University of Zurich[UZH]

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Modeling the Behavior of Malware Affecting the Integrity of Raspberry Pis

Ülkü Karagöz
Zurich, Switzerland
Student ID: 17-729-039

Supervisor: Dr. Alberto Huertas Celdrán, Eder John Scheid
Date of Submission: August 2, 2021

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Zusammenfassung

Die zunehmende Anzahl an IoT-Geräten führt zu neuen und disruptiven Anwendungsszenarien und Paradigmen. Diese Situation zeigt sich in Crowdsensing-Plattformen wie ElectroSense, wo IoT-Sensoren Spektraldaten überwachen, die an eine Backend-Plattform gesendet werden, wo sie verarbeitet und mit den Nutzern geteilt werden.

Mit der zunehmenden IoT-Nutzung steigt auch die Zahl der Malware, die jedes Jahr IoT-Geräte aufgrund von Schwachstellen befällt, die auf mangelndes Wissen der Nutzer und begrenzte Ressourcen zurückzuführen sind. Diese Schwachstellen wurden 2016 vom Botnetz Mirai ausgenutzt, das zeigte, wie leistungsfähig kleine Geräte wie Kameras, Fernseher und andere mit dem Internet verbundene Geräte in Kombination sein können. Obwohl Malware wie Viren, Würmer oder Spyware schon seit langem Computer befallen, ist die Entwicklung hybrider Malware wie Botnetze besorgniserregend [18]. Vor allem IoT-Geräte, für die es kaum Sicherheitsmassnahmen gibt, sind einem grossen Risiko ausgesetzt.

Um IoT-Geräte vor derartigen Angriffen zu schützen, werden Lösungen zur Malware-Erkennung benötigt. Herrkömmliche Malware-Erkennungstechniken können zwar bekannte Malware erkennen, sind aber nicht in der Lage, unbekannte Angriffe zu erkennen. Daher sind neue Mechanismen auf der Grundlage von maschinellem und tiefem Lernen (ML bzw. DL) von entscheidender Bedeutung, insbesondere für die Erkennung von Zero-Day-Angriffen. Diese Algorithmen versuchen, Anomalien zu erkennen und benötigen Datensätze, die aus dem internen Verhalten von IoT-Geräten bestehen, um trainiert und verbessert zu werden. Es gibt jedoch immer noch einen Mangel an Datensätzen, die das Verhalten von Botnetzen aus der Geräteperspektive modellieren.

Um die bisherigen Einschränkungen zu verbessern, ist das Ziel dieser Arbeit, Datensätze zu erstellen, die das interne Verhalten eines ElectroSense-Spektrumsensors bestehend aus einem Raspberry Pi enthalten, der mit den beiden bekannten Botnetzen Mirai und Bashlite infiziert ist. Nach erfolgreicher Ausführung der Malware werden Distributed Denial of Service - kurz DDoS - Angriffe gestartet. Während dieser Angriffe wird das Gerät mit einem Monitoring-Skript überwacht. In dieser Arbeit wird versucht, eine Grundlage für Malware-Erkennungsalgorithmen zu schaffen, die für die Erkennung von Malware, insbesondere Botnetzen, auf IoT-Geräten trainiert und eingesetzt werden können.

Am Ende der Arbeit werden die erstellten Datensätze ausgewertet und wichtige Ergebnisse mitgeteilt.

Abstract

The increasing number of IoT devices is bringing to reality new and disruptive application scenarios and paradigms. This situation can be seen in crowdsensing platforms, like ElectroSense, where IoT sensors monitor spectrum data, which is sent to a backend platform where it is processed and shared with users.

The increment in IoT usage also raises the number of malware affecting IoT devices every year due to vulnerabilities coming from poor user knowledge and limited resource capabilities. These vulnerabilties were exploited by the botnet Mirai in 2016 which demonstrated how powerful small devices like cameras, TVs, and other internet-connected devices can be when combined [5]. Although malware such as viruses, worms, or spyware are affecting computers for a very long time, the evolution of hybrid malware like botnets is concerning [18]. Especially IoT devices with almost no security measurements are at big risk.

To prevent IoT devices from such big attacks, malware detection solutions are needed. However, traditional malware detection techniques can detect well-known malware but are not capable of detecting unknown attacks. Therefore, it is crucial to have new mechanisms based on Machine and Deep Learning (ML and DL, respectively), especially to detect zero-day attacks. These algorithms try to detect anomalies and need datasets consisting of the internal behavior of IoT devices to be trained and improved. However, there is still a lack of datasets modeling the behavior of botnets from the device perspective.

To improve the previous limitations, the goal of this thesis is to create datasets that contain the internal behavior of an ElectroSense spectrum sensor running on a Raspberry Pi that is infected with the two well-known botnets Mirai and Bashlite. After executing the malware successfully, Distributed Denial of Service (DDoS) attacks are launched. During these attacks, the device is monitored using a monitoring script. This thesis tries to give a basis for malware detection algorithms to be trained and used for detecting malware, especially botnets, on IoT devices. To conclude the thesis the created datasets are evaluated and important results are shared.

Acknowledgments

A big thanks goes to my supervisors Dr. Alberto Huertas Celdrán and Eder John Scheid and to Prof. Dr. Burkhard Stiller for their support during the thesis. I want to thank Dr. Alberto Huertas especially for being available for questions and motivation throughout the thesis.

I also thank my friends proof-reading this thesis.

Contents

Chapter 1

Introduction

The Internet of Things (IoT) is defined as an ecosystem consisting of connected devices that exchange data over the internet. The first IoT device was created in 1990 and was a toaster that could be turned on and off using the internet. Today, these devices are being used in various fields, such as medicine, the military, or transport. Since 2008 there are more IoT devices than people connected to them which is also considered the "birth" of IoT [13]. While about 8.7 billion internet-connected devices were in use by the end of 2020, by 2025 that number is expected to reach 16.44 billion [46]. One of the reasons for this strong growth is the advancement in Cloud Computing and Big Data. Collected data can be stored more cost-effectively on cloud servers such as Amazon Web Services, Microsoft Azure and Alibaba Cloud [3][34][2] and with the help of Big Data technologies, the organization of such data is simplified. This also enables the understanding of large volumes of data collected by such devices [14].

The growth of IoT devices is also having a major impact on crowdsensing platforms. Crowdsensing is an approach where users of IoT devices send data to these platforms which are collected, processed, analyzed, and mapped by different algorithms (e.g. Google Maps mapping traffic information). An illustration of how such a platform works is shown in Figure 1.1. The advantage of such platforms is the ability to avoid often very expensive sensors for the extraction of information about the weather, traffic, and air pollution. Instead, everyday devices like smartphones can come in place of these sensors.

Since IoT devices have a big range of use, they can be used as low-cost sensors by crowdsensing platforms and serve for data collection. While Google Maps gathers data using mostly smartphones, the crowdsensing platform ElectroSense makes use of Raspberry Pis which are small single-board computers. The goal of this platform is to use low-cost sensors and to have a flexible backend at the same time that can scan in real-time and measure large amounts of spectrum data with low latency [42]. The ElectroSense architecture consists of three components: The sensors, the backend, and the controller which is illustrated in Figure 1.2. The analyzed data is displayed to the users via the ElectroSense website [11].

Since Raspberry Pis and IoT devices, in general, are cheap, the resources on them are limited. This makes it impossible to install well-developed security measurements. Additionally, users often are not aware of threats that come with default credentials or the lack

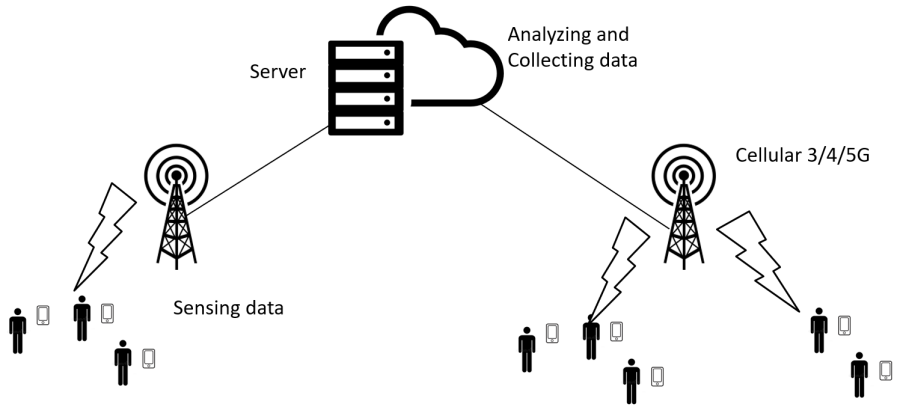


Figure 1.1: Crowdsensing

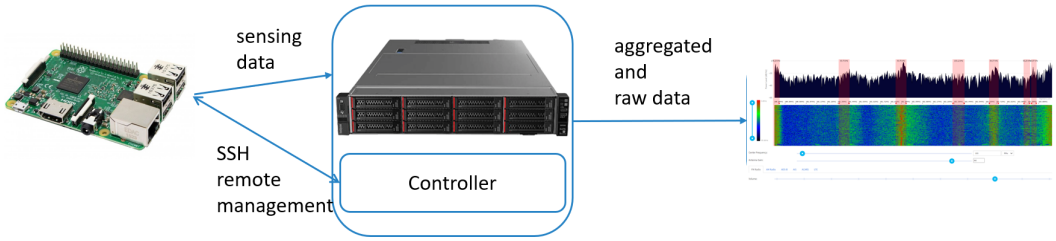


Figure 1.2: ElectroSense Architecture

of updates on devices. This leads to the devices having a lot of weaknesses that can be exploited by attackers. The growth of IoT devices also increases the number of malware attacks on them. The list of malicious software is long and there are various types of malware such as viruses, worms, botnets, spyware, and many more [18]. While some of them are only there to annoy the users, others can cause great damage. There are several reasons why IoT devices can be easy victims of malware. While users often do not change the default login credentials, they are also not being updated regularly. One of the most exploited vulnerabilities is weak and guessable passwords of which the Botnets Mirai [5] and Bashlite [30] profit. In 2014, Mirai showed how big of an impact such weaknesses can have on the economy and it is, therefore, crucial to make the users aware of such problems [5].

The detection of such malware is not always easy since malicious software is being programmed more and more to stay hidden. Therefore, malware detection algorithms based on ML and DL come into use. In contrary to traditional malware detection methods they are even able to spot hidden malware. These algorithms can be trained to detect anomalies and get better the more datasets are used while training them. These datasets consist of a device's internal behavior to improve the detection of anomalies. However, there is still a lack of datasets modeling the behavior of devices infected with botnets from their own perspective and the impact they have on crowdsensing platforms.

