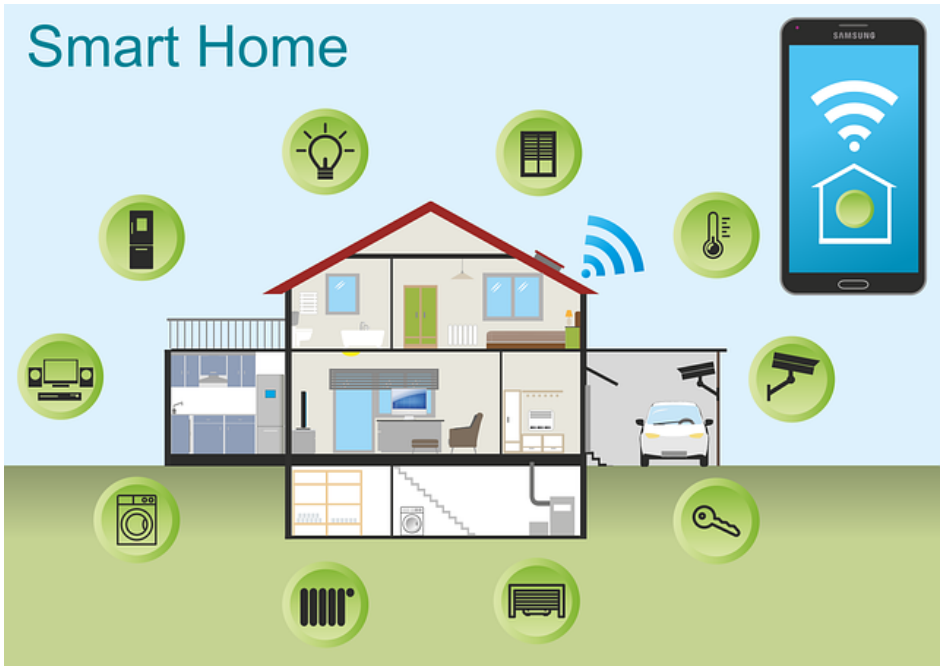


On Security Threats and Solutions for the Future Smart Home

Exploring the Viability of Intrusion Detection in a Centralized Smart Hub

Master's thesis in Computer Systems and Networks

MATZ LARSSON
LISA LIPKIN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

On Security Threats and Solutions for the Future Smart Home

Exploring the Viability of Intrusion Detection in a Centralized Smart Hub

MATZ LARSSON
LISA LIPKIN

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

On Security Threats and Solutions for the Future Smart Home
Exploring the Viability of Intrusion Detection in a Centralized Smart Hub
MATZ LARSSON
LISA LIPKIN

Advisor: Johnny Väyrynen, Scionova
Supervisor: Magnus Almgren, Department of Computer Science and Engineering
Examiner: Philippas Tsigas, Department of Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The smart home. Pixaline.

Typeset in LaTeX
Gothenburg, Sweden 2018

On Security Threats and Solutions for the Future Smart Home
Exploring the Viability of Intrusion Detection in a Centralized Smart Hub
MATZ LARSSON
LISA LIPKIN
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Security in smart homes is becoming increasingly important as more and more devices are becoming available on the market and the information that they handle is of a sensitive nature. Historical attacks have shown that smart homes can contribute to attacks that could have severe consequences both for their users and for society. While much work is being done on finding security solutions for IoT and smart homes, there has yet to be a consensus on what solutions are optimal. In this thesis we present the state of research in the field and introduce a potential security solution for the smart home. The security solution, called the security supervisor, is a network-based solution located in a smart hub which detects malicious activity within a smart home network. Detection of two attacks are implemented as a proof-of-concept, namely botnet device detection and Evil-Twin attack detection. The results show that the security supervisor is able to detect the former but not the latter, but the authors argue that a similar platform would be able to contribute to increased security and mitigate many threats towards smart homes.

Keywords: IoT, security, smart home, smart hub

Acknowledgements

First, we would like to thank Scionova for giving us the opportunity to write our master thesis with their company along with a lot of resources including test bed material and supervisor time. We would also like to give a special thanks to Johnny Väyrynen for his help and guidance on both theoretical and practical matters.

Furthermore, we would like to thank Magnus Almgren, our supervisor from Chalmers, for taking his time to supervise us. It was a bumpy road to get the project started, and he was there to support us with time, knowledge and entertaining stories about PostNord. We would also like to thank Philippas Tsigas for taking the time to be our examiner.

Finally, we want to give a shout-out to our partners, families and cats who have been there for us and supported us in making this master thesis. This would not have been possible without you.

Matz Larsson, Gothenburg, Nov 2018
Lisa Lipkin, Gothenburg, Nov 2018

Contents

List of Figures

List of Tables

1

Introduction

The growth of Internet of Things (IoT) and the spread of smart homes are steadily increasing [6]. In recent years the market has boomed up with new devices, such as smart locks, heating, ventilation and air conditioning (HVAC) systems, and communication protocols such as Zigbee and Z-Wave. Gartner has predicted the number of IoT devices to reach as many as 25 billion in 2020 [7]. Many of these devices will be inside the homes of everyday people [8], bringing technology closer than ever.

However, there have been many setbacks for smart homes regarding their security. There have been several cases where actual devices have been compromised, displaying the importance of high security within this field. Fernandes et al. [9] exploited several vulnerabilities in Samsung SmartThings with accompanying apps to e.g. disable certain functionality and cause fake fire alarms. Schwartz et al. [10] performed black-box testing on 16 smart home devices and recovered passwords from eight of them. Furthermore, there have been additional reports of actual users and companies being hacked. These attacks range from gaining access to baby monitors to accessing casino backend servers through an aquarium thermometer [11, 12, 13].

While these reports make the need for security apparent, developing a smart home product with security in mind can be difficult. The resource-constrained hardware means that old conventional methods, such as computationally heavy encryption, do not work effectively on these devices [14, 15, 16]. Even though there are several initiatives around to provide security to resource-constrained IoT devices [16], much remains to be done [17].

Intrusion detection systems (IDS) is a traditional field of security still in its incipiency for smart homes. Several new intrusion detection methodologies for resource restricted devices have been proposed by researchers. However, their focus is mostly on wireless sensor networks (WSN) or general IoT devices [18, 19, 20] and only a few bring up the case of the smart home. As far as we know, none of these methodologies take the smart hub, a central entity in many smart homes, into account. Exploring detection of threats in the hub may contribute to the overall security situation in smart homes since it increases awareness of threats among users.

1.1 Purpose of the thesis

The purpose of this thesis work is twofold. Firstly, it summarizes the security threats to current and future smart homes. Secondly, with this information as a basis, it provides a contribution to manage the security overview of a smart home environment. The contribution is in the form of a security module prototype which focuses on detecting malicious behavior and reports it to the end-user. The thesis will explore if the smart hub is a viable location for such a mechanism to reside in, both concerning resource usage and detection rate.

1.2 Problem description

Security in smart homes is a new and complex field. This section brings up the main problems and challenges with security in smart homes.

Heterogeneity: Smart homes comprises a large number of devices which differ in multiple ways. Firstly, the communication protocols differ between devices, ranging from WiFi and Bluetooth to Zigbee and Z-Wave and many more. Furthermore, the data protocols used vary from device to device and include message queueing telemetry transfer (MQTT) and constrained application protocol (CoAP) amongst others. A general security solution would need to understand and adapt to every protocol. Overall, heterogeneity makes it hard to find one solution that fits every need - especially when numerous new products enter the market every year.

User awareness: The target audience of a smart home is the average person. This target group generally have little awareness of the risks with technology and how to ensure their devices remain uncompromised. Due to this lack of security awareness, customers have little interest in paying more for a more secure product. Consequently the market is not particularly motivated to increase the security in smart home technology. From these arguments one could argue that if the awareness of the end-user increased the entire chain is affected and security in smart home devices will increase.

