table 20.

Value	Statics	Aggregated by	Number of features
Packet bytes (size)	Mean, standard variation	Source Ip, Source Mac, Source IP-mac, Transaction time	8
Packet Count	Number	Source Ip, Source Mac, Source IP-mac, Transaction time	4
Jitter	Mean , standard variation, Number	Transaction time	3
Packet bytes bidirectional	Magnitude , Radius, Covariance, Correlation	transaction, , "Source IP , destination IP"	8

Table 10 Features characteristics taken from [43]

The information extracted was stored in a csv file. The names of the columns in the file were created with the next format : <Statics function><aggregated by><time frame>

The statics function names were replaced in the csv file by the names provided in the Table 11

Statics function names	Name in file
Number	W
Mean	Mean
Standard Variation	Stdev
Magnitude	Magnitude
Radius	Radius
Covariance	Covariance
Correlation	Correlation

Table 11 Statics function representation in csv file

The names of the information used to aggregate the features were replaced in the csv file by the names provided in the Table 12.

Aggregated information	Name in file
packet size transfer in a unidirectional deprecating response (host to all)	H
packet size transfer in a bidirectional way between IPs (host to host)	HH
packet transfer from host to host taking ports as enrichment of data (host: port to host: port)	HpHp
difference in time between transaction with the same IP values (host to host)	HH_jit

Table 12 Aggregated information representation in csv file

4.4 Feature analysis

Once the features were extracted and stored in the dataset, Fisher score algorithm was used as a discrimination mechanism for the analysis of the features. This algorithm is used to solve maximum likelihood equations named after Ronald Fisher. Fisher score algorithm can be applied in a large range of different machine learning techniques in order to extract the most important features to be used in machine learning. This algorithm was used to extract a suboptimal subset of features of the dataset created in this research [3], [27], [51].

The Fisher algorithm is based on the next equation where μij and ρij are the mean and the variance of the selected feature in each of their behavior respectively. Nj is the number of instances in the respective class and μi is the mean of all features [3].

$$S_i = \frac{\sum_{k=1}^{K} n_j(\mu_{ij} - \mu_i)^2}{\sum_{k=1}^{K} n_j \rho_{ij}^2},$$

Equation 1 Fisher Score taken from [3]

Once the Fisher algorithm was applied to the generated dataset, it gave a specific score to each different feature in where bigger is better. The generated score can be observed in the Figure 6.

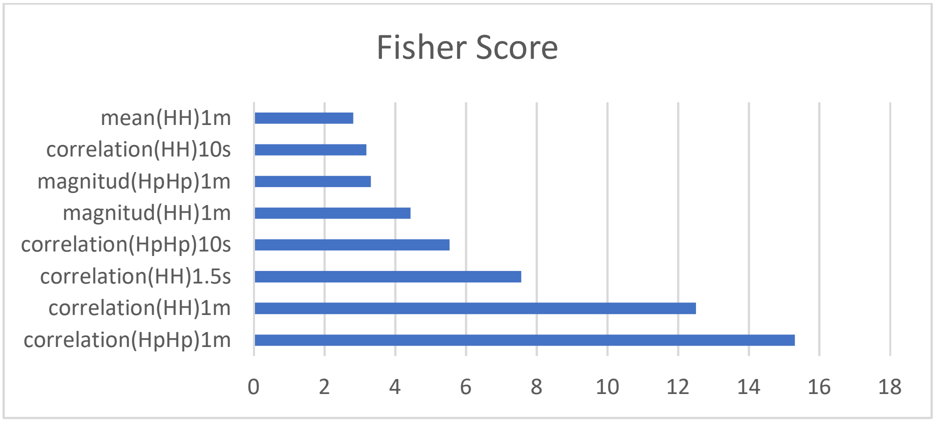

Figure 7 Higher Fisher Scores

In the Figure 6 , there can be observed that the biggest score was achieved by the features extracted from the magnitude and correlation of the tcp-streams aggregated with the “host:port to host:port” and the “host to host” information.

Feature	Fisher Score
correlation(HpHp)1m	15.31008
correlation(HH)1m	12.5085
correlation(HH)1.5s	7.570503
correlation(HpHp)10s	5.535085
magnitud(HH)1m	4.435691

Table 13 Higher Fisher Scores

The Fisher score algorithm provides the possibility to obtain the most important features from all 115 features previously generated. The 5 most important features showed in the Table 11 were considered for the analysis described below.

The highest Fisher score obtained were for the features “correlation(HpHp)1m” and the “correlation(HH)1m”. These features generated a strong correlation between the different steams and the malicious activity. This characteristic can be used for discriminating benign activity from malign activity.

Within the analysis of the features extracted in this research, 10 different histograms were generated in relation between the feature and the malware activity.

The different histograms generated showed the values extracted by the non-malware and malware activity in where the axis X shows the values extracted by the feature and the Axis Y shows the count of non-malware and malware records respectively.

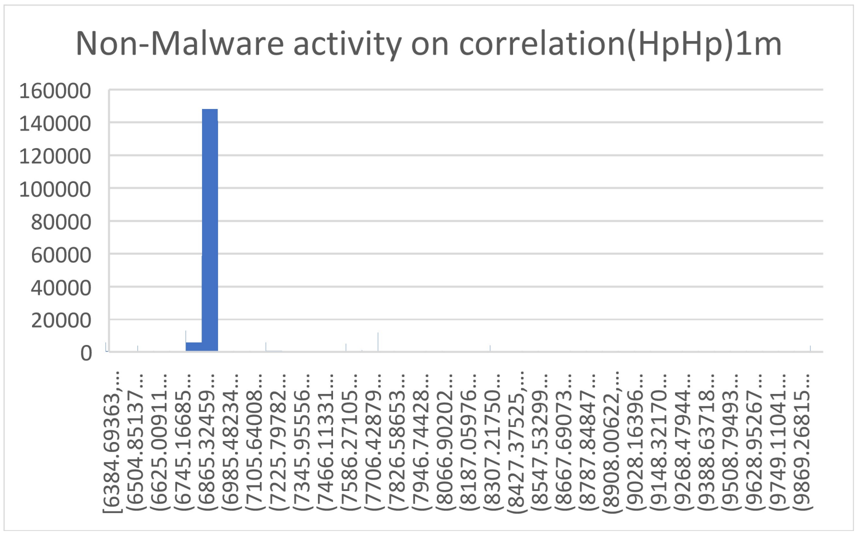

Figure 8 Non-Malware activity on correlation(HpHp)1m

In the Figure 7 the values of the non-malware activity extracted by the feature "correlation(HpHp)1m" are presented. In this Figure it is possible to observe the high number of events between the values "6805" and "7015" representing more than 80% of the events in total. The missing percentage of events are distributed across the different events generated.