To fill this gap, in this thesis, an ElectoSense sensor running on a Raspberry Pi is monitored to create datsets with its normal behavior and its behavior after being infected with two botnets: Mirai and Bashlite. The behavioral fingerprinting is made after creating a

script to monitor aspects like usage of file systems, CPU, memory, and network of the device. The monitored data can be used to identify botnets at an early stage in the future and minimize the damage coming from malicious programs.

The structure of this thesis is as follows: Chapter 2 describes the background knowledge about malware. The botnets Mirai and Bashlite are explained in detail in Section 2.1. Chapter 3 provides insight into previous work around the topic of datasets and malware detection. Chapter 4, explains the setup of the devices being used and the implementation of both botnets. In section 4.3 and 4.4, the selection of the monitoring events and the creation of the dataset is presented. In Chapter 5 the collected data is analyzed and at the end, in Chapter 6, the whole work is summarized.

Chapter 2

Background

This chapter gives an understanding of the definition of malware and a detailed explanation of what botnets are. At the end of this chapter, the two well-known botnets Mirai and Bashlite are described and Distributed Denial of Service (DDoS) attacks are explained.

2.1 Malware

Malware is malicious software implemented to harm devices. There are many malware types that all show different spreading and malicious behaviors. While some (e.g. Adware) only annoy users, others (e.g. Worms, Botnets) manipulate files on the device or try to attack servers through (Distributed)-DoS attacks [18]. Figure 2.1 lists the malware types and shows their relation to certain behaviors [38]. This bachelor thesis focuses on botnets and their behaviors on a Raspberry Pi sensor.

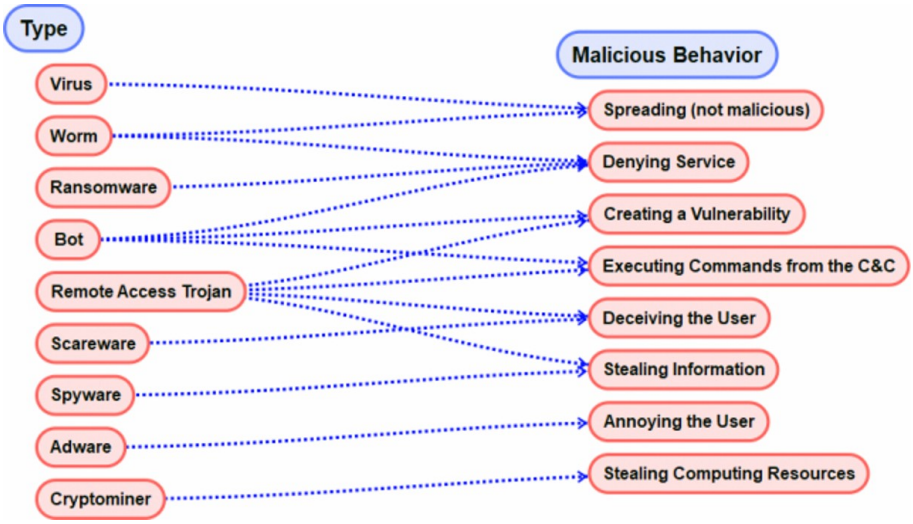


Figure 2.1: Malware type and behavior relation

2.1.1 Botnets

One type of malware that is affecting more and more IoT devices like surveillance cameras, routers, and Raspberry Pis is botnets. The increase in IoT usage leads also to a higher number of malware attacks on such devices.

Botnets are multiple connected IoT devices running bots - sometimes also referred to as zombies - controlled by an attacker called botmaster. The botmaster controls the bots over a Command and Control (C&C) server and uses them mainly to launch DDoS attacks against servers with the goal of overwhelming them with fake requests. However, they can also be used to steal data on the device, consume resources, e.g. to mine cryptocurrency, and send spam. Botmasters even offered botnets with up to 400'000 bot devices for rent which allowed customers to launch powerful DDoS attacks against servers [28].

The C&C is the most notable feature on a botnet. After being infected, the devices connect to a C&C server. That way, the botmaster is able to run commands on the bots. These commands can consist of launching DDoS attacks, installing crypto-mining software, sending spam, and much more. The bots are not only command runners, but they also have the functionality to search for new vulnerable devices. After logging into a new host successfully, they communicate this information to a report server. The device is then registered by the server and the loader runs the architecture-specific bot which is compiled for many different device architectures, including the Raspberry Pi architecture ARM. After that, the bot is ready to be controlled by the botmaster. The structure of such a botnet is illustrated in Figure 2.2 with the Mirai botnet as an example [26]. Devices are infected by exploiting vulnerabilities. When the malware is executed on the devices, they connect to a C&C server which is controlled by the botmaster. Now, either the botmaster or another user who rented the botnet can control the zombies over the C&C server. Then, the bots are used to launch DDoS attacks to specified victims and simultaneously scan the internet for more vulnerable devices to infect. After a new device is found, it is reported to the report server and the malware is executed on the device.

Users of infected devices usually do not know that they are contributing to cybercrime since botnets are programmed to stay hidden. They also can have long sleeping phases which in consequence leads to the fact that they are hard to detect.

2.1.1.1 Mirai

In 2016, several organizations were victim of a big amount of DDoS attacks. These attacks came from thousands of IoT devices that were controlled by a new botnet named after the Japanese word for "future" - the Mirai botnet. At its peak, Mirai consisted of 600'000 bots causing severe damage to Krebs on Security, OVH, and Dyn [5].

To spread, Mirai first sends TCP SYN requests to randomly chosen IPv4 addresses on Telnet or SSH ports (22, 23, 2323). There is a list in the source code containing IPv4 addresses that should not be scanned, probably to not attract attention from these organizations. If Mirai finds a potentially vulnerable device, it tries to create a Telnet connection with default credentials as listed in Table 2.1. After logging in successfully,

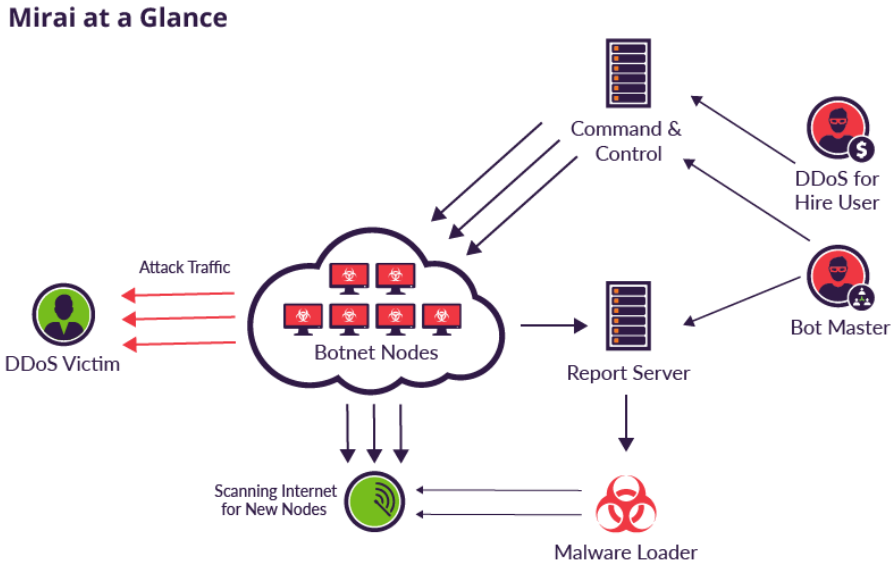


Figure 2.2: The Mirai botnet structure

the malware sends the IP address of the device to a report server. The vulnerable devices then are infected by the loader script which first logs into the host and identifies the architecture and then executes the bot module. To hide, Mirai deletes the executed script. Consequently, this results in Mirai not being present on the device after a reboot. However, if the vulnerability is still not fixed, it can be easily reinfected. Furthermore, Mirai kills other processes that could be in its way and waits for attack commands from the C&C server to execute DDoS attacks. Mirai installs the tool *zmap* on the device to Simultaneously scan further vulnerable devices and tries to connect to them again using the default credentials [5].

After the source code of Mirai was published online in October 2016, the number of variants increased strongly, thus increasing the number of zombie devices as well [35].

2.1.1.2 Bashlite

Bashlite, also known as Lizkebab, Gafgyt, or Qbot was first detected in 2014 where it made use of a vulnerability on devices running BusyBox [33]. Its source code was leaked in 2015, which lead to the creation of a big amount of Bashlite variants infecting IoT devices [49].

Bashlite represents a predecessor of Mirai. Like Mirai, it belongs to a DDoS launching malware family that infected Linux devices by using default credentials. While Mirai has a dictionary with 62 credentials, Bashlite uses only 14 username and password pairs as shown in Listing 2.1. The additional credentials help Mirai to have a wider range of infectable devices [5]. The IoT-botnet Bashlite has very similar behavior to Mirai. The scanner module first finds vulnerable devices then the malware tries to log in with the default credentials. Once successfully logged in, the malware is executed and the bot is

Table 2.1: Hard-coded default credentials used by Mirai

username	password
admin	(none)
admin	1111
admin	1111111
admin	1234
admin	12345
admin	123456
admin	54321
admin	7ujMko0admin
admin	admin
admin	admin1234
admin	meinsm
admin	pass
admin	password
admin	smcadmin
admin1	password
administrator	1234
Administrator	admin
guest	12345
guest	guest
mother	fucker
root	(none)
root	00000000
root	1111
root	1234
root	12345
root	123456
root	54321
root	666666
root	7ujMko0admin
root	7ujMko0vizxv

username	password
root	888888
root	admin
root	anko
root	default
root	dreambox
root	hi3518
root	ikwb
root	juantech
root	jvbzd
root	klv123
root	klv1234
root	pass
root	password
root	realtek
root	root
root	system
root	user
root	vizxv
root	xc3511
root	xmhdipc
root	zlxx.
root	Zte521
service	service
supervisor	supervisor
support	support
tech	tech
ubnt	ubnt
user	user
666666	666666
888888	888888

connected to the C&C server. After this process, the botmaster is able to send commands to the bot and launch DDoS attacks onto servers [30].

In 2016, it was reported that up to 1 million IoT devices were infected with Bashlite worldwide [7].

```
char *usernames[] = {"root\0", "\0", "admin\0", "user\0", "login\0", "guest
    \0"};
char *passwords[] = {"root\0", "\0", "toor\0", "admin\0", "user\0", "guest
    \0", "login\0", "changeme\0", "1234\0", "12345\0", "123456\0", "default
    \0", "pass\0", "password\0"};
```

Listing 2.1: Default username and password pairs used by Bashlite

2.2 Distributed Denial of Service

A Denial of Service (DoS) attack is a form of denying service from a user. In combination with a DDoS attack, the damage made can get very expensive. The goal of these flooding attacks is to overwhelm a server by sending large amounts of data. On the network/transport level, these attacks include UDP Flood, TCP Flood, DNS Flood, and GRE Flood which overuse the resources of the victim server by abusing certain vulnerabilities. The HTTP flooding attack is on the application level and interrupts legitimate requests by using the resources of the victim excessively [19].