Position of defense: The positioning of defense mechanisms are troublesome in smart homes since we want to secure the network not only from external threats but internal threats as well. A network provider will never notice whether a device sends malicious data locally on an internal network. A router can not detect if a Z-Wave node abuses the multi-hop feature, in which a node can act as an intermediate router, to alter or drop messages. A specific node does not have the capability to run advanced detection due to its resource restricted properties. From these examples we see that no obvious position for a strong defense mechanism that solves every problem is apparent. We use a smart hub as the basis for our defense mechanism. The smart hub provides a position within the local network, relatively close to the end nodes, allowing for detailed knowledge of the network data flows. In addition,

the smart hub can provide information on the devices' states.

Resource restriction: Even though smart hubs are generally connected to wall sockets, their computational power and memory usage are resource restricted. Consequently, it is not possible to keep a large database over all known devices and how they communicate. Neither is it possible to run computationally heavy algorithms to detect every behavior of the device and find anomalies. To adapt to the resource restriction and use our resources in the most effective way, we need to use a combination of these two approaches.

Based on the presented issues in this section, the following research questions will be treated:

- What are the threats against smart homes according to current research?
- Is it viable to detect security threats against the smart home in a smart hub?
- How can we detect these threats without negatively impacting the performance of the hub noticeably?

1.3 Limitations

The following limitations apply to the project:

- Only research which concern smart homes published post 2010 are included in our surveys in order to limit outdated information. There may be exceptions in special cases, e.g. field defining publications or attack countermeasures that are easily adapted to the smart home setting.
- We prioritize attacks based on e.g. severity, detectability, preventability and implementation difficulty, as time only allow us to focus on a limited number of attacks.
- Our prototype is integrated with only one smart hub, *Home Assistant*, due to the time constraint on the project.
- We do not focus on which media to use for communication with the user, e.g. logging, text-to-speech (TTS), smartphone notifications etc.

1.4 Disclosure of Vulnerabilities

All related parties were informed of discovered weaknesses in products or services before this thesis was published. The aim of this thesis is not to demonstrate how weaknesses in products or systems can be abused, but rather to investigate how they

can be detected in a smart hub.

1.5 Thesis outline

This thesis is structured as follows: Chapter 1 introduces the background, purpose and scope of the thesis. In Chapter 2 we bring up related work in the field. Chapter 3 gives relevant background knowledge concerning smart homes and Chapter 4 presents security in traditional and smart home systems. In Chapter 5 we describe the development of the security module, in Chapter 6 we present the experiments in which we test the security module, and in Chapter 7 we present the results of the tests performed on the system. We discuss the results and future research directions in Chapter 8 before we summarize the thesis in Chapter 9.

2

Related work

Even though the security of smart homes is a rather young research area, many studies on the subject exists. This chapter presents research relevant to this thesis and explain how it relates to this work.

2.1 Risk analyses from other researchers

Risk analyses have long been used to evaluate risks and threats in a system and several methods on how to perform these analyzes exist on the market. However, Nurse et al. [21] claim there is a need for new approaches for conducting risk analyzes within IoT systems as many of the traditional risk assessment methodologies were developed prior to the pervasive cyber-physical networks currently deployed. Consequently, these methodologies have weaknesses when assessing modern IoT environments. Due to these factors we focused on risk analyses that specifically targeted IoT or smart homes.

In 2016 Jacobsson et al. [1] conducted a smart home project for which they did a rigorous risk analysis focused on IoT using the ISRA risk analysis method [1]. The analysis resulted in a wide range of possible issues, several applicable to smart homes. Furthermore, other studies investigating the security in smart home applications and devices mainly present vulnerabilities in the systems and propose solutions on how to mitigate these vulnerabilities [9, 19, 22, 23]. They sometimes touch on the risks within the system, but this topic is not their primary focus. Our focus is to understand a large variety of risks to provide an extendable platform on which prevention for threats against smart homes can be implemented.

2.2 Security management in smart homes

Regarding security management in smart homes, Batalla et al. [23] published an extensive paper on smart home security challenges in which they propose a solution where network providers implement a security layer between internet service

providers (ISP) and home networks. They propose to do this via a multi-functional home gateway residing in the home of the end user, and with a home area network management system. Sivaraman et al. [22] in turn propose a similar solution with a security management provider. However, contrary to Batalla et al. they argue that this device does not necessarily need to be operated by the network provider. Their justification for the decoupling is that it opens up for more actors on the market of security solutions which allows for competition within the security field, ideally leading to more secure solutions. In our work we instead explore the viability of placing a security mechanism in a smart hub since it has additional information on the smart home state.

2.3 Intrusion detection in IoT devices

Multiple works on intrusion detection for IoT exists. However, the topic is still in its incipiency and many problems need to be solved before the technology can be considered mature [24]. Many of these works on intrusion detection focus on distributed solutions in sensor networks, where each node in the network contains an instance of or part of an IDS. Drawbacks of a distributed IDS in the smart home environment include reluctance from manufacturers to agree on a common system, and the heterogeneous nature of smart homes comprising several devices using different technologies. Our solution instead investigates a centralized system for detecting security breaches.

Sforzin et al. [25] developed a project called RPiDS in which they provided a portable Raspberry Pi 2, a small single-board computer, running the well-known general purpose intrusion detection system (IDS) Snort. The authors highlighted two main features. Firstly, due to its portability and ease of use deploying the system on any desired location would be easy. Secondly, the architecture allowed multiple Raspberry Pis to work together to increase detection rates and reduce false positives. However, they found that a single Raspberry Pi had issues with high network loads.

Kyaw et al. [26] made a comparative study between Snort and Bro, another general purpose IDS. The study, run on a Raspberry Pi 2, focused on the performance impact of each IDS in terms of processor (CPU) usage, random access memory (RAM) usage and packet loss rate (PLR) amongst others. Snort used around 25% CPU and 23% RAM whilst the same measurements for Bro was 80% CPU and 6% RAM. Noticeably, PLR was as high as 4% and 19% for the two systems, respectively. The authors argued this indicated that too high network loads might crash the system.

Our research differs from these studies in how our focus is on the smart hub and how to use the data it provides. Using intelligent filters and data from the smart hub, the amount of network traffic necessary to examine can be minimized. Furthermore, smart hub data could help create more situationally adapted detection filters than otherwise possible, leading to increased efficiency and decreased CPU/RAM load.

3

Definition of the smart home

In this chapter, we present relevant background theory about smart homes. In section 3.1 we explore a high-level definition of smart homes, expected devices and the restrictions these devices generally have. Section 3.2 presents common architectures and Section 3.3 explores common communication protocols and technologies. Finally, Section 3.4 gives an overview of the most likely user of smart homes and Section 3.5 summarizes the information that has been presented.

3.1 Properties of the smart home

The idea of smart homes has existed for at least 70 years [27] and has been defined by different authors multiple times ever since [28, 29, 30, 31]. However, three aspects are almost always present among the definitions from the past 20 years. Firstly, the devices in the home need to be connected, primarily to each other, but also to the internet. Secondly, there needs to be an intelligent way to control the system, such as a central gateway or intelligent smartphone apps. Finally, there needs to be some degree of home automation within the system.

There are mainly three different types of devices present in smart homes - sensors, actuators and mixed devices. Sensors, e.g. thermometers, light sensors or button switches, provide information of the real world environment into the smart home network. Actuators, e.g. light bulbs, smart locks or coffee machines, act upon this environment information and perform actions according to some preset automation rules or manual instructions. Finally, mixed devices are more powerful devices with both sensors and actuators, such as entertainment systems or surveillance systems. In addition to this, most types of smart homes has some central gateway that connects the home and enables devices to communicate with each other. Personal computers and smartphones are not generally considered to be smart home devices, even though they may interact with other smart home devices.

3.1.1 Resource restriction

Smart home devices often have restrictions in their resources. Some devices may be battery operated, giving them a restriction on their power supply. Other devices have restrictions in CPU, memory or bandwidth. These restrictions give way to challenges that are not present for less resource-restricted devices such as workstations and laptops.