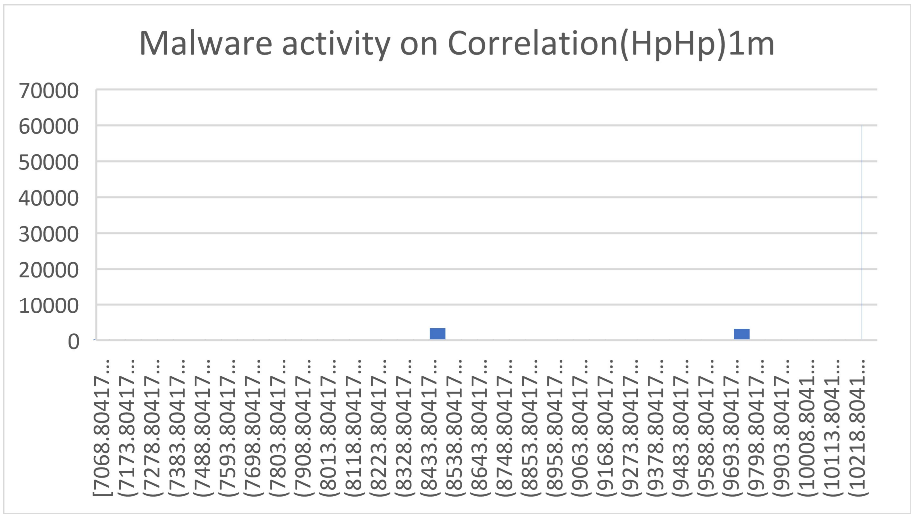

Figure 9 Malware activity correlation(HpHp)1m

In the Figure 8 the values of the malware activity extracted by the feature "correlation(HpHp)1m" are presented. In the Figure 8 is observable the concentrated number of events presented over the values "10218" representing the 89% of the events. The remaining 11% of events can be found among the values "9693" and "8538".

The Figures 7 and 8 shows the different values generated for each specific behavior. Using this information, it is possible to observed that the malicious behavior generates values over "10000" in almost all the cases. These values can be used in order to detect malicious behavior by the computation of the feature over the tcp streams. However, a small part of malicious behavior will not be able to be detected with the use of this values. The events generating values under "10000" could not be considered as malicious leading to a false negative.

Additionally, was analyzed the feature "correlation(HH)1m" presenting the same behavior of the previous analyzed feature "correlation(HpHp)1m".

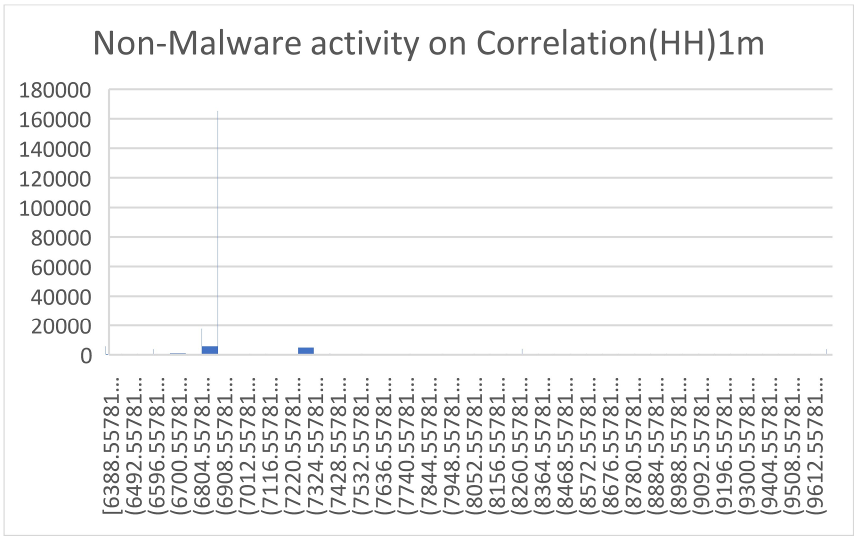

Figure 10 Non-Malware activity on correlation(HH)1m

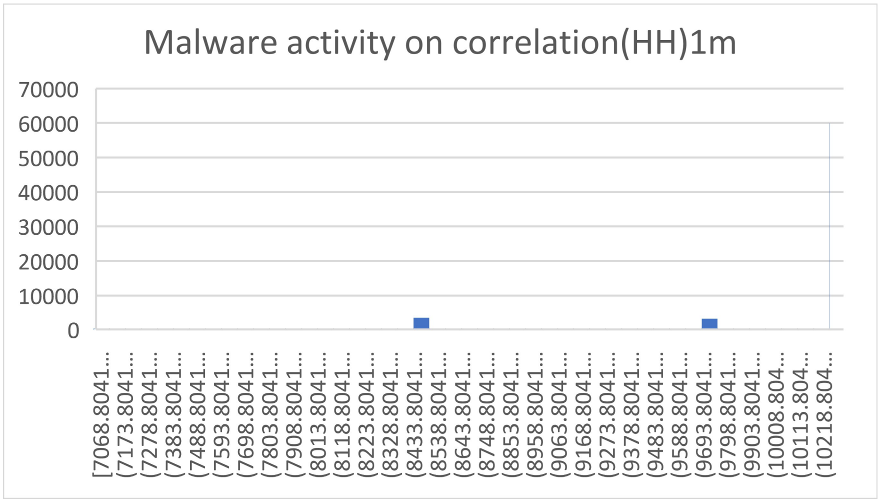

Figure 11 Malware activity on correlation(HH)1m

In the Figure 11 the values of the non-malware activity extracted by the feature "correlation(HH)1.5s" are presented.