There are several attack types used by the botnets Mirai and Bashlite which will be elaborated on in Section 4.4. The most important ones for this thesis are listed below:

- TCP SYN: Normally, when a client wants to connect to a server, they exchange messages. This exchange is called a three-way handshake which is illustrated in Figure 2.3. During a Transmission Control Protocol (TCP) attack, the attacker sends a high number of Synchronization (SYN) packets. Since the server thinks that these are legitimate requests, it sends SYN-ACK packets back but does not receive an awaited (Acknowledgement) ACK packet from the client. Hence, the server waits for a specific time for an answer. But before it can close the started connection, another SYN packet arrives which leads to many open connections. When no resources are available, the victim cannot connect to legitimate clients anymore and denies service [48].

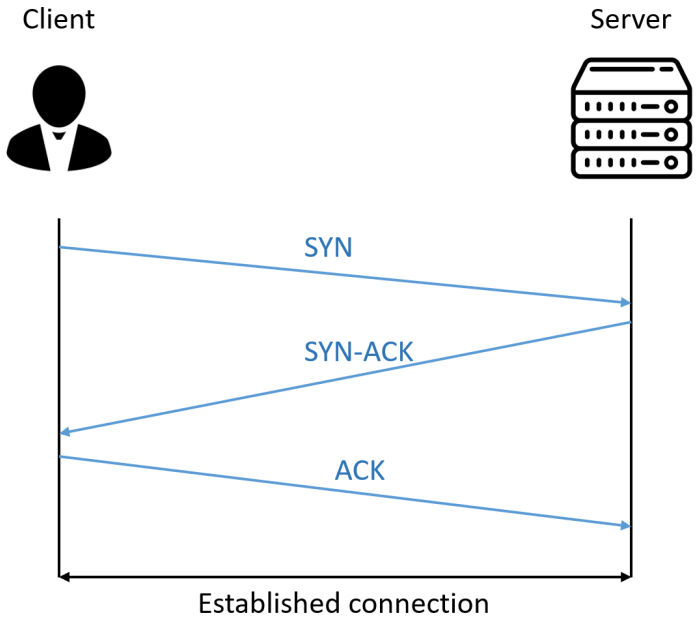


Figure 2.3: A three-way handshake process between a client and a server.

- TCP ACK: An ACK flood works in a similar way to a SYN flood. The attacker sends ACK packets in a large amount and thus slows down the server which then denies service to clients [48].

- UDP: During a User Datagram Protocol (UDP) attack, the attacker floods several ports of the target server with UDP packets. The victim checks if the port is used and then sends an Internet Control Message Protocol (ICMP) packet to tell the client that the destination is unreachable. If the number of such UDP requests is large, the server denies service to clients [48].

Chapter 3

Related Work

In this chapter, related work is introduced that deals with malware detection algorithms. Furthermore, for the training of such algorithms, some of the most relevant datasets focusing on IoT devices are introduced.

3.1 Datasets

The creation of datasets is the most crucial step in the process of detecting malware. Hence, it is important to choose the most informative data to monitor. When it comes to botnets, network traffic is the most used data source to indicate the infection of a device since bots communicate with the C&C server of the malware or send network packets in form of DDoS attacks.

The dataset InvesAndMal2019 [47] consists of both static and dynamic data from mobile devices infected with Android malware. The goal of this dataset is to detect malware on devices running Android. Every year new Android malware analyzer frameworks are proposed by researchers who need datasets to evaluate and train their analyzers. BE-HACOM [43] is another dataset that focuses on the behavior of users using a computer. With this dataset, it is possible for ML and DL-based algorithms to detect anomalies on devices running Windows and Linux OS. The dataset NGIDS-DS is generated for future intrusion detection systems for the detection of botnets in [21]. It contains normal as well as abnormal behavior of critical infrastructures. The creation of datasets can be a difficult task since IoT devices have complex services. That is why ML-based algorithms can be used to create datasets for the detection of anomalies. The authors of [40] present such an algorithm that also considers the limited resources of IoT devices. The datasets proposed in [32], [27], [41], and [20] consist of raw capture files with normal and anomalous behavior of common IoT devices. The goal is to teach ML-based anomaly detectors what normal behavior means and optimize the results with the dataset modeling the infected behavior. The dataset offered in [8] models not only the network traffic but also other system resources of the device and focuses on IoT devices affected by botnets.

In Table 3.1, selected datasets are presented. The table lists the used device types and the data sources during the monitoring phase and also states if the data is from a static or dynamic data source. Data that remains the same after its collection is named static while dynamic data refers to data that changes continuously [45]. Static data includes logs while resource usage and network traffic is part of dynamic data. As seen in the table, datasets modeling the behavior of botnets are based on dynamic data, especially focusing on network traffic.

Table 3.1: Relevant datasets with the used malware types and the generated data with their data source. The last column states if the monitored data is static or dynamic.

Dataset	**Year**	**Device Type**	**Malware**	**Data Source**	**Data**	**Static/ Dynamic**
InvesAnd Mal2019 [47]	2019	Mobile Devices	Android Malware	System Logs and Network	Processed Logs and Features	Both
BEHACOM [43]	2019	General Computers	-	Resource Usage	CPU and Memory Statistics	Dynamic
NGIDS-DS [21]	2017	Critical Infra structures	Botnet	Network and System Logs	Raw Network Packets and Audit Logs	Dynamic
DS2OS [40]	2018	IoT Devices	Botnet	Network Communi cation	Application Traces	Dynamic
N-BaIoT [32]	2018	IoT Devices	Botnet	Network	Processed Features	Dynamic
USNW IoT Benign and Attack Traces [23]	2019	IoT Devices	Botnet	Network	Raw Captures and Processed Features	Dynamic
IoT Network Intrusion Dataset [27]	2019	IoT Devices	Botnet	Network	Raw Captures	Dynamic
IoT-23 [41]	2020	IoT Devices	Botnet, Trojan	Network	Raw Captures	Dynamic
IoT-KEEPER Dataset [20]	2020	IoT Devices	Botnet	Network	Raw Captures	Dynamic

Dataset	Year	Device Type	Malware	Data Source	Data	Static/ Dynamic
IoT Host-Based Dataset [8]	2018	IoT Devices	Botnet	Network, System Resources	Raw Captures, CPU and Memory Statistics	Dynamic

However, there is still a lack of datasets consisting of the behavioral fingerprinting of IoT devices affected by botnets, especially from the device's perspective as well as considering crowdsensing platforms. To fill this gap, this thesis offers new datasets with the internal behavior of a spectrum sensor affected by botnets.

3.2 Malware Detection Algorithms

The vulnerabilities on IoT devices are big malware attractors. Therefore, malware detection algorithms are needed more than ever, and with the enormous growth of IoT, will be needed even more in the future. Botnets can remain completely undetected on innumerable IoT devices. That is why it is necessary to train malware detection algorithms, especially with IoT-based datasets. The detection of such malware is difficult but can be improved by using datasets consisting of the internal behavior of different IoT devices. One approach is to train the algorithms to classify certain types of devices by analyzing their behavior. This can be done by adding different types of IoT devices into a white list that should be recognized by the algorithm. Devices not contained in the list should be marked as unknown. This approach is used in [31] and [39]. The goal of device classification is to detect new devices in a network quickly and to recognize suspicious behavior at an early stage. Other algorithms classifying IoT devices are [15], [4], [22], [24], [9], and [6] which use datasets with the network behavior of several devices. The most accurate algorithms in the classification of devices and anomalies are BotFP [9], RF, and ANN [24] with accuracies of 92% .

It is also possible to detect the malware by analyzing their behavior. Using different ML and DL based algorithms, they can be distinguished from one another and even unknown malware can be detected. Such malware detection algorithms try to identify anomalies on a device which is done by [17], [50], [23], [36], [44], [1], and [25]. The most outstanding anomaly detection algorithms are RPNI and RANSAC [50] using neural networks .

Table 3.2 summarizes some important aspects of the most relevant malware detection algorithms. The main goal of these algorithms is to classify malware or detect anomalous behavior (approaches C and AD, respectively) based on network traffic or hardware events. In addition to the analyzed device types, the accuracies of the chosen algorithms are listed. Most importantly, it is shown which monitored behavior type and what type of attack or malware is used by the algorithm.

Table 3.2: Malware detection algorithms and their used approaches and analyzed behaviors listed by the publishing year of the work. Two different approaches are mentioned where C stands for Classification and AD for Anomaly Detection.

Work	**Year**	**IoT Device Type**	**Behavior Source**	**Attack Type/ Malware**	**App roach**	**Algorithms**	**Accu racy**
[31]	2017	Monitor, Sensor, Refrigerator, Camera, TV, Watch	Network	untargeted/ targeted attacks	C	RF	94%-99%
[15]	2017	Not Indicated	Network	unusual changes and attacks	C	ARIMA Euclidean distance	Not Indicated
[4]	2018	Cooja Simulator	Network	Traffic anomalies	C	DT, Linear Regression	Not Indicated
[39]	2018	Sensors	Network	common/ network attacks	C	Euclidean Distance	98%
[17]	2018	Smart Cameras, Smart Light	Hardware Events	Adversarial attacks	AD	EMM	Not Indicated
[50]	2019	Router, Switches, Camera, DVR, Smart Light, Fire Alarm	Network	IoT anomalies	AD	RPNI, RANSAC	99.99%
[23]	2019	Switch, Motion Sensor, Camera, Lights	Network	DDoS/ Botnets	AD	IF	94%
[22]	2019	Camera, Switch, Sensor	Network	Attack prevention	C	RF	91%
[24]	2019	HP Laptop	Network	DoS, control, scan	C	SVM, RF, DT, ANN, LR	98%
[36]	2019	Camera, Light, Sensor	Network	IoT attacks	AD	GRU	95.6%

Work	**Year**	**IoT Device Type**	**Behavior Source**	**Attack Type/ Malware**	**App roach**	**Algorithms**	**Perfor mance**
[44]	2019	Switch, Camera, Printer, Light	Network	Network attacks	AD	PCA k-means	94%
[1]	2019	Not Indicated	Network	DDoS attacks	AD	Neural Network	99.2%
[9]	2020	Not Indicated	Network	Botnet detection	C	BotFP	98%
[25]	2020	IIoT [1]	Software signatures	Software modification	AD	Hash equality checking	Not Indicated

[1]Industrial Internet of Things

Chapter 4

Creation of the Datasets

This chapter explains the analyzed malware and their execution on the device. It summarizes the adaptations that had to be made to the source codes of Mirai and Bashlite. At the end of the chapter, an understanding of the implementation of the Monitoring-Script and the creation of the datasets is given.

4.1 Setup

In this section, the devices used during the thesis and their network connection are described.