The main challenges can be divided into two categories: communication issues and computational issues. A common communication challenge is that battery operated devices may periodically go offline in order to save power. Furthermore, there might be a need to combine data, since the data is transmitted less often. As these changes affect the prerequisites for communication, it also creates a need for new or updated communication protocols where transmission only happens during specific periods of time with concatenated packets.

Computational challenges are affected both by CPU- and memory restrictions. Low CPU resources may make traditional solutions for computation and security impossible to implement [14]. For example, some cryptographic functions need a substantial amount of CPU resources to be fast enough to be practical. As such these are currently not an alternative for CPU restricted devices. However, there exist projects to find low demanding cryptographic algorithms for resource restricted environments [32].

3.2 Architectures and setups

There are several ways to set up smart homes in order to provide an interface for the end user. This section presents four common architectures used for smart homes and highlights their main advantages and disadvantages.

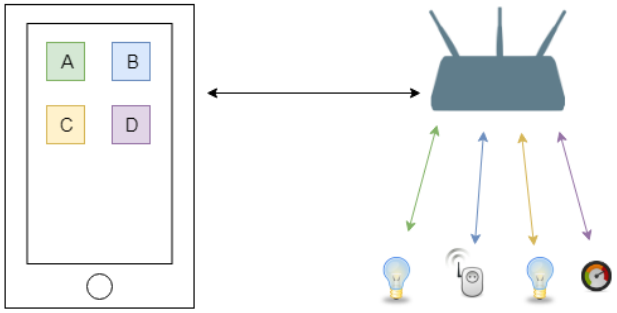


Figure 3.1: A WiFi based architecture. The user needs multiple apps for the smart home.

WiFi enabled consumer devices are often provided with an accompanying smartphone app to control them. The apps are usually focused around being easy to use

in order to be accessible even for non-technical users. Smart devices from the same manufacturer are often compatible with each other and can be controlled from the same application. However, if users want to have devices from different manufacturers, they may end up with many apps to control all their different devices. This easily becomes overwhelming when the number of devices grow larger as can be seen in Figure 3.1.

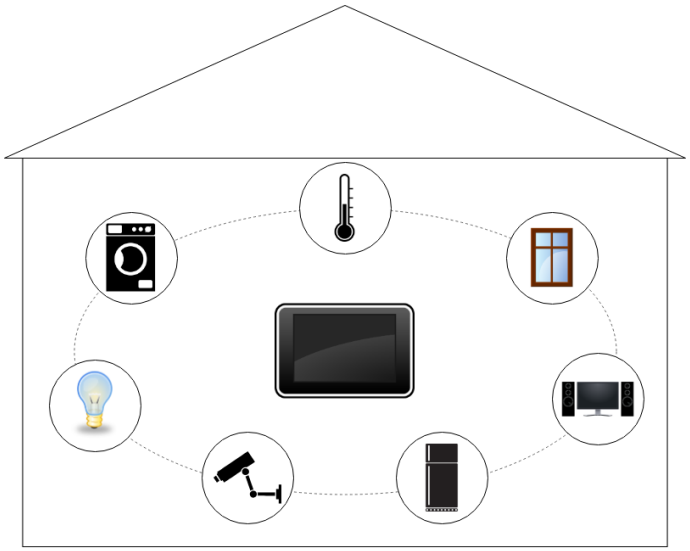


Figure 3.2: An integrated architecture such as the one in Jacobsson et al. [1].

In modern houses there are sometimes integrated smart home systems built into the building itself. Such a system could for example include components for heating, ventilation and air-conditioning systems. In these cases the user interface is often composed of physical devices located in the building, e.g. on a dedicated touch screen or with hardware buttons and sensors. The control panels communicate with a central entity within the building which carries out the requested changes. These types of smart homes or building has an efficient structure and are usually easy to use, however they can rarely be extended. Figure 3.2 illustrates a smart home with this architecture.

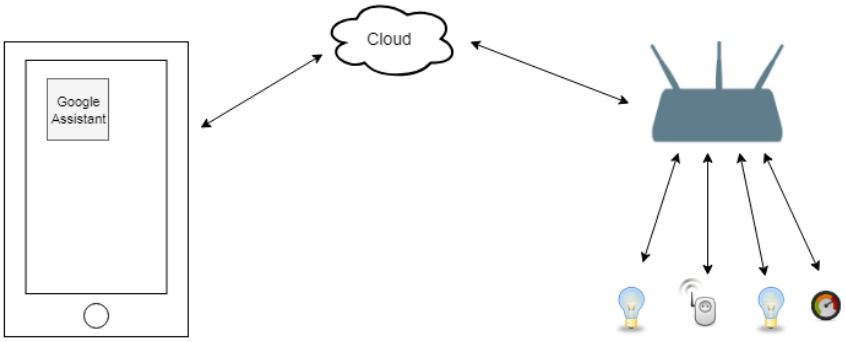


Figure 3.3: A cloud based architecture such as Google Assistant.

Some players on the market, such as Amazon and Google, have cloud based support

for smart homes. In these set-ups they provide an interface, such as Amazon Alexa or Google Assistant, and allow the user to connect supported devices to the interface. This allows for extendability of the home, with cross-compatibility between different vendors, and a single interface for all the devices. In most of these cases the end-user can use one smartphone app to control all the smart home devices. An example of a smart home cloud architecture is depicted in Figure 3.3.

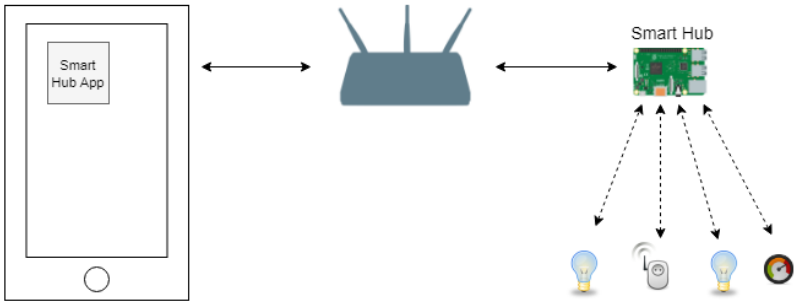


Figure 3.4: A hub-based architecture, such as Samsung SmartThings and HomeAssistant.

The final architecture we present is a hub-based system, such as Samsung SmartThings hub, or the Home Assistant hub. A hub is a central unit that, like the cloud supported architecture, connects supported devices to one single interface. A strength compared to the cloud based system is that the devices do not necessarily need to communicate over the internet. However, since it requires a physical hub to be installed and configured, this is more expensive and might be time consuming for the end user.

3.3 Communication protocols

The IoT has brought with it many new communication technologies, the development of which has often been driven by the challenges caused by resource restrictions and new use cases. As an example, many smart home devices broadcast their wireless traffic on 433MHz or 900MHz since a lower frequency penetrates walls better. Some technologies used in smart homes also utilize multi-hop routing in order to reach even further with the help of other nodes. This section brings up some of the most common protocols and communication technologies within smart home devices and compares it to more traditional technology.

3.3.1 Common network protocols

In traditional networking and the Internet at large the layered structure of the OSI model are mostly used [33]. Layers 3 and 4 are dominated by the TCP/IP

communication stack, whilst either Ethernet or IEEE 802.11 (WiFi) usually inhabits the two lowest layers. In layer 5 and 6 we find important encryption protocols such as SSL and TSL. Layer 7, the application layer, contains a large variety of protocols including well-known ones such as HTTP, domain name system (DNS) and SMTP. Table 3.1 presents common protocols in terms of the OSI layers [3].

Table 3.1: Common protocols in traditional technology in terms of the OSI model [3].

Layer	Name	Protocols
1	Physical layer	Ethernet, IEEE 802.11, USB, Bluetooth, DSL
2	Data link layer	Ethernet, IEEE 802.11, PPP, ARP, NDP, USB
3	Network layer	IPv4, IPv6, ICMP, OSPF, RIP, NAT
4	Transport layer	TCP, UDP
5	Session layer	SSL, TSL, RPC, SMB, SMPP
6	Presentation layer	SSL, TSL
7	Application layer	HTTP, DNS, SMTP, POP3, UPnP, SOAP

3.3.2 Network protocols in the smart some

In order to be easy to set up and use, smart home devices are almost always wireless and hence they almost exclusively use network protocols designed for wireless transfers. Some of the most common network protocols used are described in this section.