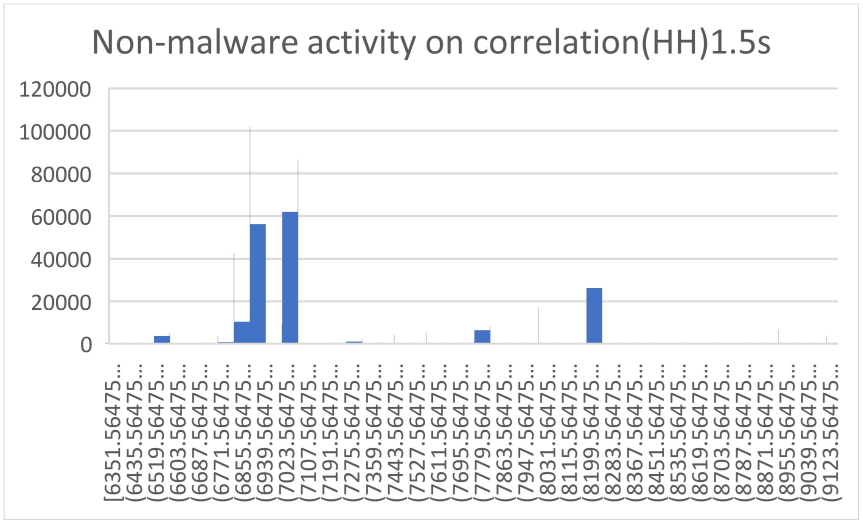

Figure 12 Non-malware activity on correlation(HH)1.5s

In the Figure 11 it is possible to observe a distributed number of events across the values extracted by feature "correlation(HH)1.5s". It is possible to be observed a centralized number of events among the values 6771 and 7107.

The Figure 12 represents the histogram generated with the values of the malware activity extracted by the feature "correlation(HH)1.5s" are presented. In this Figure the 84.5% of events generated by the malware activity are presented over the value "10268" that provides a similar behavior of the Figure 8.

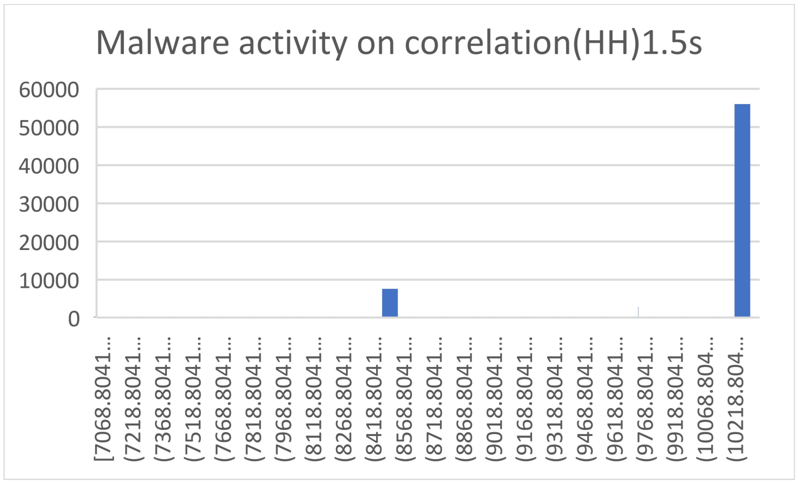

Figure 13 Malware activity on correlation(HH)1.5s

In comparation with the previous analysis of the Figures 7 and 8, the Figures 11 and 12 presents a bigger number of events distributed across the values. These values could create a bigger number of false negatives as the values of non-malware and malware activity are more dispersed.

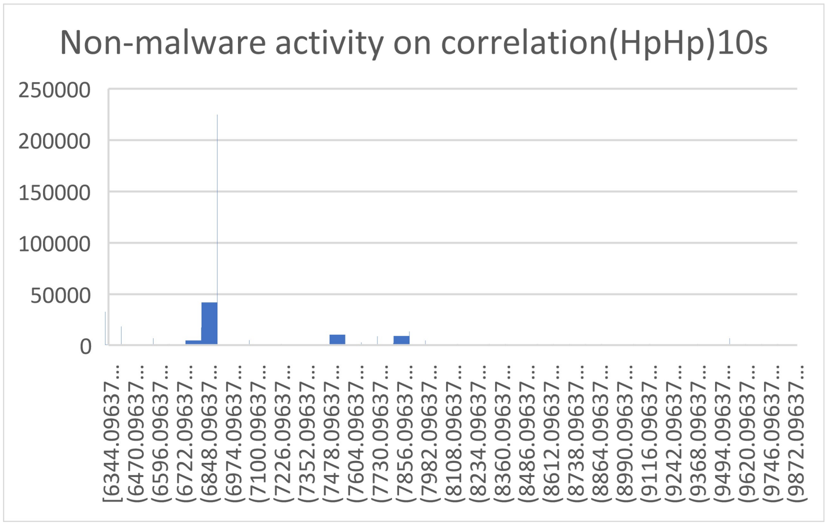

Figure 14 Non-malware activity on correlation(HpHp)10s

In the Figure 13 and 14 the values of the non-malware activity and malware activity extracted by the feature "correlation(HpHp)10s" are presented respectively. In these Figures it is possible to be observed the same behavior as the previous Figures presented.

The main observable characteristic across the different histograms presented (Figure 7-12),it is the close relation between the time frame used within the feature and the dispersion of values. This characteristic shows a less dispersed values of events over a bigger time frame.