4.1.1 Devices

The goal of this thesis is to create datasets to be able to detect malware attacks on IoT devices serving as crowdsensing sensors. Raspberry Pis are being used as spectrum sensors by ElectroSense. Therefore, the device used during this thesis runs the ElectroSense software on it. A Software Defined Radio (SDR) is connected over a USB port to the Raspberry Pi which is necessary for the collection of spectrum data such as radio protocols. The device also communicates with the backend of the ElectroSense platform and sends data for processing. The processed data is available on the website [11]. Therefore, the Raspberry Pi indicates a certain behavior receiving data from the SDR and sending this information to the backend. To model the infected behavior, the device becomes a bot controlled by a C&C server and launches attacks against hosts. The created datasets are a collection of different internal parameters of the Raspberry Pi. As a malware-protection-method Secure Shell (SSH) is disabled on the Raspberry Pi per default and has to be enabled manually. After enabling SSH, it is important to change the default password to prevent the device from malware infection. SSH is used to access the device.

To simulate the C&C server, a Virtual Machine (VM) running on a Windows-Laptop is created. Table 4.1 lists the specifications of the selected devices along with the used static IP addresses.

Table 4.1: Device Specifications

	Raspberry Pi 3	**VM**
Operating System	Raspbian Stretch	Kali Linux
IP Address	10.8.0.2	10.8.0.3
Architecture	ARMv7	i686
CPU Core	4	1
RAM	1 GB	2 GB
CPU Clock	1.2 GHz	2.7 GHz

4.1.2 Network

Since the botnets spread over the internet, both the C&C server and the Raspberry Pi have to be connected to a network. This is done by connecting both devices to a sub-network in a private network by a single router. The sub-network is necessary to prevent the malware from spreading to other vulnerable devices uncontrollably. The network topology is illustrated in Figure 4.1.

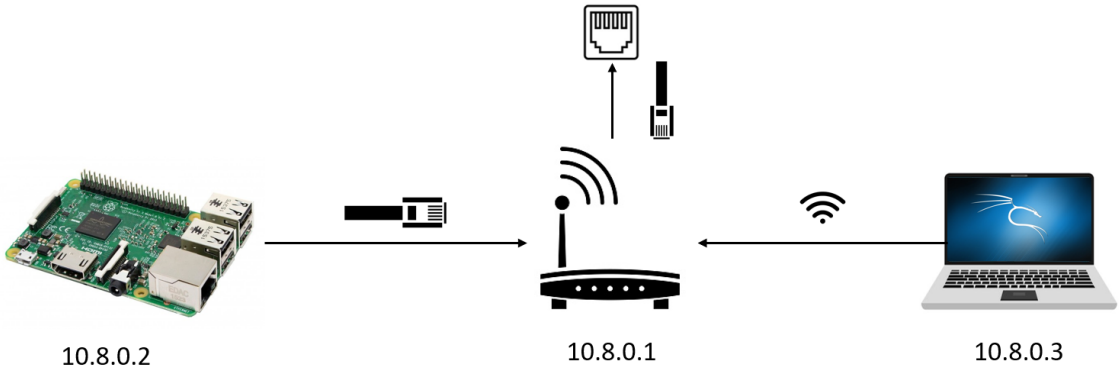


Figure 4.1: Network Setup

4.2 Malware

This section describes the changes made to the source codes of Mirai and Bashlite to execute them on the ElectroSense sensor.

4.2.1 Mirai

The Mirai source code used for this thesis is made available by Jerry Gamblin on the platform GitHub [16]. To be able to get the source code to work, a few changes have to be made. Some bugs are solved by [37], others are fixed during the implementation and explained further in this chapter.

4.2.1.1 Fixed Bugs

Mirai uses several tables to write and receive data. A null pointer exception occurs when Mirai tries to access the table `TABLE_KILLER_STATUS` in the `killer.c` module placed in the folder `mirai/bot`. This error can be fixed by first unlocking and then locking the table as demonstrated in Listing 4.1.

```
// Here the table is unlocked
table_unlock_val(TABLE_KILLER_PROC);
table_unlock_val(TABLE_KILLER_STATUS);
table_unlock_val(TABLE_KILLER_EXE);
// The table is locked again
table_lock_val(TABLE_KILLER_PROC);
table_lock_val(TABLE_KILLER_STATUS);
table_lock_val(TABLE_KILLER_EXE);
```

Listing 4.1: Locking and unlocking table TABLE_KILLER_STATUS

4.2.1.2 Bot

The connection between the bot and the other elements showed in the next subsections is visualized in Figure 2.2. Since the source code does not support architecture ARMv7 of the Raspberry Pi 3, the bot also has to be compiled on the IoT device itself. For that, the command in Listing 4.2 is used which creates the `mirai.arm7` file in the folder `mirai/release` ready to be run. This command has to be executed in the `mirai` folder.

```
gcc -std=c99 -DMIRAI_TELNET -DKILLER_REBIND_SSH -static bot/*.c -O3 -fomit-
    frame-pointer -fdata-sections -ffunction-sections -Wl,--gc-sections -o
    release/mirai.arm7 -DMIRAI_BOT_ARCH=\""armv7l"\"
```

Listing 4.2: Compiling the bot on the Raspberry Pi

4.2.1.3 C&C Server

The source code contains entries with the IP address of the C&C server which have to be changed for the bot to connect. Since the VM acts as the C&C server, the IP address used is 10.8.0.3. Listings 4.3 and 4.4 show the adaptations made in the Loader and ScanListen modules located in `loader/src/main.c` and `mirai/tools/scanListen.go`.

```
addrs[0] = inet_addr("10.8.0.3"); // Address to bind to
addrs[1] = inet_addr("10.8.0.3"); // Address to bind to
```

Listing 4.3: IP address change in the Loader

```
func main() {
    l, err := net.Listen("tcp", "10.8.0.3:48101")

}
```

Listing 4.4: IP address change in ScanListen

4.2.1.4 Database

The C&C module has to be connected to the MySQL database in order to write and read data. Examples of database entries include user and attack history. Since both the C&C and the database run on the same computer, they also share the same IP address. The adaptation and the general database information can be seen in Listing 4.5 of the file `main.go` located in the folder `mirai/cnc`. After setting up the database, three tables have to be created: *history*, *users* and *whitelist*. Then a user with the credentials "admin" / "password" is added to the *user* table. This user is later used to connect to the C&C server. The SQL command to add the user looks like demonstrated in Listing 4.6.

```
const DatabaseAddr string   = "127.0.0.1"
const DatabaseUser string   = "root"
const DatabasePass string   = "password"
const DatabaseTable string  = "mirai"

func main() {
    tel, err := net.Listen("tcp", "10.8.0.3:23")

    api, err := net.Listen("tcp", "10.8.0.3:101")

    }
```

Listing 4.5: Setting up the database

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

Listing 4.6: SQL Command for creating a new C&C user

After all these adaptations, Mirai is ready to be built with the following command: `./build.sh release telnet`. It is important to execute it in the `mirai` folder. After the execution is finished a new folder named `/release` containing the files `cnc`, `scanListen` and several bot scripts compatible for different architecture types is created. Running the file `cnc` starts the C&C server which listens to port 23. Through Teletype Network (Telnet), registered users can connect to the server and send attacks to all connected bots after entering their credentials. Figure 4.2 shows the terminal after a successful Telnet connection to the C&C server.

The available attacks can be displayed by entering the command `?` into the terminal. They are listed in Table 4.2 in detail and also demonstrated in Figure 4.3. An example of a full command for executing a UDP-attack looks like demonstrated in Listing 4.7. Additionally, every attack has its specific flag list which is shown in Figure 4.4 presenting the flag list of the UDP command.

```
udp 192.168.1.220 10 len=100
```

Listing 4.7: UDP-attack command on Mirai

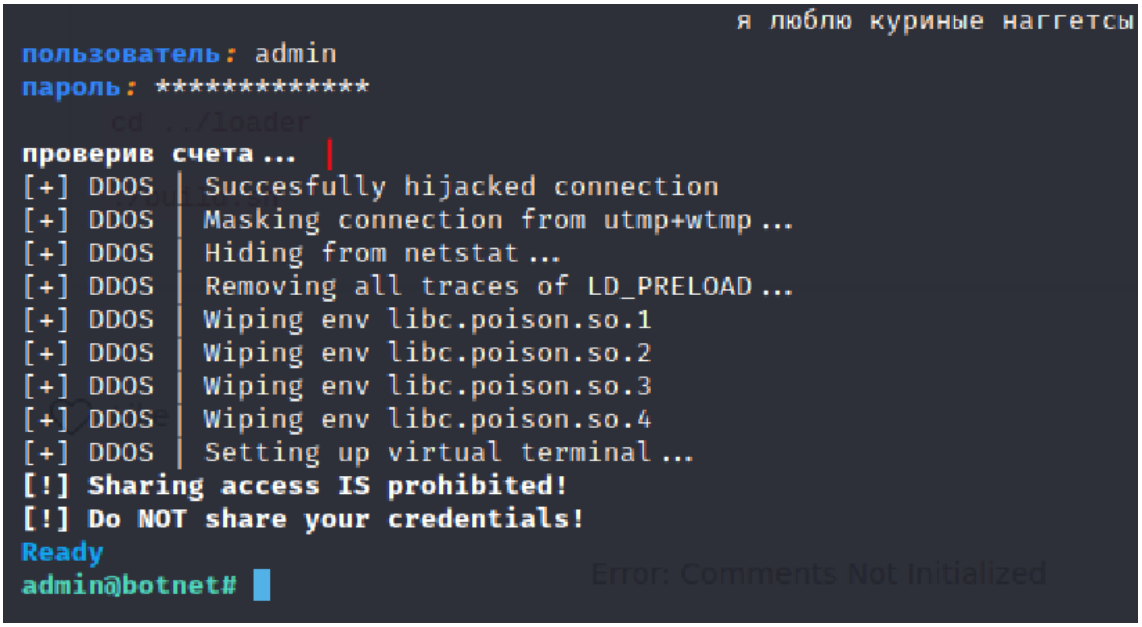


Figure 4.2: Terminal showing a successful connection to the Mirai C&C server.