WiFi is frequently used in smart home devices since it makes the devices compatible with smartphones and other already existing equipment, which in turn makes it easy for the user to get started without any additional products. The introduction of WiFi mesh networks means better coverage which in turns enables smart home devices by weakening the demand on the power of their radios.

Bluetooth is another traditional technology which has made its way into the smart home. Many users are already used to using Bluetooth headphones, speakers and keyboards with smartphones and tablets. All this makes it easy for the user to incorporate new devices that use the same technology. There also exists a more energy efficient version of Bluetooth called Bluetooth Low Energy (BLE) which makes battery driven devices last longer.

Z-Wave is a relatively young technology which broadcasts on frequencies around 900MHz. It is specifically designed to be used in smart homes and is therefore energy efficient, uses multi-hop routing and focuses on low latency and reliability instead of maximizing data throughput. Z-Wave is a proprietary protocol owned by the Z-Wave Alliance, which means little open research has been done on its security aspects. However, in 2016 a mandatory security standard, called the S2 standard, was issued by the Z-Wave Alliance. All products incorporating Z-Wave must follow this standard to receive a Z-Wave certificate.

Zigbee, on the other hand, is a non-proprietary protocol developed by the Zigbee Alliance. It broadcasts either on 2.4GHz or 900MHz and is based on the IEEE

802.15.4 standard. As with Z-Wave, it is energy efficient with multi-hop routing and relatively low transmission rates. Aside from this, the hardware is designed to be as cheap as possible in order to enable product owners to lower their costs and bring consumers cheaper technology. Over the history of Zigbee there have been multiple reports of breaches and security issues [34, 35, 36] but these have been resolved and nowadays Zigbee has grown into a more secure protocol [37].

3.3.3 Application protocols in the smart home

CoAP, defined in RFC 7252, is a web transfer protocol on top of UDP which is designed for constrained devices and low-power, lossy networks [38]. It is a machine-to-machine protocol suitable for smart energy systems and building automation [38]. CoAP is similar to the HTTP protocol, which it is designed to integrate with, in that it provides a request/response interaction model. However, it also provides multicast support, very low overhead and simplicity [38].

MQTT is a messaging protocol based on the publish/subscribe pattern, a pattern which is very suitable for a highly generic environment such as the smart home. The protocol requires a central server, called message broker, through which all messages are routed. One key feature of MQTT is its low network overhead which enables resource-restricted devices such as low power sensors and actuators to use it [39].

3.4 The user

With the exception of smart home technology for elderly or disabled people, there are few distinguishing characteristics of the smart home user to be found in research [40]. Instead, what can be found are inferred or assumed characteristics of prospective users [40]. To find out who the smart home user is one has to look at marketing surveys and statistics instead. In 2000, Pragnell et al. [41] identified the people interested in smart home technologies as young people under 35 who were used to multimedia devices and internet in their homes that also had a relatively high income. This corresponds to current statistics, that shows that the most likely user is male, between 25 and 34 years of age with a high income [42]. The spread of smart home technology is still low, with USA topping the list with a penetration rate of 32%; worldwide this number is only 7.5% [42].

The most distinctive user group of smart home technology in research are elderly or disabled people [40]. There is much enthusiasm on how smart home technology can help elderly and disabled people obtain a safer and more independent life. To date, the diffusion of smart home technology among elderly or disabled people have been very low. While many users have been happy with the technology they have used, especially with such things as communicating with their physicians, there is a lot of fear and reluctance towards the new technology from a large group of the target

audience [43]. Most of the fear stems from distrust towards the technology, both in the functionality, but also of hidden agendas from corporations or institutions in society [43, 44].

Market research has found that smart home services need to have high availability and to be easy to use in order to be attractive to the user [41, 31]. The research presented a strong connection between users not being able to carry out the task they originally set out to do and a decreasing trust for the system. User interest in a system corresponds directly with the suers trust for said system [31].

An interesting aspect in research concerning smart home users' is that there is a disconnect between what technical researchers see as the users' needs compared to what the social science researcher see as the users' needs [40]. In fact, this divide is so big, that while the technical side pushes the vision that smart home technology is the next revolution in electrification, some social science researchers question what kind of needs smart home technology actually fulfills [40]. In product development there have often been very little contact between the actual potential users and the developers [45], and the same pattern can be observed in research about smart homes. This has created research that is often not based on the true needs of the users, but rather a push for the technology developed [46]. Wilson et al. [40] argue that the lack of focus on who the user is and what they want is a contributing factor to the relative slow uptake of smart home technology that has been seen historically.

3.5 Summary

This capter presented the main properties of smart homes. The smart home consists of connected devices with a central intelligent gateway with ability to control the devices and to automate behaviour. Devices are generally resource restricted concerning CPU and/or RAM, and can be categorized into either sensors, actuators or both.

Multiple architectures were presented: WiFi-based architectures, integrated architectures, cloud based architectures and hub-based architectures. These differ in terms of device requirements, extendability and how the devices in the network are controlled. Even though the WiFi-based architecture was very popular in the dawn of smart homes, cloud based and hub-based architectures are increasing in interest.

Multiple protocols and communication technologies have been created to fit the need of smart homes. These include Zigbee, Z-Wave, CoAP and MQTT. Traditional technologies such as WiFi and Bluetooth are also used frequently in smart homes.

The chapter was concluded with the characteristics of a general smart home user. Even though research claims that elderly and disabled people have much to gain from smart homes, the technology has mostly been met with fear and reluctance. Instead young males with a high income is currently the most frequent user of smart

homes. Necessary features for a smart home product to succeed among users include high availability and ease of use.

4

Security in smart homes

This chapter brings up the subject of security in smart homes. Section 4.1 introduces the most urgent risks within smart homes. In Section 4.2 we present proof-of-concept and real-world attacks in order to concretize threats and vulnerabilities. Section 4.3 presents traditional ways of handling security, while Section 4.4 explains why these traditional methods are not optimal for a smart home setting and proposes a different set of techniques and approaches to secure smart home devices.

4.1 Security risks

In the literature for security in smart homes the terms risk, threat and vulnerability are often used. Sometimes they are used interchangeably, and it can be difficult to pinpoint what the words mean at a given moment. In this thesis report, we are using the following definitions as defined in [47]:

- A threat is anything that has the potential to damage a system.
- A vulnerability is an existing weakness in a system, which can be exploited.
- A risk is the probability of a threat, coupled with the cost of the consequence of said threat.

Jacobsson et al. [1] published an extensive risk analysis on IoT, where many of the found risks fit into the profile of smart homes as well. We have highlighted some of the most pressing security risks for smart homes from this report, namely those of a mean risk value higher than 8, and present them in more detail in Table 4.1. In summary, these risks fall into two categories, inadequate security mechanisms in all levels of the system, and bad behaviour from users. These risks are often brought up as pressing risks in the field.

The Open Web Application Security Project (OWASP) is a well-known foundation which have compiled a list of the top 10 vulnerabilities for IoT devices in 2014 [4]. Even though the fields of IoT and smart homes do not entirely overlap, the main security issues of IoT can be applied to a smart home setting as well. The top five

Table 4.1: Risks of a mean risk value greater than 8, as identified by Jacobsson et al. [1].

Description of vulnerability	Consequence
Threats originating in Software Components	
Inadequate authentication in the gateway	Unauthorized access to the system
Inadequate accountability in the gateway	Unregistered system events
Inadequate software security in applications	Unauthorized modification of functions
Software vulnerability in API	Unauthorized modification of functions
Threats originating from the information/data process in the system	
Inadequate authentication and access control in the gateway	Manipulation, duplication, surveillance, and deletion
Inadequate access control policy and configuration in the gateway	Inadequate authentication and access control
Threats originating in the utilized communication channels	
Inadequate authentication and confidentiality in sensors	Manipulation, duplication, surveillance, and deletion
Inadequate authentication and confidentiality in cloud services	Manipulation, duplication, surveillance, and deletion
Threats originating with the end users	
Sloppy end-users, etc.	Social engineering
Gullible users, etc.	Privacy threats
Poor password selection by users	Circumvention of authentication mechanisms
Bad systems configuration by users	Hacking exploration attacks

vulnerabilities reported by OWASP are presented in Table 4.2.