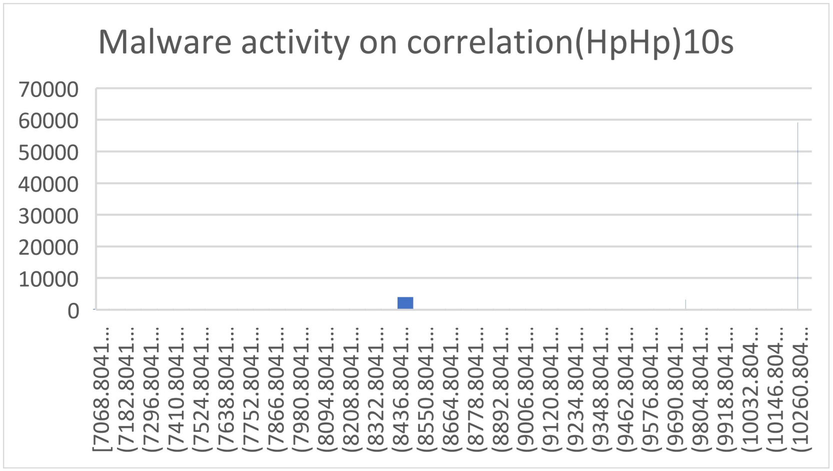

Figure 15 Malware activity on correlation(HpHp)10s

Additionally, 2 histograms were generated for the feature with worst Fisher score within the research "W(M)100ms". In the Figure 15 and 16 the values of the non-malware activity and malware activity extracted by the feature "W(M)100ms" are presented respectively. In the figure 15 it is showed 3 accumulation points of non-malware activity among the values 1 to 2.83.

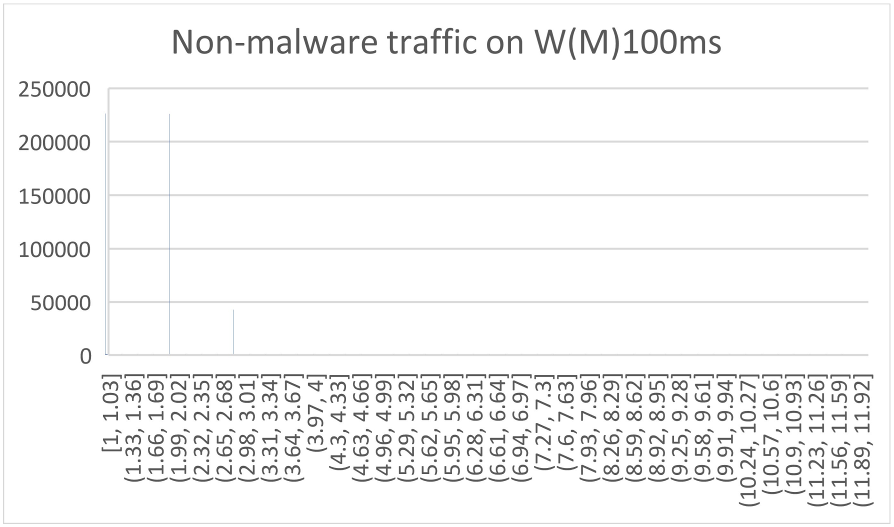

Figure 16 Non-malware activity on magnitude(HH)1m

The Figure 16 for the other side shows the main malware activity belonging to the values between 1 and 1.088. This accumulation represents more than 85% of the malware activity and unfortunately it is presented in the same range of non-malicious values. This reason could be the one why this feature got the worst Fisher score within this research.

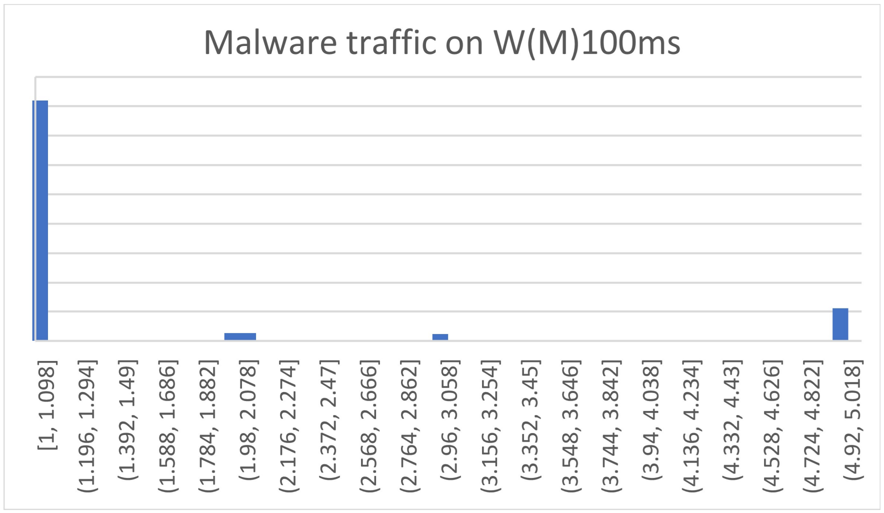

Figure 17 Malware activity on magnitude(HH)1m

Additionally, In the research [84], the Fisher score algorithm was applied to the dataset "detection_of_IoT_botnet_attacks_N_BaIoT" created by the University of California Irvine.

The 5 most important features extracted by the Fisher score can be observed in the Table 11

Feature	Fisher Score
stdev(M)10s	1.7035
stdev(H)10s	1.7035
stdev(H)1.5s	1.7051
stdev(M)1.5s	1.7051
stdev(M)1m	1.7073

Table 14 Higher Fisher Score N_BaIoT dataset taken from [84]

The first 2 features obtained by the fisher score were considered for the analysis described below.

Within the analysis of the features obtained by the fisher score applied on the 'N_BaIoT" dataset, 4 different histograms were generated in relation between the feature and the malware activity.

The different histograms generated shows the values extracted by the non-malware and malware activity in where the axis X shows the values extracted by the feature and the Axis Y shows the count of values extracted.

In the Figure 17 the 76% of non-malware activity is concentrated under the value "42". The missing percentage of events are distributed among the different events generated within the values "42" to "4470".