Table 4.2: Available attack types with Mirai and the corresponding commands

Attack Type	Command
ACK Flood	`ack`
DNS Resolver Flood	`dns`
GRE Ethernet Flood	`greeth`
GRE IP Flood	`greip`
HTTP Flood	`http`
SYN Flood	`syn`
TCP Stomp Flood	`stomp`
UDP Flood	`udpplain` or `udp`

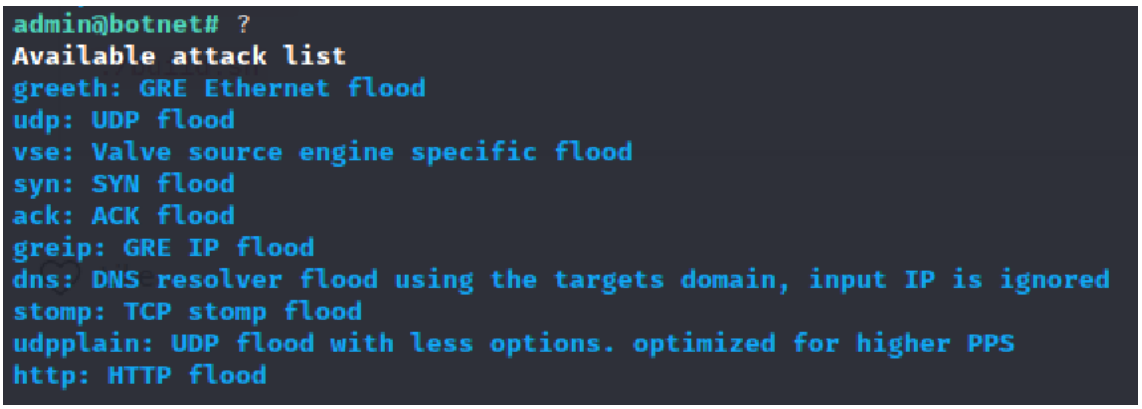


Figure 4.3: Mirai terminal showing all available attack types

```
admin@botnet# udp 10.3.25.130 5 ?
List of flags key=val seperated by spaces. Valid flags for this method are

tos: TOS field value in IP header, default is 0
ident: ID field value in IP header, default is random
ttl: TTL field in IP header, default is 255
len: Size of packet data, default is 512 bytes
rand: Randomize packet data content, default is 1 (yes)
df: Set the Dont-Fragment bit in IP header, default is 0 (no)
sport: Source port, default is random
dport: Destination port, default is random
source: Source IP address, 255.255.255.255 for random

Value of 65535 for a flag denotes random (for ports, etc)
Ex: seq=0
Ex: sport=0 dport=65535
```

Figure 4.4: Flags of the UDP-attack listed on the Mirai-C&C terminal

4.2.2 Bashlite

The source code of Bashlite used for this thesis is made available by Fei Ding on Github [12]. A few small adaptions have to be made to the code to be able to run the botnet.

4.2.2.1 Client

The source code of Bashlite consists of the files `client.c` and `server.c`. For the bot to properly connect to the C&C server, the IP address in `client.c` has to be changed to the address of the VM. This is demonstrated in Listing 4.8.

```
unsigned char *commServer[] =
{
        "10.8.0.3" //This is the IP of
                   //the command server (you'll need to change this)

};
```

Listing 4.8: Changing the IP address of the Bashlite C&C server

4.2.2.2 Server

To connect to the C&C server, a Telnet connection on port 8888 is needed with the correct management password. Listing 4.9 shows how both can be changed in `server.c`.

```
#define MY_MGM_PASS "password"
#define MY_MGM_PORT 8888
```

Listing 4.9: Management Bashlite

After compiling and executing the server, the attacker can connect through Telnet using the specified IP address and Management-Port: `telnet 10.8.0.3 8888`. Figure 4.5 shows the terminal after a successful login to the C&C server.

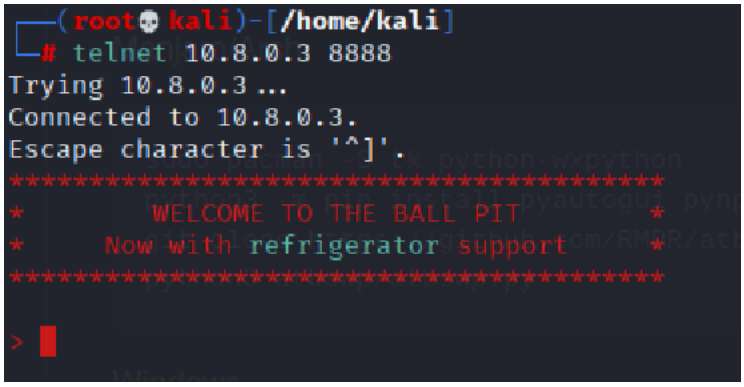


Figure 4.5: Bashlite C&C server terminal after a successful connection

The attacker is now ready to send attack commands to the bots. The chosen variant of Bashlite supports four attack types which are listed in Table 4.3 with the right command syntax.

Table 4.3: Available attack types with Bashlite and the corresponding commands

Attack Type	Command
Hold Flood	`! HOLD {ip_address} {port} {time}`
Junk Flood	`! JUNK {ip_address} {port} {time}`
TCP Flood	`! TCP {target} {port} {time} {netmask} {flags (syn, ack, psh, rst, fin, all) {packet size} {time poll interval}`
UDP Flood	`! UDP {target} {port} {time} {netmask} {packet size} time poll interval}`

The Bashlite source code also has some commands to check the connection to the bot or to interrupt already started attacks:

- `! PING` to check the connection to the bot
- `! GETLOCALIP` to receive the IP address of the bot
- `! SCANNER ON/OFF` to turn the scanner on/off
- `! KILLATTK` to interrupt an ongoing attack
- `! LOLNOGTFO` to stop the bot

4.3 Monitored Events

As already explained in Chapter 2, the main goal of botnets is to launch DDoS attacks and consequently overwhelm chosen servers. Several different protocols can be used for

such an attack. A bot creates network traffic by sending protocol packets to the victim. Hence, monitoring the network traffic is one of the most crucial methods to determine if a device is being used by a botmaster. Apart from that, botnets can also have an impact on the CPU and memory consumption. Since a script is executed on the IoT device, there can also be changes to system reads and writes. All these resource families can be analyzed to detect anomalies on a device at an early stage.

For the monitoring of the sensor, the bash script `create_sample_dataset.sh` was created. The script executes a *Perf* command to monitor chosen events on the device. *Perf* is a monitoring tool for Linux devices and has a wide range of events that can be monitored [29]. During the execution of the script, a sample was created every 5 seconds, converted to CSV format, and written into a file. Since botnets can affect different resources on a device, events belonging to the families CPU and memory usage, network, file systems, and the scheduler were selected to be monitored. In Table 4.4 these events are listed grouped by their family.

Table 4.4: All monitored Perf events grouped by resource families.

Family	**Perf Events**	
Network	fib:fib_table_lookup	sock:inet_sock_set_state
	udp:udp_fail_queue_rcv_skb	tcp:tcp_destroy_sock
	qdisc:qdisc_dequeue	tcp:tcp_probe
	net:net_dev_queue	skb:kfree_skb
	net:netif_rx	skb:consume_skb
	net:net_dev_xmit	skb:skb_copy_datagram_iovec
Memory	page-faults	kmem:kmem_cache_alloc
	pagemap:mm_lru_insertion	kmem:kmem_cache_free
	kmem:kfree	kmem:mm_page_alloc_zone_locked
	kmem:kmalloc	kmem:mm_page_pcpu_drain
	kmem:mm_page_free	kmem:mm_page_alloc
File Systems	jbd2:jbd2_handle_start	writeback:wbc_writepage
	jbd2:jbd2_start_commit	writeback:writeback_dirty_inode
	filemap:mm_filemap_add_to_page_cache	writeback:writeback_pages_written
	block:block_dirty_buffer	writeback:writeback_single_inode
	block:block_bio_backmerge	writeback:writeback_dirty_page
	block:block_bio_remap	writeback:writeback_write_inode
	block:block_unplug	writeback:writeback_mark_inode_dirty
	block:block_touch_buffer	writeback:writeback_dirty_inode_enqueue
	block:block_getrq	writeback:writeback_written
	cachefiles:cachefiles_create	writeback:global_dirty_state
	cachefiles:cachefiles_lookup	writeback:sb_clear_inode_writeback
	cachefiles:cachefiles_mark_active	
CPU	clk:clk_set_rate	rpm:rpm_resume
	ipi:ipi_raise	rpm:rpm_suspend
Scheduler	cs	task:task_newtask
	cpu-migrations	sched:sched_process_exec
	signal:signal_deliver	sched:sched_process_free
	signal:signal_generate	sched:sched_process_wait
	alarmtimer:alarmtimer_fired	sched:sched_switch
	alarmtimer:alarmtimer_start	sched:sched_wakeup

4.4 Datasets Creation

Creating datasets with the internal behavior of IoT devices is the first step to be able to detect malware. This section describes how the datasets were created looking at the internal behavior of a Raspberry Pi 3 acting as an ElectroSense spectrum sensor.

One malware-detection approach is the detection of anomalies in the resource usage of a device. Anomalies caused by botnets are described in Section 4.3. An algorithm programmed to detect malicious software on a machine needs to understand what is considered normal to be able to distinguish anomalous behavior from normal behavior. The datasets created during these thesis allow future researchers to train ML and DL based algorithms and enable the detection of known as well as unknown malware on IoT devices.

For the creation of the datasets, the monitoring script explained in Section 4.3 was used. The Raspberry Pi has a specific normal behavior since it communicates with ElectroSense and receives spectrum data. First, this normal behavior of the device was monitored. While monitoring, the device continued receiving spectrum data and sending it to the ElectroSense backend which makes mostly use of resources like CPU and network usage [10]. The normal behavior was monitored twice, first during two days and later during one day.

After monitoring the normal behavior, the device was infected with Mirai. From all attacks available on Mirai listed in Table 4.2, UDP, TCP Stomp, TCP SYN, and TCP ACK were chosen to be monitored since the most common DDoS attacks are made using the protocols TCP and UDP. During the attacks, the Raspberry Pi sensor sent network packets to devices in the same network to overwhelm them by overusing their resources. The DDoS attack types are explained in more detail in Section 2.2. Each attack was monitored two times for approximately two hours each. The victims of the attacks were randomly chosen devices connected to the same network. Since the bot program does only stay in memory, rebooting the Raspberry Pi was enough to remove the malware in order to infect it again with Bashlite. It was important to remove the first malware from the device to make sure that they would not interfere with each other. Infected with Bashlite, all four attacks shown in Table 4.3 were monitored but the focus was laid on UDP and TCP attacks. Each attack was monitored two times between one and two hours each. During all attacks, the sensor was communicating with ElectroSense while also launching attacks against defined devices which increased the network traffic and other resources.

Table 4.5 lists all created datasets modeling the behavior of the ElectroSense sensor grouped by the behavior of the device.

Table 4.5: All created datasets grouped by the behavior. The column Monitoring refers to the first and second monitoring of the same behavior.

Dataset	**Behavior**		**Monitoring**
	Malware	**Attack**	
1	-	Normal	1
2	-	Normal	2
3	Mirai	UDP	1
4	Mirai	UDP	2
5	Mirai	TCP Stomp	1
6	Mirai	TCP Stomp	2
7	Mirai	TCP SYN	1
8	Mirai	TCP SYN	2
9	Mirai	TCP ACK	1
10	Mirai	TCP ACK	2
11	Bashlite	UDP	1
12	Bashlite	UDP	2
13	Bashlite	TCP	1
14	Bashlite	TCP	2
15	Bashlite	HOLD	1
16	Bashlite	HOLD	2
17	Bashlite	JUNK	1
18	Bashlite	JUNK	2

Chapter 5

Evaluation

This chapter evaluates the datasets created in Chapter 4 and demonstrates the differences between normal and infected behavior. The datasets containing the malware behavior show significant changes compared to the normal behavior and help to train malware detection algorithms.