Insecure control interfaces, whether it is a web-based architecture or a smartphone app, are important attack surfaces to secure. If an attacker is able to gain access to said interface, they can operate the device to send and retrieve data as the original user. Furthermore, they might be able to reconfigure the device entirely, detaching the original owners and leaving them with an uncontrollable device [36]. Hence, securing both virtual and physical access to the device is of great importance.

Denning et al. [5] published a study which not only brings up risks with smart homes but also the end consequences of a possible breach. Their reasoning describes the process in three steps:

- *Low-level mechanisms* are exploitable security risks in the smart home.
  Example: Hack a smart lock.
- *Intermediary goals* are the action(s) the attacker is trying to accomplish.
  Example: Enable physical entry to the apartment. Viewing private data.
- *High-level goals* are the end-goal for the attacker. This step provides the actual benefit of the entire process for the attacker.
  Example: Blackmailing.

They argue that by recognizing possible end goals for attackers, new risks for smart

Table 4.2: OWASP Top 5 IoT vulnerabilities [4].

Rank	Vulnerability	Examples of issues	Impact
1	Insecure Web Interface	Use of default passwords No lockout on multiple failed logins Valid usernames can be determined	Severe
2	Insufficient Authentication Insufficient Authorization	Insecure passwords are allowed Credentials transmitted in clear text No separation in levels of privilege	Severe
3	Insecure Network Services	Unused ports are open Services vulnerable to buffer overflow Abnormal traffic not filtered out	Moderate
4	Lack of Transport Encryption Integrity Verification	Use of unencrypted protocols Encryption protocols are out of date Packet integrity is not checked	Severe
5	Privacy Concerns	Arbitrary data is collected Personal data not properly secured Not using a data retention policy	Severe

homes can be discovered that were not previously considered. The study published a thorough list with examples of low-level mechanisms, intermediate goals and high-level goals [5]. This list is presented in a compact format in Table 4.3

4.2 Attacks on smart homes

There have been a number of attacks against devices that can be found in smart homes. Some have been documented in research as proof-of-concept attacks in an attempt to motivate further development of security countermeasures, while others are real-life breaches which can mostly be found in online articles, on news-sites, and blogs. Some of the most recent academic papers on attacks against smart home devices are presented in Table 4.4.

4.2.1 Proof-of-concept attacks in research

Ling et al. [48] published a paper in 2017 on the Edimax SP-2101W smart plug in which they are able to retrieve full control of the smart plug with little or no access to the network it is residing in. Even if the end-user tries to use a password to protect the device, the infrastructure of the system allows an attacker to stealthily acquire this password in multiple ways. Furthermore, the authors show the possibility to upload malicious firmware to the device in order to gain total control of it.

Hernandez et al. [49] evaluated security in the Nest Thermostat and found lacking hardware protection. They showed that an attacker with physical access can overwrite the firmware, using only a standard USB-cable, to gain root access on the

Category	Examples		
Low-level Mechanism	Altering logs	Altering/destroying data	
	DoS attacks	Viewing/Altering traffic	
	Viewing sensors	Viewing data	
	Using actuators		
Intermediate Goals	Accessing financial data	Causing environment damage	
	Causing device damage	Causing physical harm	
	Enabling physical entry	Gathering incriminating data	
	Misinformation	Planting fake evidence	
	Viewing private data		
High-level Goals	Physical Theft	Blackmail	Espionage
	Resource Theft	Exposure	Extortion
	Kidnapping	Framing	Fraud
	Stalking	Terrorism	Vandalism
	Voyeurism		

Table 4.3: An overview of intermediary goals with an attack towards a smart home, as defined by Denning et al. [5].

device. This technique only required 15 seconds alone with the device to perform the attack.

In a paper by Lei et al. [50], the security properties of the voice assistant Alexa is studied. The focus of the authors was specifically put on protection from acoustic attacks, where the attacker plays a sound and tricks the voice assistant into believing a human is issuing the command, e.g. purchasing products online. The authors developed two proof-of-concept attacks which show that Alexa is susceptible to these kinds of attacks.

Sivaraman et al. [22] explored security in a range of different smart home devices, Philips Hue light bulb; Nest smoke alarm; Withings smart baby monitor; Withings smart body analyzer and Belkin Wemo motion sensor & switch kit, and found flaws or potential weaknesses in most of the devices. Four out of the five devices were susceptible to attacks resulting in giving the attacker the ability to masquerade as a legitimate user in order to send or retrieve data to control the device. Furthermore, in the case of the Belkin Wemo switch the attacker was able to activate remote controlling of the device.

Tang et al. [51] explored the Evil-Twin attack, where an adversary deploys a compromised access point (AP) close to the victim's home in order to trick smart home devices into connecting to the compromised AP instead of the real one. This attack works because devices readily connect to any AP that looks like theirs, even if it in actuality is an AP masquerading as the one the device usually connects to. Once a device has connected to the attackers AP, the adversary can gain full control over it. Tang et al. [51] developed an approach to detect the existence of such a fake

Table 4.4: Summary of proof-of-concept papers.

Paper	Devices	Attack type	Countermeasure
Ling [48]	Edimax SP-2101W	Evesdropping Device scanning Brute force Device spoofing Firmware	Various
Hernandez [49]	Nest Thermostat	Physical, firmware	Hardware modification and physical security
Lei [50]	Amazon Alexa	Auditorial	Presence confirmation by WiFi motion detection
Sivaraman [22]	Belkin WeMo Nest smoke-alarm Philips Hue Withings baby-monitor Withings body-analyzer	Various privacy issues Masquerading Evesdropping Remote access	Security management provider
Tang [51]	Various	Evil-Twin masquerading	RSSI positioning verification
Fernandes [9]	Samsung SmartThings	Abuse privilege model	More fine-grained permission policies

AP by using the received signal strength indicator (RSSI), as opposed to traditional approaches which focus on properties such as MAC, SSID or network patterns - which are possible to forge.

Fernandes et al. [9] performed an analysis of Samsung SmartThings, a smart home platform consisting of a gateway, device apps to support a wide range of different devices and support for third-party app development. The article included an in-depth study of nearly 500 device apps and discovered two general flaws. Firstly, device apps were able to access privileges not explicitly given to them due to an over-privilege bug. For example, if an app were allowed to read the status of a smart lock it was automatically also allowed to perform all other actions on the lock, including locking and unlocking it. Secondly, the internal communication were not sufficiently secured, which could expose keys and other confidential material to an attacker. Based on these flaws Fernandes et al. [9] were able to perform four proof-of-concept attacks to steal door codes, implant new door codes, induce a fake fire alarm and deactivate vacation mode in a building.

4.2.2 Real-world attacks on smart homes

One attack type that can especially benefit from the increased number of devices due to the growth of smart homes is distributed denial of service (DDoS) attacks. This was clearly shown by the Mirai worm, a worm that has affected hundreds of thousands of IoT devices, including many found in smart homes [52]. Mirai was discovered in 2016 and has been the cause of several big DDoS attacks. One of these attacks had a traffic peak at 1.1Tbps, magnitudes above what most sites on the

internet can handle [52]. Mirai exploits naive vulnerabilities in IoT devices, mainly the use of common default passwords, but has the potential to severely impact the infrastructure of the internet. This was shown by the October 2016 attack against the service provider Dyn which caused several large sites on the internet to become unavailable [52].

There have been multiple news reports concerning a few minor single-target attacks as well [11, 12, 13]. In 2014 a man hacked a connected baby monitor and screamed "Wake up baby" multiple times at a sleeping baby [11]. In 2016, hackers attacked the thermostats in two buildings in Finland which resulted in non-working heating in the facilities [13]. At the time the outside temperature was around $-5°C$. One attack in 2018, which went rather viral, concerned not an attack against a private home but against a casino [12]. The attackers managed to get their hands on the high-roller database by entering the network through a smart fish tank thermostat.