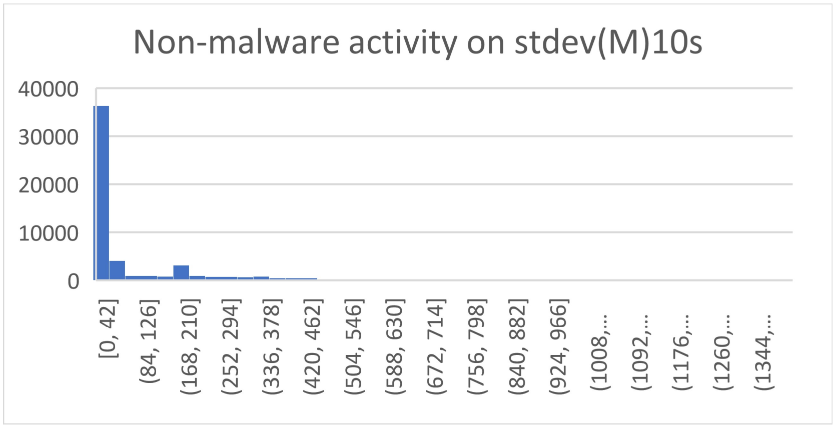

Figure 18 Non-malware activity on stdev(M)10s

In the Figure 18 the 73% of malware activity is concentrated under the value “76”. The missing percentage of events are distributed among the different events generated within the values “76” to “2856”.

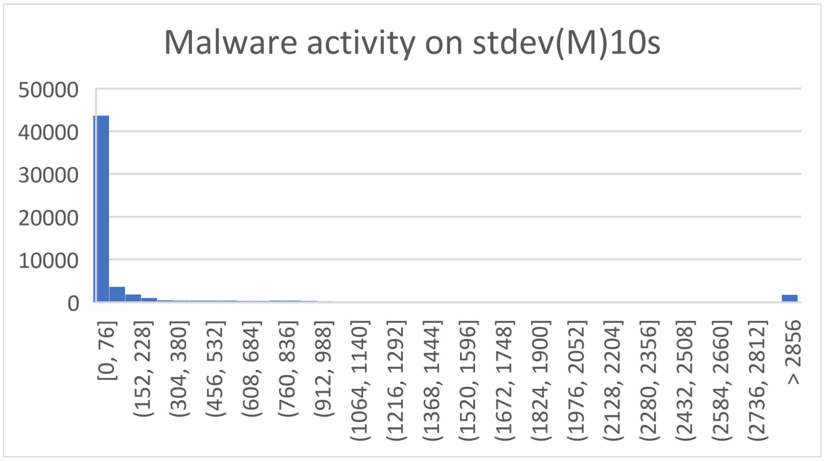

Figure 19 malware activity on stdev(M)10s

The biggest problem that could be presented considering the previous analysis of the Figures 17 and 18, are the values generated between the malware and non-malware activity. These values can overlap between them or being close to each other. This problem could lead to false positives and false negatives in the detection of non-malware and malware activity.

In the Figure 19 and 20 the values of the non-malware activity and malware activity extracted by the feature “H_L1_variance” are presented respectively. The Figure 19 presents 73% of non-malware activity with similar values than the Figure 17, with the biggest number of concentrated events under the value “42”.

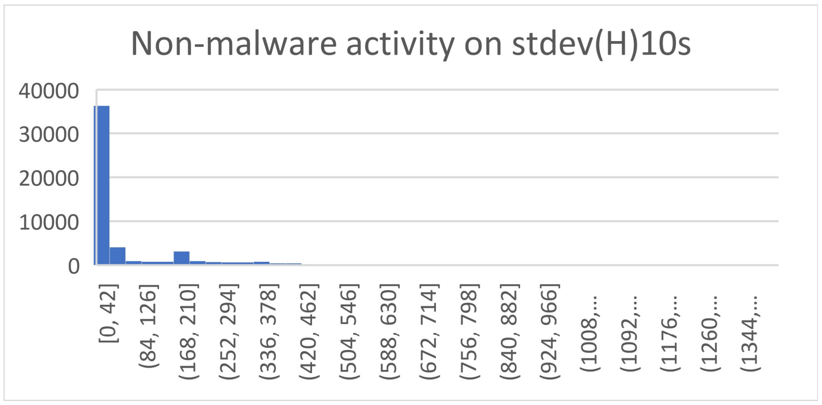

Figure 20 Non-malware activity on stdev(H)10s

For the other side, the Figure 20 shows an almost same percentage of events concentrated under the value “72” than the Figure 18 previously described.

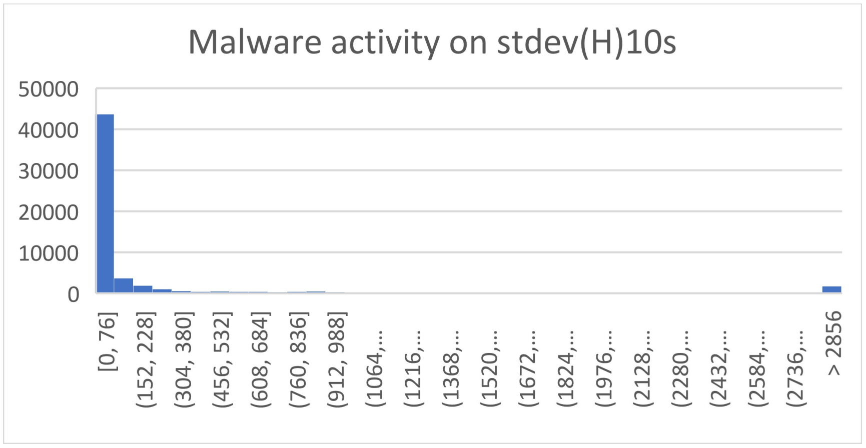

Figure 21 Malware activity on stdev(H)10s

Within the Figures 19 and 20 the problems previously observed in the Figures 17 and 18 are equally presented. These problems as before could lead to false positives and false negatives in the detection of non-malware and malware activity.

Within this research and “N_BaIoT” was observed different features with the highest fisher score.

It is possible to observed that the host-based features within this research were discriminated, unlike in the research “N_BaIoT”. This discrimination could be based on the different approaches of the dataset generated. While "N_BaIoT" dataset was focused on the capture of network packets related to the attack of botnets. This research was more focused on the capture of network packets related to the spread of the malwares.