The first step in the evaluation of the datasets was to get rid of events that were not stable during monitoring. For this, the mean values of each of the normal behavior datasets 1 and 2 were calculated. Then, the values of each event in dataset 2 were subtracted from the values in dataset 1. To see the fluctuations of each event during the normal behavior in relation to each other, the difference values were divided by the values of dataset 1. The result of this is shown in Figure 5.1. Since the internal behavior of a sensor can fluctuate due to sensing information and communication with the backend, events showing strong differences were considered unstable. These events were *block:block_dirty_buffer*, *block:block_touch_buffer*, *filemap:mm_filemap_add_to_page_cache*, *jbd2:jbd2_handle_start*, and *writeback:writeback_dirty_page* and were excluded from all further analysis. This shows that resources belonging to the file systems family fluctuated strongly in the normal behavior of an ElectroSense sensor are therefore not good indicators of malicious behavior. The exclusion of the unstable events helped to spot events that changed through the malware instead of through natural fluctuation. That way, resources on the sensor affected by botnets could be identified. Figure 5.2 illustrates again the differences between the datasets 1 and 2 but without the unstable events. In further analysis, only the stable events were considered for the detection of variation between normal and infected behavior.

In the next step, the differences between the normal behavior and the infected behavior were analyzed. This was done by calculating the mean values of the datasets and subtracting the values of each malware dataset from the normal behavior. For direct comparison, attacks using the UDP and TCP protocols were chosen to visualize both botnets. Figure 5.3 demonstrates the most significant events during UDP and TCP attacks with Mirai. As seen in Figure 5.3a and Figure 5.3b, during both monitored UDP attacks, the outstanding events were *fib:fib_table_lookup*, *skb:skb_copy_datagram_iovec*, and *tcp:tcp_destroy_sock*. In general, dataset 3 presented a higher variation of the events than dataset 4 and had some additional significant events belonging to the network family. The

UDP attack also changed the CPU usage which is illustrated by *rpm:rpm_resume*. From this emerges that network events are good signifiers in the detection of the Mirai botnet. For better visualization, only the events with the most extrem differences are displayed in the figures.

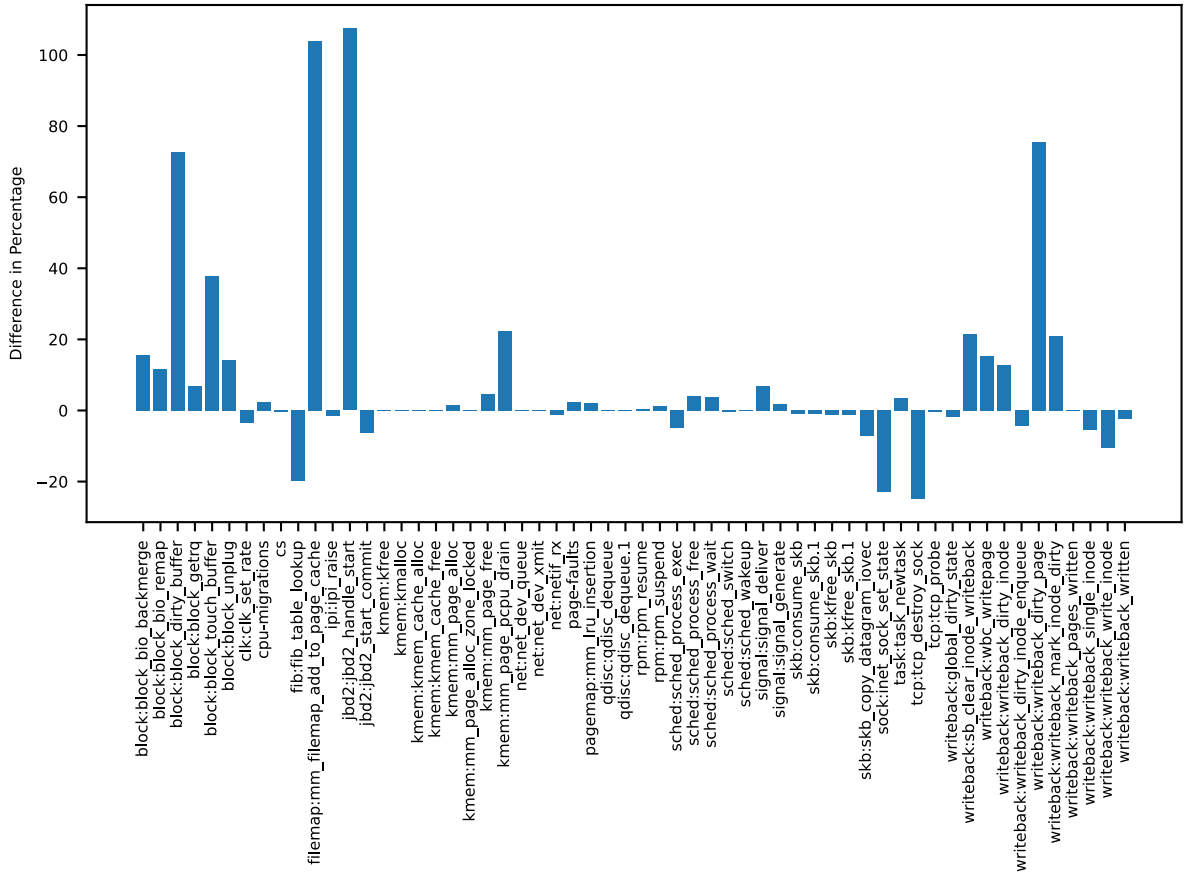


Figure 5.1: Differences between two normal behavior datasets (datasets 1 and 2) of an ElectroSense sensor in percentage. Events standing out are considered as unstable.

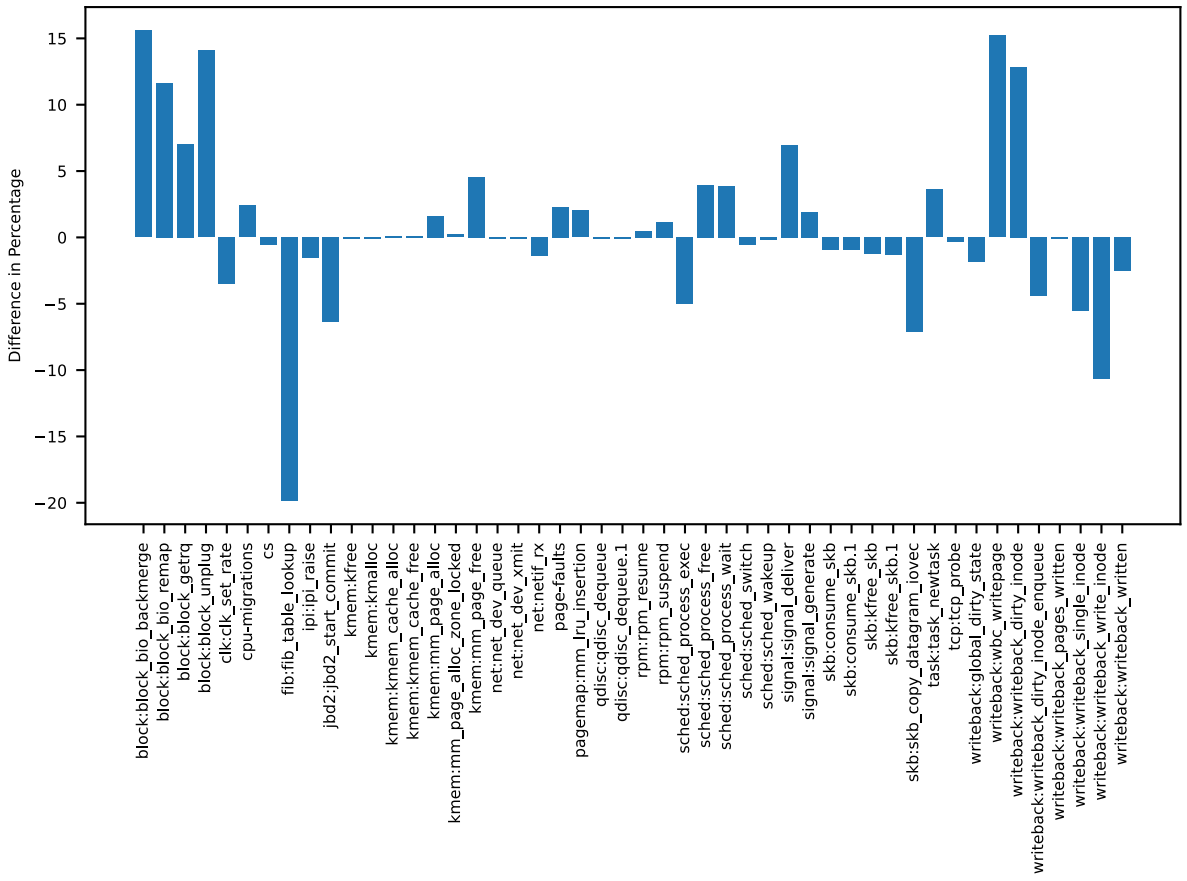
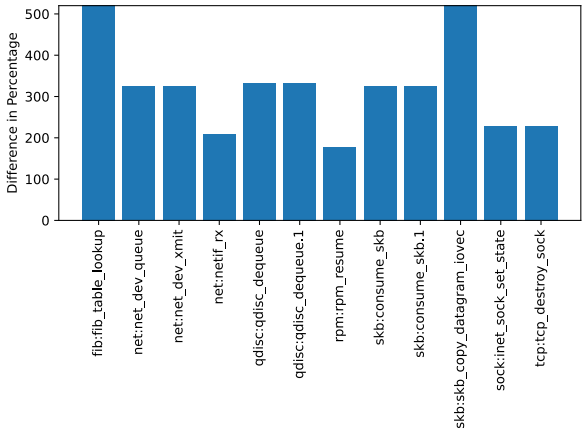
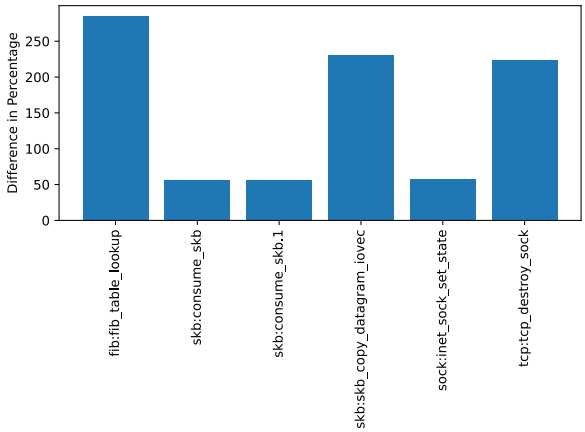


Figure 5.2: Differences between two normal behavior datasets (datasets 1 and 2) of an ElectroSense sensor in percentage without unstable events. Only stable events are considered in the identification of possible malware anomalies.