4.3 Traditional security measures

Traditional network security consist of several techniques which are often bundled together to provide a secure environment. This section brings up a few of the most common techniques.

One of the most used techniques is encryption. Modern protocols, in multiple stages of data transfer, use public-key cryptography and intricate hand-shaking protocols to set up secure connections. Encryption is often seen as a base premise for confidentiality.

Authorization is the process of deciding whether a certain entity is allowed to perform a certain action. This is a crucial concept in computer security as it validates that only legitimate users and services can access protected data. A prerequisite for authorization is authentication. Traditionally, authentication can be done in various ways: passwords, passcodes, biometric validations and Near Field Communication (NFC) amongst other. Two-factor-authentication (2FA), where users need to authenticate themselves in two different ways, is becoming increasingly popular [53].

One of the oldest protective security measures is firewalls. Nowadays most homes are equipped with a home WiFi router in order to supply a wireless internet connection for portable devices such as smartphones, tablets and laptops. These routers usually include a basic firewall. Modern operating systems for computers, such as Microsoft Windows, Mac OS and various Linux distributions, provide a basic firewall by default as well.

The existence of zero-day exploits means that not all attacks can be prevented from entering a system. However, it is important to detect when such a breach has occurred. Traditionally, intrusion detection systems and intrusion prevention systems (IPS) have been used to fill this task. IDSs only monitor traffic passively

while an IPS also has the possibility to take action against the threat.

Another way to improve the chance that attacks do not go undetected is extensive logging of all activity on the device. These logs help system administrators analyze system actions in order to find anomalies, understand the passage of events, and use this data to improve the security.

Penetration testing is often used by companies to make sure their product or office network is impregnable. These tests are run by security experts and quality of the outcome relies heavily on the skill and experience of the testers.

Common Criteria (CC) is a certificate standard which reassures that a product has undergone significant security review by an expert third-party certifier. It is an international recognized standard which assesses security functions, performs penetration tests and evaluates the development process of the product to assure security compliance. Certificates need to be re-issued if any hardware of software changes occur.

4.4 Smart home security measures

This section brings up issues with traditional security measures based on the environment of smart homes and the current risk situation. Furthermore, it presents a set of security techniques and approaches adapted to a smart home setting.

4.4.1 Issues with traditional security measures

One general issue with traditional security in smart homes is the expectations from the user. In a research study by Sandström [31] about smart home users, he found that users are less willing to use a smart home device if it is harder to access. This means that if security measures of a smart home make it noticeably more difficult or more time-consuming for a user to use services of the smart home, the user are less likely to use them as much. As a consequence, it is important to provide security solutions which does not affect the user productivity in the system. As an example, new ways to authenticate a user might be needed to ease the process whilst still keeping high security.

The non-technical nature of the user [31], or rather the lack of knowledgeable system administrators, renders some security countermeasures unusable. This includes logging, where the user generally neither has the knowledge nor the interest to assess device actions via data logs. Furthermore, evaluation of penetration tests would not be feasible either.

In addition to the above issues, the heterogeneous and resource restrained nature of smart homes can make it difficult or impossible to apply traditional security

measures. Hossain et al. [14] pinpointed a number of important constraints and how they make applying traditional security measures difficult. The limitations on resources in smart home devices mean that computationally heavy cryptographic algorithms are not applicable, and security solutions that have not specifically been designed with memory and storage limitations in mind may not work [14]. For example, local firewalls or IDSs on smart home devices may not be possible due to lack of processing power.

The heterogeneity of smart home networks also come with problems. Not only do smart home networks contain a lot of devices, making scalability of the security solution an issue, these devices may also be mobile, joining and leaving the network at any time, something that existing security models may not be able to cope with. In addition, the heterogeneity when it comes to communication protocols can also leave traditional security measures lacking. Hossain et al. [14] also point out that the slimmed, embedded operating systems on IoT devices often have a thin network protocol stack that might lack the security measures of traditional systems. In addition, the OS or protocol stack on the device may not be able to integrate new code, making it very difficult to patch discovered vulnerabilities.

4.4.2 New proposed techniques and approaches

Even though the field of security in smart homes is relatively young, some new techniques and approaches have been proposed. This section brings up a few of them as a contrast and extension to the section on traditional approaches. All approaches either address the problem of resource restriction or the lack of technical experience among users.

One of the most used encryption layers, transport layer security (TLS), operates on top of TCP in the TCP/IP stack. However, a similar protocol called datagram transport layer security (DTLS) has been developed to provide support for encryption over UDP and can therefore be used with the resource efficient protocol CoAP [54, 55]. This enables some nodes to introduce encryption into data transmissions, leading to higher security. However, the devices which are most resource restricted will still be unable to use this feature due to the complexity of the encryption algorithms themselves.

There is ongoing work to secure MQTT, a frequently used protocol within smart homes. In a study from 2015 researchers used a lightweight attribute based encryption (ABE) algorithm to establish secure communications in an MQTT broadcast domain and specified the protocol SMQTT [56]. The technique was also adapted for MQTT-SN, MQTT for sensor nodes, in order to provide support for power constrained devices.

Several new ways to authenticate users have been explored. Voice authentication has been a concept for a long time and patents dates back all the way to 2000 [57]. However, this technique was not commonly used until a few years ago and nowadays

companies like Nuance [58], VoicePin [59] and Google [60] offer this service on a large scale. Zhang et al. [61] extended this idea to involve both voice- and gesture authentication, in order to prevent an attacker from carrying out a replay attack by using a pre-recorded audio clip. Lei et al. [50] suggested that voice features only should be available if that particular user is present at home.

IDSs can not stop vulnerabilities from being exploited but detection of an attack can provide the end-user with information concerning potential breaches of particular devices in their smart home setup. This information could help the users, even those with little to no technological knowledge, to take action by either sending the device to a service shop for repairs or by replacing it. When an infected device only gets a small time window to transfer the virus which infected it, there is less chance it will succeed and hence virus spread will be mitigated in that particular network. The work in this thesis is focused on this approach.

4.5 Summary

This chapter presented security aspects within a smart home. First, the result of multiple risk analyses were listed. Risks are generated both from hardware and software components as well as from the user itself and can result in loss of privacy, invalid data transmission or a device being controlled by a malicious user. A compromised smart home device can also be used as a stepping-stone to perform other crimes, such as physical theft or blackmailing.

A few historical attacks on smart homes were brought up. These included both proof-of-concept attacks, where researchers find security holes in products, and reports about actual real-life breaches. The botnet Mirai was mentioned as an alarming consequence of inadequate smart home security.

The chapter continued with explanations of traditional security concepts such as encryption, authorization, firewalls, IDS, IPS, penetration testing and logging. We also presented issues with applying these techniques to smart homes, the two main issues are the resource-restricted property of devices and the lack of technical knowledge among users. Finally, a new set of techniques and approaches were presented which take these shortcomings into consideration.

5

Development of a prototype Security Supervisor

In this chapter we present the development process of a security module for detecting malicious activity in a smart home network. The module will be referred to as the *Security Supervisor* and is integrated with the popular open source smart hub platform Home Assistant. Section 5.1 specifies the design of the Security Supervisor and how it interacts with Home Assistant on an abstract level whilst Section 5.2 provides details on the actual implementation.

5.1 Design

As previously stated, we use a smart hub as the location for the Security Supervisor. We compared several smart hubs and chose Home Assistant due to its popularity, availability, and openness. Thus we designed our Security Supervisor to fit into the Home Assistant architecture even though the general principles of the software are adaptable to any smart hub system. A key design feature was that the Security Supervisor should be easily extendable.

5.1.1 Architecture

This section describes the architecture of both Home Assistant and the Security Supervisor and shows how they interface with one another. The Security Supervisor is designed to be able to integrate with any smart hub but the implementation described in this thesis interfaces with Home Assistant.