This could be the principal reason why the Fisher score algorithm showed different results within the features.

Conclusion

The main goal of this thesis was the creation of a medium behavioral datasets for Mirai, Qbot and Torii Botnets.

The answer to the research questions are presented below:

1. How to create a medium infrastructure of IoT devices for testing.

Using the containerization technology, it was possible to create a medium infrastructure with a limited number of resources. In this infrastructure with the use of a Python script, it was possible to emulate the operation of the IoT devices. Once this was achieved, the script was embedded in a Docker container. Using the previous Docker container, an iterative method was implemented in order to create the desired infrastructure of IoT devices with their own characteristics.

2. How to obtain information about the communication and infection methods of Botnets, in a medium infrastructure.

Using the analysis capabilities of the SIEM Splunk it was possible to extract the information of the different infection and communication methods used by the botnets.

3. How to create an expected behavior of IoT devices.

The IoT device management systems provide many methods for automation. With this kind of automation, an IoT devices can be configured to perform different actions depending of a trigger action. With this, it can be performed a simple script to perform actions that are expected for a normal behavior.

4. How to address the Torii botnet in a behavioral dataset.

Thanks to the obtention of an infection file, it was possible to spread the malware within the laboratory. With this approach it was possible to capture the behavioral information of the IoT devices that were under the Torii attack. Once this information was captured, it was generated a behavioral dataset of this new threat.

The main result of this thesis is the creation of 3 different datasets based on a medium infrastructure. Each dataset is contains the features extracted within the network traffic for the different botnets Qbot, Mirai and Torii. Additionally, each dataset is enriched with the features extracted from benign behavior.

With these datasets the possibility of detection has been extended taking into account also the new infrastructures and the new threat Torii.

New improvements can be performed to the current datasets available today. These improvements taking into consideration new threats and new infrastructures can be done. Additionally, methods like the ones used for the emulation of the IoT devices can be improved or modified to create new tools for detection (for example honeypots).

References

[1] 6 IoT Communication Protocols for Web Connected Devices. 30 May 2018, www.getkisi.com/blog/internet-of-things-communication-protocols. Accessed 10 May 2019.

[2] 8 Best SIEM Tools: A Guide to Security Information and Event Management. 7 May 2019, www.comparitech.com/net-admin/siem-tools/. Accessed 10 May 2019.

[3] Aggarwal, Charu C. Data Classification: Algorithms and Applications. CRC, 2015.

[4] Alrashdi, Ibrahim, et al. "AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning." 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), 2019, doi:10.1109/ccwc.2019.8666450.

[5] Arm Ltd. IoT Technology – Arm. www.arm.com/solutions/iot/iot-technology. Accessed 10 May 2019.

[6] Auliva, Ridlo Savvidina, et al. "IIoT Testbed: A DDS-Based Emulation Tool for Industrial IoT Applications." 2018 International Conference on System Science and Engineering (ICSSE), 2018, doi:10.1109/icsse.2018.8520091.

[7] Benkhelifa, Elhadj, et al. "A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems." IEEE Communications Surveys & Tutorials, vol. 20, no. 4, 2018, pp. 3496–3509., doi:10.1109/comst.2018.2844742.

[8] Big Data Analytics Solutions: Machine Data Can Reveal Customer Behavior, Security Threats. www.splunk.com/en_us/solutions/solution-areas/big-data.html. Accessed 10 May 2019.

[9] Bolzoni, Damiano. "Revisiting Anomaly-Based Network Intrusion Detection Systems." doi:10.3990/1.9789036528535.

[10] Chen, Yi-Jui, and Hung-Yu Chien. "IoT-Based Green House System with Splunk Data Analysis." 2017 IEEE 8th International Conference on Awareness Science and Technology (ICAST), 2017, doi:10.1109/icawst.2017.8256458.

[11] CoAP. coap.technology/. Accessed 10 May 2019.

[12] Colon, Alex, et al. The Best Smart Home Devices for 2019. 24 Jan. 2019, www.pcmag.com/article/303814/the-best-smart-home-devices-for-2019. Accessed 10 May 2019.

[13] Configuring a Processor for Low-Power IoT Applications. www.synopsys.com/designware-ip/technical-bulletin/processor-configurations.html. Accessed 10 May 2019.

[14] Cyber Risk in an Internet of Things World | Deloitte US. 12 July 2018, www2.deloitte.com/us/en/pages/technology-media-and-telecommunications/articles/cyber-risk-in-an-internet-of-things-world-emerging-trends.html. Accessed 10 May 2019.

[15] Desai, Anuja S., and D. P. Gaikwad. "Real Time Hybrid Intrusion Detection System Using Signature Matching Algorithm and Fuzzy-GA." 2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT), 2016, doi:10.1109/icaecct.2016.7942601.

[16] Ding, Yu-Xin, et al. “Research and Implementation on Snort-Based Hybrid Intrusion Detection System.” 2009 International Conference on Machine Learning and Cybernetics, 2009, doi:10.1109/icmlc.2009.5212282.

[17] Dupont, Corentin, et al. “Edge Computing in IoT Context: Horizontal and Vertical Linux Container Migration.” 2017 Global Internet of Things Summit (GIoTS), 2017, doi:10.1109/giots.2017.8016218.

[18] Empowering the Smart Home. www.openhab.org/. Accessed 10 May 2019.

[19] Emulate a City's Worth of IoT Devices!www.valid8.com/solutions/iot-load-tester. Accessed 10 May 2019.

[20] Gopal, Tatikayala Sai, et al. “Mitigating Mirai Malware Spreading in IoT Environment.” 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, doi:10.1109/icacci.2018.8554643.

[21] Gopal, Tatikayala Sai, et al. “Mitigating Mirai Malware Spreading in IoT Environment.” 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, doi:10.1109/icacci.2018.8554643.

[22] Harsha, M S, et al. “Analysis of Vulnerabilities in MQTT Security Using Shodan API and Implementation of Its Countermeasures via Authentication and ACLs.” 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, doi:10.1109/icacci.2018.8554472.