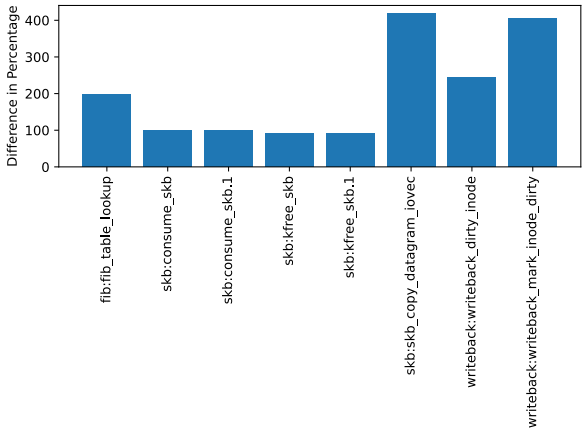
The most extrem events of the TCP attacks visualized in Figure 5.3c and Figure 5.3d were the same events as during UDP attacks and the figures show that similar changes occurred during both monitoring phases. In contrast to UDP attacks, also some significant events belonging to the memory usage and file systems family were present with differences being more than 100%.



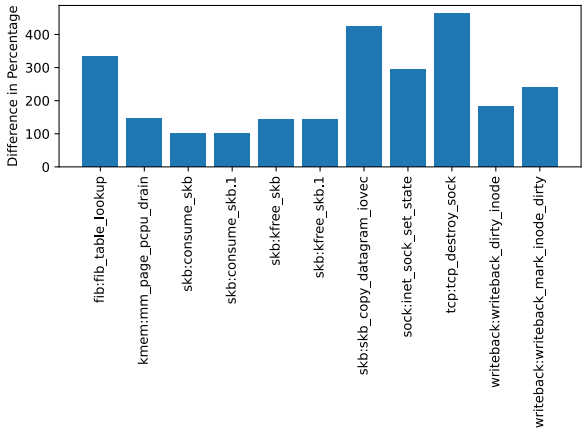
(a) Normal versus UDP behavior of Mirai (dataset 3)



(b) Normal versus UDP behavior of Mirai (dataset 4)



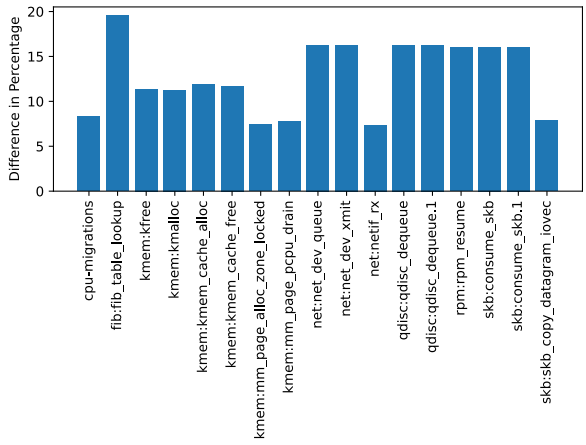
(c) Normal versus TCP behavior of Mirai (dataset 5)



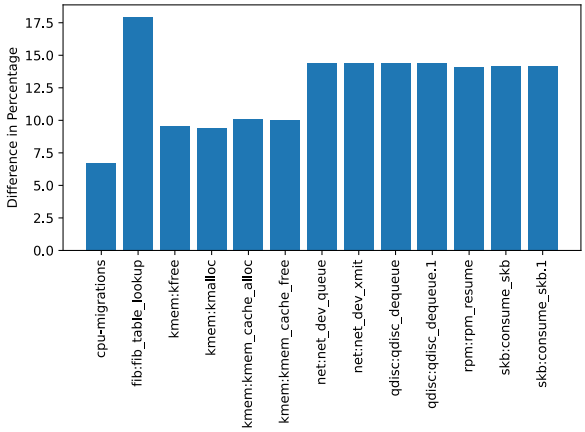
(d) Normal versus TCP behavior of Mirai (dataset 6)

Figure 5.3: Differences between the normal behavior and the behavior during Mirai's UDP and TCP attacks of the sensor. The y-axis represents the difference in percentage.

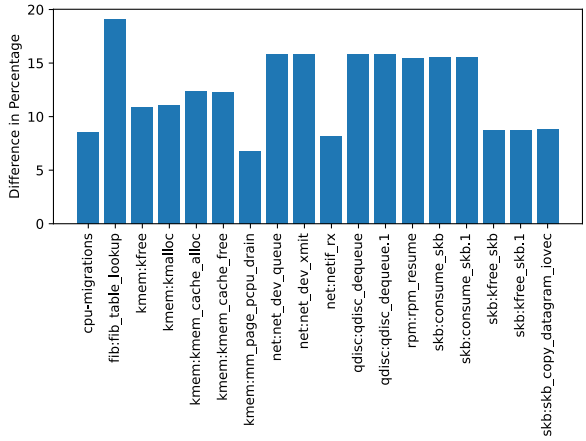
The same behavior could be spotted again during UDP and TCP attacks with Bashlite. In this sense, Figure 5.4 shows that the outstanding events were from the network family as already discovered during the Mirai attacks. However, the differences during the Bashlite attacks surpassed the differences during the Mirai attacks to a large extent. That is why, to have a more readable visualization, the y-axes of the figures are log-scaled. This showed how a machine infected with Bashlite would have much more significant indicators than one infected with Mirai. Additionally, some events belonging to the file systems family were present. Bashlite also showed some changes in the CPU usage of the device which is illustrated by the event *rpm:rpm_resume*.



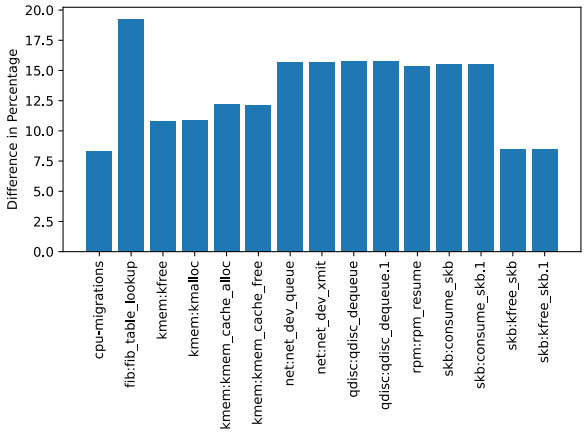
(a) Normal versus UDP behavior of Bashlite (dataset 11)



(b) Normal versus UDP behavior of Bashlite (dataset 12)



(c) Normal versus TCP behavior of Bashlite (dataset 13)



(d) Normal versus TCP behavior of Bashlite (dataset 14)

Figure 5.4: Differences between the normal behavior and the behavior during Bashlite's UDP and TCP attacks of the sensor. The y-axis represents the difference in percentage in log-scale.

Furthermore, other Mirai and Bashlite attacks were evaluated. The summary of them is found in Table 5.1. The table demonstrates the top few events that stood out during the attacks and confirms the already observed aspects of both botnets since the events belong mostly to the network family.

Table 5.1: The difference of the significant events between normal and infected behavior in percentage. The relevant datasets and attack types along with the malware are given. Column "Difference" describes the values in the datasets listed in column "Datasets".

Malware	**Attack Type**	**Events**	**Data sets**	**Difference**
Mirai	TCP ACK	fib:fib_table_lookup	9, 10	192%, 132%
		skb:skb_copy_datagram_iovec		334%, 256%
	TCP SYN	fib:fib_table_lookup	7, 8	1'542%, 167%
		skb:skb_copy_datagram_iovec		391%, 257%
		writeback:writeback_mark_inode_dirty		312%, 105%
Bashlite	HOLD	tcp:tcp_destroy_sock	15, 16	25'488%, 435%
		sock:inet_sock_set_state		10'316%, 120%
		fib:fib_table_lookup		2'662%, 117%
	JUNK	tcp:tcp_destroy_sock	17, 18	32'087%, 42'135%
		sock:inet_sock_set_state		12'716%, 16'415%
		fib:fib_table_lookup		5'323%, 6'684%

Although Bashlite showed higher variation in resource usage, similar mannerisms were detected with both Mirai and Bashlite. This is unsurprising and only confirms that malware related to other malware show similar behavior and implies that, with collecting data from known botnets, malware detection algorithms can be trained to be able to detect not yet known botnets.

In conclusion, the most significant indicator of the spectrum sensor being connected to a botnet was network traffic. These datasets are a good step in the direction of being capable of detecting botnets on IoT devices used by crowdsensing platforms as well as IoT devices running Linux OS in general. Since there were visible changes in the internal behavior of the used device, the datasets can be of use to train malware detection algorithms.

Chapter 6

Summary

6.1 Summary and Conclusion

The goal of this thesis was to create datasets with the normal and the Mirai- and Bashlite-infected behavior of a Raspberry Pi acting as an ElectroSense spectrum sensor as a first step towards the detection of botnet-activity on an IoT device. During the modeling of the behavior, network traffic, CPU and memory usage, and file systems data were collected.

In order to achieve this goal, an ElectroSense sensor had to be set up by registering the device on the website, adding a static IP address, and opening ports 22 and 23 for SSH and Telnet. In the next step, both botnets had to be executed on the sensor. For the C&C server of both malware, a VM running Kali-Linux was created and the Raspberry Pi was used as a bot. This gave the basis for launching attacks starting from the zombie device and creating datasets with the internal behavior of the sensor.

After the implementation of the monitoring script, the behavior of the sensor was modeled. In total, 18 datasets were created consisting of the normal behavior and the behavior after the infection with botnets. Every dataset contains data about network traffic, file systems and CPU and memory usage. The datasets show how these resources increased during botnet attacks and lay a foundation for the training and deployment of malware detection algorithms.

In the evaluation of these datasets, the changes in the behavior were made visible by plotting the relative difference values between the means of the normal behavior and the behavior during the execution of several attacks. Some events stood out that showed botnet activity which mostly belonged to network traffic but also to file systems and CPU usage. Both botnets showed big differences, mostly on the same events. This showed that by analyzing already known botnets, even the detection of unknown botnets using malware detection algorithms is possible.

Generally, the goal of this thesis could be achieved by creating the relevant datasets. This successfully shows that malware detection algorithms can be trained to detect anomalies in the behavior of crowdsensing sensors using these datasets and analyzing the data.