5.1.1.1 Architecture of Home Assistant

Home Assistant has a straightforward and modular software architecture [62]. It consists of a central core, a user interface, home automation, and components to communicate with smart devices in the home. In Figure 5.1 the shaded parts are

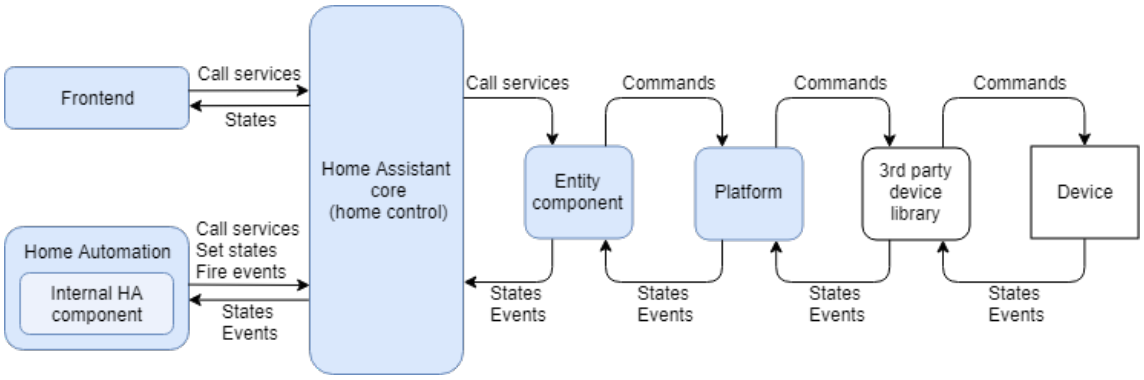


Figure 5.1: Architecture of Home Assistant [2].

part of the Home Assistant source code while the non-shaded parts are external libraries and physical devices.

The core of Home Assistant consists of an EventBus and a StateMachine. The EventBus allows other parts of Home Assistant to fire and listen to events in order to communicate changes to each other. The StateMachine holds the current state of the smart home, and changes to the states are facilitated through the events sent over the EventBus.

Communication with smart home devices is handled through entity components and platforms. The entity components are components which handle communication of a type of device, i.e. lights or switches. They contain shared functionality common to a type of device. The platforms expand the entity components so that they are compatible with certain brands of devices. The actual communication with the device is in the shape of external 3rd party libraries. The platforms communicate commands, states and events with these libraries through API calls. The modularity allowed by the entity components and platforms makes it easy for developers to add support to more devices.

The final part of Home Assistant is the home automation. This is controlled by user configurations and internal components that use triggers from events together with information from the core to activate commands. An example of an automation would be to turn on a light when the user comes home and it is dark outside.

5.1.1.2 Architecture of Security Supervisor

The threat detection is added to Home Assistant as an internal component. As a component it has access to the event bus and internal knowledge of the hub, allowing it to factor details of the smart home setup into its threat detection mechanisms.

The threat detection component in itself has a layered and modular structure with three main layers: collection and pre-processing, threat analysis, and data presentation. These layers are depicted in Figure 5.2 and are explained below. We focus

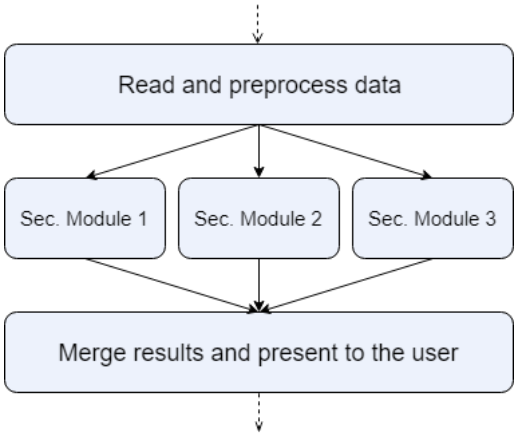


Figure 5.2: The three layered architecture of the Security Supervisor.

on layer one and two in this thesis.

The first layer handles collection and pre-processing of data. This is not only network data but also information about states of devices in the smart home. The data is processed and categorized in order to match filters provided from analysis modules in layer two. All data which matches a module filter is passed on to that specific module for further processing. The first layer also provides utility functionality for profiling.

The second layer is populated by security modules. Each module focuses on detecting a particular threat and provides, as previously mentioned, a filter to match to general characteristics of this threat. Once data enters the module, an internal algorithm evaluates the data on a deeper level and alarms layer three if any malicious activity is found.

The third layer acts as an aggregator and summarizes all information from security modules in layer two in order to present the data to the end-user. Even though the design of this layer might affect the outcome of the end product, since users prioritize easy-to-use tools, this layer has not been the focus of the thesis and will not be presented further.

The layered structure of the Security Supervisor not only fits in with the general layered structure of Home Assistant, it also allows for easy extension of the Security Supervisor. With the fast pace of the market and the always evolving cyber-attacks, easy extendability is a key feature for software which focuses on detecting security breaches.

5.1.2 Layer 1: Data collection and preprocessing

The main purpose of layer one is to act as a utility layer and provide easy and fast access to data gathered from the device for the analysis modules in layer two. The

layer itself can be divided into three separate parts: reading input data, communicating with the smart hub and profiling. These parts will be explained in detail below.

In order to scan for threats, the Security Supervisor needs to be able to intercept network traffic and process it. Since, as mentioned in Section 3.3, smart homes consist of devices with multiple communication protocols it is important to provide a generic interface for input data. This interface component is responsible to gather data from multiple sources and merge it into one single data stream, a process which is depicted in Figure 5.3. The concatenated data from the data merger will be profiled and processed by each analysis module for which the data match the announced filter of corresponding analysis module.

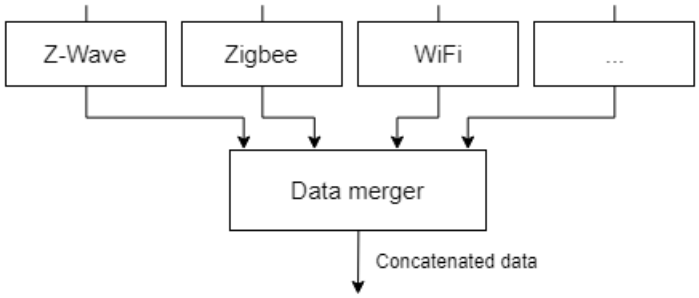


Figure 5.3: The traffic concatenation module provides a generic interface for multiple technologies.

One big advantage with locating threat detection within a smart hub is that it gains access to the data of the smart hub. Layer one is responsible for gathering updated information about the state of the smart home and to communicate any changes to the analysis modules. It also provides the network capture drivers with information about smart device states in order to limit the amount of network traffic which need to be processed. Instead of listening to all devices on the network, including PCs streaming movie content or performing other high network load tasks, the Security Supervisor needs only to process smart home device data. With less data to be processed, the amount of dropped packets decrease. The packet drop rate was mentioned both by Sforzin et al. [25] and Kyaw et al. [26] as a main issue with threat detection on resource restricted devices.

The Security Supervisor uses profiling as a central concept. Layer one provides a set of high-level functions to maintain profiles for each device. Once network data has been captured and pre-processed, it is run through a set of profiling functions provided by the analysis modules. By giving the responsibility of profiling functions to the analysis modules, we assure that the data needed for the modules to work is guaranteed to be profiled. However, a set of default profiling functions for common protocols, such as IPv4, TCP and UDP, are provided by layer one by default in order to reduce redundant profiling functions. The profiling process and how it affects the other layers in the Security Supervisor is depicted in Figure 5.4. Especially notice how layer two is separated into two sections to adhere to the profiling - one part to

perform the profiling and one part to run the actual analysis.