[23] Himanshu. Application Layer Protocols for IOT : IOT Part 11. 23 Oct. 2018, www.engineersgarage.com/Articles/IoT-Application-Layer-Protocols. Accessed 10 May 2019.

[24] Himanshu. Network Layer Protocols: IOT Part 8. 23 Oct. 2018, www.engineersgarage.com/Articles/IPv4-IPv6-Network-Layer--Protocols. Accessed 10 May 2019.

[25] Home Assistant. Hass.io. www.home-assistant.io/hassio/. Accessed 10 May 2019.

[26] Home-Assistant. Home-Assistant/Home-Assistant. 10 May 2019, github.com/home-assistant/home-assistant. Accessed 10 May 2019.

[27] Huang, Hong, et al. “Complete Local Fisher Discriminant Analysis with Laplacian Score Ranking for Face Recognition.” Neurocomputing, vol. 89, 2012, pp. 64–77., doi:10.1016/j.neucom.2012.02.020.

[28] Ifttt. IFTTT. ifttt.com/openhab. Accessed 10 May 2019.

[29] Infrastructure for Container Projects.linuxcontainers.org/. Accessed 10 May 2019.

[30] Internet of Things (IoT). www.trendmicro.com/vinfo/us/security/definition/internet-of-things. Accessed 10 May 2019.

[31] Intrusion Detection Techniques, Methods & Best Practices: Detecting Network Intrusion in 2019. www.alienvault.com/blogs/security-essentials/intrusion-detection-techniques-methods-best-practices. Accessed 10 May 2019.

[32] IoT and Smart Energy: Constructing Digital Buildings. 12 Feb. 2019, www.iotforall.com/iot-smart-energy-digital-buildings/. Accessed 10 May 2019.

[33] IoT Device Management: What Is It and Why Do You Need It?20 Feb. 2019, www.iotforall.com/what-is-iot-device-management/. Accessed 10 May 2019.

[34] IoT Standards & Protocols Guide | 2019 Comparisons on Network, Wireless Comms, Security, Industrial. www.postscapes.com/internet-of-things-protocols/. Accessed 10 May 2019.

[35] Jokinen, Kristiina, et al. "Dialogues with IoT Companions: Enabling Human Interaction with Intelligent Service Items." 2017 International Conference on Companion Technology (ICCT), 2017, doi:10.1109/companion.2017.8287082.

[36] Kim, Young-Ho, et al. "Processing of Multi-Pattern Signature in Intrusion Detection System with Content Processor." 2007 6th International Conference on Information, Communications & Signal Processing, 2007, doi:10.1109/icics.2007.4449753.

[37] Kuwabara, Yoshiki, et al. "Hardware Emulation of IoT Devices and Verification of Application Behavior." 2017 23rd Asia-Pacific Conference on Communications (APCC), 2017, doi:10.23919/apcc.2017.8304040.

[38] Kuwabara, Yoshiki, et al. "Hardware Emulation of IoT Devices and Verification of Application Behavior." 2017 23rd Asia-Pacific Conference on Communications (APCC), 2017, doi:10.23919/apcc.2017.8304040.

[39] Learn the Application Layer Protocols Used In IoT. 8 Sept. 2017, blog.eduonix.com/internet-of-things/learn-application-layer-protocols-used-iot/. Accessed 10 May 2019.

[40] Lee, Kyungwoon, et al. "Analysis on Network Performance of Container Virtualization on IoT Devices." 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017, doi:10.1109/ictc.2017.8190937.

[41] Looga, Vilen, et al. "MAMMOTH: A Massive-Scale Emulation Platform for Internet of Things." 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, 2012, doi:10.1109/ccis.2012.6664581.

[42] Marzano, Artur, et al. "The Evolution of Bashlite and Mirai IoT Botnets." 2018 IEEE Symposium on Computers and Communications (ISCC), 2018, doi:10.1109/iscc.2018.8538636.

[43] Meidan, Yair, et al. "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders." IEEE Pervasive Computing, vol. 17, no. 3, 2018, pp. 12–22., doi:10.1109/mprv.2018.03367731.

[44] Mendki, Pankaj. "Docker Container Based Analytics at IoT Edge Video Analytics Usecase." 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), 2018, doi:10.1109/iot-siu.2018.8519852.

[45] MQTT. mqtt.org/. Accessed 10 May 2019.

[46] Nguyen, Huy-Trung, et al. "IoT Botnet Detection Approach Based on PSI Graph and DGCNN Classifier." 2018 IEEE International Conference on Information Communication and Signal Processing (ICICSP), 2018, doi:10.1109/icicsp.2018.8549713.

[47] Ni, David C., and Pete Chen. “Integration of Wired and Wireless Lighting Control Systems for Green Energy Managment.” 2017 International Conference on Mechanical, System and Control Engineering (ICMSC), 2017, doi:10.1109/icmsc.2017.7959472.

[48] Nijhof, Franck. Awesome Home Assistant. www.awesome-ha.com/. Accessed 10 May 2019.

[49] Osborne, Charlie. Meet Torii, a New IoT Botnet Far More Sophisticated than Mirai Variants. 19 Jan. 2019, www.zdnet.com/article/meet-torii-a-new-iot-botnet-far-more-sophisticated-than-mirai/. Accessed 10 May 2019.

[50] Pamukov, Marin E., and Vladimir K. Poulkov. “Multiple Negative Selection Algorithm: Improving Detection Error Rates in IoT Intrusion Detection Systems.” 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017, doi:10.1109/idaacs.2017.8095140.

[51] Parzen, Emanuel. “Change Analysis and Fisher-Score Change Processes.” 1992, doi:10.21236/ada254707.

[52] Poyner, I.k., and R.s. Sherratt. “Privacy and Security of Consumer IoT Devices for the Pervasive Monitoring of Vulnerable People.” Living in the Internet of Things: Cybersecurity of the IoT - 2018, 2018, doi:10.1049/cp.2018.0043.