6.2 Future Work

This thesis focused on the two botnets Mirai and Bashlite which are only a small fraction of numerous Linux-Botnets infecting devices. One device was used as a bot and an additional one was created to serve as a C&C server. In general, the source codes of both botnets are available online for everyone and can be analyzed using techniques such as reverse engineering. For future work, a bigger number of botnets affecting Linux devices could be considered to create larger datasets for training malware detection algorithms. Especially, collecting data from several crowdsensing sensors over a longer period could give even more meaningful results. Monitoring not only the bots launching DDoS attacks but also the connection to the C&C server could yield interesting insights as well and contribute to more secure IoT devices.

Bibliography

[1] Jawad Ali et al. "Towards a secure behavior modeling for IoT networks using Blockchain". In: *arXiv preprint arXiv:2001.01841* (2020).

[2] *AlibabaCloud*. URL: `https://eu.alibabacloud.com` (visited on July 30, 2021).

[3] *Amazon Web Services*. URL: `https://aws.amazon.com/` (visited on July 30, 2021).

[4] Amar Amouri, Vishwa T Alaparthy, and Salvatore D Morgera. "Cross layer-based intrusion detection based on network behavior for IoT". In: *2018 IEEE 19th Wireless and Microwave Technology Conference (WAMICON)*. IEEE. 2018, pp. 1–4.

[5] Manos Antonakakis et al. "Understanding the mirai botnet". In: *26th {USENIX} security symposium ({USENIX} Security 17)*. 2017, pp. 1093–1110.

[6] Amin Azmoodeh, Ali Dehghantanha, and Kim-Kwang Raymond Choo. "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning". In: *IEEE transactions on sustainable computing* 4.1 (2018), pp. 88–95.

[7] *BASHLITE botnets Ensnare 1 Million iot devices*. 2016. URL: `https://www.securityweek.com/bashlite-botnets-ensnare-1-million-iot-devices` (visited on July 14, 2021).

[8] Vitor Hugo Bezerra et al. "Providing IoT host-based datasets for intrusion detection research". In: *Anais do XVIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*. SBC. 2018, pp. 15–28.

[9] Agathe Blaise et al. "BotFP: Fingerprints clustering for bot detection". In: *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2020, pp. 1–7.

[10] Roberto Calvo-Palomino et al. "Electrosense+: Crowdsourcing radio spectrum decoding using IoT receivers". In: *Computer Networks* 174 (2020), p. 107231.

[11] *Collaborative spectrum Monitoring*. URL: `https://electrosense.org/#!/` (visited on July 13, 2021).

[12] Fei Ding. *Bashlite-Source-Code*. 2017. URL: `https://github.com/ifding/iot-malware/tree/master/BASHLITE`.

[13] Dave Evans. "The internet of things: How the next evolution of the internet is changing everything". In: *CISCO white paper* 1.2011 (2011), pp. 1–11.

[14] *Evolution of internet of things (iot): Past, present and future*. 2021. URL: `https://www.techaheadcorp.com/knowledge-center/evolution-of-iot/` (visited on July 13, 2021).

[15] Roman Ferrando and Paul Stacey. "Classification of device behaviour in internet of things infrastructures: towards distinguishing the abnormal from security threats". In: *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*. 2017, pp. 1–7.

[16] Jerry Gamblin. *Mirai-Source-Code*. 2017. URL: https://github.com/jgamblin/Mirai-Source-Code.

[17] Tomer Golomb, Yisroel Mirsky, and Yuval Elovici. "CIoTA: Collaborative IoT anomaly detection via blockchain". In: *arXiv preprint arXiv:1803.03807* (2018).

[18] Roger A. Grimes. *9 types of malware and how to recognize them*. 2020. URL: https://www.csoonline.com/article/2615925/security-your-quick-guide-to-malware-types.html (visited on July 13, 2021).

[19] Brij B. Gupta and Amrita Dahiya. "Ddos attacks on various platforms". In: *Distributed Denial of Service (DDoS) Attacks* (2021), pp. 75–98.

[20] Ibbad Hafeez et al. "IoT-KEEPER: Detecting malicious IoT network activity using online traffic analysis at the edge". In: *IEEE Transactions on Network and Service Management* 17.1 (2020), pp. 45–59.

[21] Waqas Haider et al. "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling". In: *Journal of Network and Computer Applications* 87 (2017), pp. 185–192.

[22] Salma Abdalla Hamad et al. "IoT device identification via network-flow based fingerprinting and learning". In: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE. 2019, pp. 103–111.

[23] Ayyoob Hamza et al. "Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity". In: *Proceedings of the 2019 ACM Symposium on SDN Research*. 2019, pp. 36–48.

[24] Mahmudul Hasan et al. "Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches". In: *Internet of Things* 7 (2019), p. 100059.

[25] Sen He et al. "BoSMoS: A blockchain-based status monitoring system for defending against unauthorized software updating in industrial Internet of Things". In: *IEEE Internet of Things Journal* 7.2 (2019), pp. 948–959.

[26] Imperva. *Mirai DDoS Attack Explained*. 2017. URL: https://www.imperva.com/blog/how-to-identify-a-mirai-style-ddos-attack/ (visited on July 14, 2021).

[27] Hyunjae Kang et al. "IoT network intrusion dataset". In: *IEEE Dataport* (2019).

[28] Constantinos Kolias et al. "DDoS in the IoT: Mirai and other botnets". In: *Computer* 50.7 (2017), pp. 80–84.

[29] *Main page*. URL: https://perf.wiki.kernel.org/index.php/Main_Page (visited on July 19, 2021).

[30] Artur Marzano et al. "The evolution of bashlite and mirai iot botnets". In: *2018 IEEE Symposium on Computers and Communications (ISCC)*. IEEE. 2018, pp. 00813–00818.

[31] Yair Meidan et al. "Detection of unauthorized IoT devices using machine learning techniques". In: *arXiv preprint arXiv:1709.04647* (2017).

[32] Yair Meidan et al. "N-baiot-network-based detection of iot botnet attacks using deep autoencoders". In: *IEEE Pervasive Computing* 17.3 (2018), pp. 12–22.

[33] Trend Micro. *BASHLITE affects devices running on busybox*. 2014. URL: https://www.trendmicro.com/en_us/research/14/k/bashlite-affects-devices-running-on-busybox.html.

[34] *Microsoft Azure*. URL: https://azure.microsoft.com/ (visited on July 30, 2021).

[35] *Mirai source code release leads to huge increase in botnet*. 2017. URL: `https://www.pindrop.com/blog/mirai-source-code-release-leads-to-huge-increase-in-botnet/` (visited on July 14, 2021).

[36] Thien Duc Nguyen et al. "DÏoT: A federated self-learning anomaly detection system for IoT". In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2019, pp. 756–767.

[37] Maria Fernanda Ojeda Adan. "Designing an Internet of Things Attack Simulator". In: (2019).

[38] Ori Or-Meir et al. "Dynamic Malware Analysis in the Modern Era-A State of the Art Survey". In: *ACM Comput. Surv.* 52.5 (Sept. 2019).

[39] Jesus Pacheco and Salim Hariri. "Anomaly behavior analysis for IoT sensors". In: *Transactions on Emerging Telecommunications Technologies* 29.4 (2018), e3188.

[40] Marc-Oliver Pahl and François-Xavier Aubet. "All eyes on you: Distributed Multi-Dimensional IoT microservice anomaly detection". In: *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE. 2018, pp. 72–80.

[41] A Parmisano, S Garcia, and MJ Erquiaga. "Aposemat IoT-23: A labeled dataset with malicious and benign IoT network traffic". In: *Accessed: Jul* 31 (2020), p. 2020.

[42] Sreeraj Rajendran et al. "Electrosense: Open and big spectrum data". In: *IEEE Communications Magazine* 56.1 (2017), pp. 210–217.

[43] Pedro M Sánchez Sánchez et al. "BEHACOM-a dataset modelling users' behaviour in computers". In: *Data in brief* 31 (2020), p. 105767.

[44] Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman. "Detecting behavioral change of IoT devices using clustering-based network traffic modeling". In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7295–7309.

[45] *Static data vs. dynamic data: Why companies must transition*. 2018. URL: `https://blog.zoominfo.com/dynamic-data/` (visited on July 19, 2021).

[46] RD Statista. "Internet of Things-Number of connected devices worldwide 2019-2030". In: *Statista Research Department* (2019).

[47] Laya Taheri, Andi Fitriah Abdul Kadir, and Arash Habibi Lashkari. "Extensible android malware detection and family classification using network-flows and API-calls". In: *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE. 2019, pp. 1–8.

[48] KS Vanitha, SV Uma, and SK Mahidhar. "Distributed denial of service: Attack techniques and mitigation". In: *2017 International Conference on Circuits, Controls, and Communications (CCUBE)*. IEEE. 2017, pp. 226–231.

[49] Waqas. *BASHLITE malware turning millions of Linux Based iot devices into DDoS botnet*. 2018. URL: `https://www.hackread.com/bashlite-malware-linux-iot-ddos-botnet/` (visited on July 19, 2021).

[50] Tianlong Yu et al. *RADAR: A robust behavioral anomaly detection for IoT devices in enterprise networks*. Tech. rep. Tech. rep., CMU CyLab, 2019. 103, 2019.

Abbreviations

ACK	Acknowledgement
AD	Anomaly Detection
C	Classification
CPU	Central Processing Unit
C&C	Command and Control
DDoS	Distributed Denial of Service
DL	Deep Learning
DNS	Domain Name Server
DoS	Denial of Service
GB	Gigabyte
GHz	Gigahertz
GRE	Generic Routing Encapsulation
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IIoT	Industrial Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol Version 4
ML	Machine Learning
OS	Operating System
RAM	Random Access Memory
SDR	Software Defined Radio
SSH	Secure Shell
SYN	Synchronisation
TCP	Transmission Control Protocol
Telnet	Teletype Network
UDP	User Datagram Protocol
VM	Virtual Machine

List of Figures

List of Tables

Listings

Appendix A

Contents of the ZIP file

The following files are contained in the ZIP file.

Documentation

- Thesis as a PDF file
- Thesis as a .tex file (Latex)
- Midterm presentation as PPTX

Code

- Mirai source code
- Bashlite source code
- Monitoring script `create_sample_dataset.sh`

Datasets

The datasets are in CSV format.

- Datasets 1 and 2 containing the normal behavior
- Datasets 3 and 4 containing the behavior during a UDP attack with Mirai
- Datasets 5 and 6 containing the behavior during a TCP attack with Mirai
- Datasets 7 and 8 containing the behavior during a TCP SYN attack with Mirai
- Datasets 9 and 10 containing the behavior during a TCP ACK attack with Mirai
- Datasets 11 and 12 containing the behavior during a UDP attack with Bashlite
- Datasets 13 and 14 containing the behavior during a TCP attack with Bashlite
- Datasets 15 and 16 containing the behavior during a HOLD attack with Bashlite
- Datasets 17 and 18 containing the behavior during a JUNK attack with Bashlite