[53] Providing IoT Host-Based Datasets for Intrusion Detection ...www.researchgate.net/publication/328531254_Providing_IoT_host-based_datasets_for_intrusion_detection_research. Accessed 10 May 2019.

[54] Radanliev, P., et al. “Economic Impact of IoT Cyber Risk - Analysing Past and Present to Predict the Future Developments in IoT Risk Analysis and IoT Cyber Insurance.” Living in the Internet of Things: Cybersecurity of the IoT - 2018, 2018, doi:10.1049/cp.2018.0003.

[55] Radware. A Quick History of IoT Botnets. 2 Mar. 2018, blog.radware.com/uncategorized/2018/03/history-of-iot-botnets/. Accessed 10 May 2019.

[56] Sekharan, S. Sandeep, and Kamalanathan Kandasamy. “Profiling SIEM Tools and Correlation Engines for Security Analytics.” 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017, doi:10.1109/wispnet.2017.8299855.

[57] Shukla, Anurag, and Sarsij Tripathi. “Security Challenges and Issues of Internet of Things: Possible Solutions.” SSRN Electronic Journal, 2018, doi:10.2139/ssrn.3166735.

[58] Sinanovic, Hamdija, and Sasa Mrdovic. “Analysis of Mirai Malicious Software.” 2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2017, doi:https://ieeexplore.ieee.org/document/8115504. Accessed 4 May 2019.

[59] Sonmez, Ferda Ozdemir, and Banu Gunel. “Evaluation of Security Information and Event Management Systems for Custom Security Visualization Generation.” 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), 2018, doi:10.1109/ibigdelft.2018.8625291.

[60] Tanaka, Hiroaki, and Shingo Yamaguchi. “On Modeling and Simulation of the Behavior of IoT Malwares Mirai and Hajime.” 2017 IEEE International Symposium on Consumer Electronics (ISCE), 2017, doi:10.1109/isce.2017.8355547.

[61] Tanganelli, G., et al. “CoAPthon: Easy Development of CoAP-Based IoT Applications with Python.” 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015, doi:10.1109/wf-iot.2015.7389028.

[62] Team, Breadware. Top 7 Internet of Things IoT Communication Protocols. 11 Dec. 2018, breadware.com/blog/iot-communication-protocols/. Accessed 10 May 2019.

[63] The CTU-13 Dataset. A Labeled Dataset with Botnet, Normal and Background Traffic.mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-background-traffic.html. Accessed 10 May 2019.

[64] The Current State of IoT Cybersecurity. 9 Oct. 2018, www.iotforall.com/current-state-iot-cybersecurity/. Accessed 10 May 2019.

[65] The New IoT Era Requires Better Standardization. 14 Dec. 2018, www.iotforall.com/iot-era-requires-better-standardization/. Accessed 10 May 2019.

[66] Top 10 IoT Vulnerabilities of 2018. www.itworldcanada.com/article/top-10-iot-vulnerabilities-of-2018/413433. Accessed 10 May 2019.

[67] Top 15 Standard IoT Protocols That You Must Know About. 7 Nov. 2018, www.ubuntupit.com/top-15-standard-iot-protocols-that-you-must-know-about/.

[68] Urunov, Khamdamboy, et al. “High-Level Architectural Design of Management System for the Internet of Underwater Things.” 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), 2018, doi:10.1109/icufn.2018.8437002.

[69] Voica, Alex. Alex Voica. 21 June 2016, www.mips.com/blog/a-guide-to-iot-processors/. Accessed 10 May 2019.

[70] What Is A Botnet?us.norton.com/internetsecurity-malware-what-is-a-botnet.html. Accessed 10 May 2019.

[71] What Is a Container?www.docker.com/resources/what-container. Accessed 10 May 2019.

[72] What Is Containerization in DevOps?20 Nov. 2018, www.linuxnix.com/what-is-containerization-in-devops/. Accessed 10 May 2019.

[73] What Is Internet of Things (IoT)? - Definition from WhatIs.com. internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT. Accessed 10 May 2019.

[74] What Is Intrusion Detection System (IDS)? - Definition from WhatIs.com. searchsecurity.techtarget.com/definition/intrusion-detection-system. Accessed 10 May 2019.

[75] What Is IoT Botnet (Internet of Things Botnet)? - Definition from WhatIs.com. internetofthingsagenda.techtarget.com/definition/IoT-botnet-Internet-of-Things-botnet. Accessed 10 May 2019.

[76] What Is Security Information and Event Management (SIEM)? - Definition from WhatIs.com. searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM. Accessed 10 May 2019.

[77] Why Splunk?www.splunk.com/view/SP-CAAAFV2. Accessed 10 May 2019.

[78] Yang, Ming, et al. “Semi_Fisher Score: A Semi-Supervised Method for Feature Selection.” 2010 International Conference on Machine Learning and Cybernetics, 2010, doi:10.1109/icmlc.2010.5581007.

[79] Yokotani, Tetsuya, and Yuya Sasaki. “Transfer Protocols of Tiny Data Blocks in IoT and Their Performance Evaluation.” 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), 2016, doi:10.1109/wf-iot.2016.7845442.

[80] Yokotani, Tetsuya, and Yuya Sasaki. “Comparison with HTTP and MQTT on Required Network Resources for IoT.” 2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016, doi:10.1109/iccerec.2016.7814989.

[81] You, Ilsun, et al. “On IoT Misbehavior Detection in Cyber Physical Systems.” 2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC), 2018, doi:10.1109/prdc.2018.00033.

[82] ¿Qué Es Docker?www.redhat.com/es/topics/containers/what-is-docker. Accessed 10 May 2019.

[83] Gifford, Will, et al. “Residential Lighting End-Use Consumption Study: Estimation Framework and Initial Estimates.” 2012, doi:10.2172/1162372.

[84] Bahsi, Hayretdin, et al. “Dimensionality Reduction for Machine Learning Based IoT Botnet Detection.” 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV), 2018, doi:10.1109/icarcv.2018.8581205.

[85] Jorge Albert Medina Galindo. IoT Behavior Dataset. github.com/saranted/IoT-behavior-Dataset/. Accessed 13 May 2019.