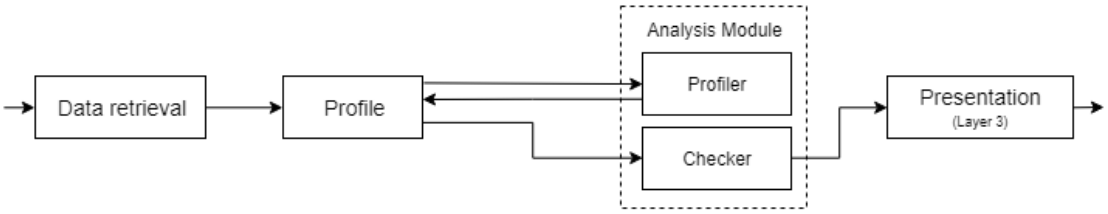


Figure 5.4: Data flows through the profiling functionality.

5.1.3 Layer 2: Analysis modules

In the Security Supervisor, the analysis modules are the parts that are responsible for detecting specific security issues. The selection of security issues to detect, and thus what security issues to solve was based on the properties of risks in Section 4.1 and attacks presented in Section 4.2. The leading properties for the selection of threats include the probability of the attack, the severity of the consequences, and the feasibility of the attack.

The first analysis module analyses outgoing traffic to detect if any device in the network has been compromised and joined a malicious botnet. Botnets can have a potentially devastating effect on both large internet infrastructures and society. With the number of devices predicted to be available in smart homes, the potential pool of bots in a botnet is magnitudes larger than what have been possible historically. Thus, it is very important to detect if any device in the smart home has been compromised in this manner.

The second analysis module handles so called Evil-Twin attacks, where an attacker-controlled AP takes over the devices in the smart home. This has the potential to leak huge amounts of personal data and facilitates easy access to the smart home or network in question. The module uses the RSSI as described by Tang et al. [51] to evaluate if an AP is an evil twin or not, as this is a metric that is harder to forge for the adversary than network protocol data.

5.2 Implementation

The implementation of the threat detection was done in Python 3.6. This is the language used in the rest of the Home Assistant project, which made it an attractive choice for the Security Supervisor. Utilities to read network data was based on open source projects written in C and C++.

5.2.1 Data collection and pre-processing

In the process of network packet collection, network data is handled in multiple stages in order to allow the use of various network protocols. Figure 5.5 illustrates the stages of the process for both encrypted and unencrypted data.

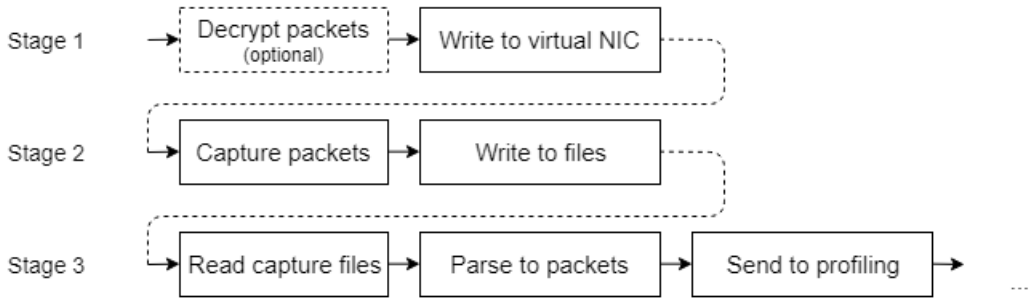


Figure 5.5: Information flow in the network traffic collection process.

The actual capturing of data is handled in the first two stages as external services outside of Home Assistant. The reason for this being done externally is twofold. Firstly, libraries for capturing and decrypting data are not efficient enough in Python. Secondly, capturing network traffic requires root privileges and since Home Assistant runs as a non-root user, it is more convenient to capture the traffic externally. These services are entirely separated as well; the decryption stage is designed to be transparent, meaning the traffic capture in the second stage will be carried out in the same way regardless of encryption or not. In order to capture all network data on Ethernet networks a network card in monitor mode is required. Other technologies may have additional hardware requirements.

Once the data is captured, it is written to files which are regularly scanned for and read by the threat detection module in order to process data. Many smart hubs use flash memories and it has been shown that an excessive amount of disk writes decreases life expectancy of the memory cells [63]. However, even though this architecture will increase the number of disk writes to permanent storage, it instead decreases the load on the RAM on the device.

Two possible libraries were considered for reading data from network files into the Security Supervisor. Scapy, a very well-known library focuses mostly on being easy to use. Packets are fully parsed from network files which allows the programmer to access pre-formatted properties by name. The properties have been formatted to an easily expected form, e.g. IPv4 addresses as strings of the form *XXX.XXX.XXX.XXX*. Pypacker is an alternative network library which instead focuses on performance and low overhead. Properties of a packet can be retrieved by name although their values are exposed as the original byte representations.

Profiling is implemented in a straightforward fashion based on the design presented in Section 5.1.2. For each smart home device, an instance of the Python class *Profile*

is created. This object keeps track of whether the device is in the initial profiling phase, where all network activity are recorded and modelled in a simplified form, or whether it should be matched against the profile in order to detect malicious activity. The *Profile* object is also the data container for the modelled device behaviour. If a smart home is extended with a new device where a modelled device behaviour already exists, this device inherits the profile. However, it does not skip the profiling phase since there might be communication between devices of the same type that would not be considered unless a full profiling phase is carried out.

Access to Home Assistant data is implemented through internal state buffers which are updated based on events from the EventBus. There exists functionality for analysis modules to subscribe to updates which are triggered when a smart home device state has changed. These are implemented as wrapper functions which pass on data from event bus events.

5.2.2 Analysis module: Botnet detection

The botnet detection analysis module is responsible for detecting if devices in the smart home are used as part of a botnet. Since DDoS attacks constructed with botnets are generally performed towards internet services, we focus on devices using WiFi technology and the TCP/IP stack. The module monitors the outgoing traffic of devices and warns the user if a device is communicating with unsuitable targets. In our implementation, unsuitable targets are all targets which the device did not communicate with during the profiling phase.

Whilst this might seem like a straightforward solution, there are some issues with name resolution in DNSs. The server IP address of a service may differ over time which in turn may result in a smart device contacting an IP address which was not encountered during the profiling phase. Consequently, unless DNS requests and responses are monitored to update the profile accordingly, erroneous warnings may be issued. The implementation solves this by not only keeping track of IP addresses, but hostnames as well. Once a DNS response is found, profiles are updated accordingly with both hostnames and IP addresses. However, trusting DNS records is not without risk, as elaborated on in Section 8.2.2.

A simplified version of the used algorithm is shown in pseudo code in Listing 5.1. The *normal* traffic is defined by fingerprinting traffic for a new device for a short period, defaulting to 24 hours, when the device is connected to the network. Theoretically, it can also be defined by the developers of the device platforms. This creates a whitelist of addresses that the device is allowed to contact. If the device contacts any other address after the training period is over, a warning will be issued.

Code Listing 5.1: Algorithm for detecting if devices are used in a botnet.

```
# Catalouge normal outgoing traffic
normal_traffic = []

if profiling
    for address in outgoing traffic
        add address to normal_traffic
else
    for address in outgoing traffic
        if address not in normal_traffic
            warn user
        else
            do nothing
```

5.2.3 Analysis module: Evil-Twin

The implementation of the Evil-Twin attack detection uses the algorithm for single device single position (SDSP) presented by Tang et al. [51]. In the profiling phase, RSSI between the smart hub and nearby APs are continuously measured and compiled to a model for each device representing the variation of received signal strength. This model is then used in order to make sure that no fake AP is trying to compromise nearby devices.

Two different models are implemented and tested for the Security Supervisor. The first one is a basic statistical model in which valid RSSI entries are to be in the interval $average \pm variation$, where average and variation are values found during profiling phase. Furthermore, a number of detections outside this interval is acceptable each minute if that number does not surpass the average number of detections outside the interval per minute during the profiling phase.

The second model used is a min-max-model. The minimum and maximum signal strength is recorded during the profiling phase and each recorded signal strength entry which is not in the interval $minimum \leq signalstrength \leq maximum$ is marked as a threat.

We retrieve RSSI of an AP by observing the RadioTap header in received packets. This header is added by most WiFi adapter drivers and includes properties such as signal strength, signal attenuation and transmission power [64]. This data is safe to use since RadioTap headers cannot be spoofed externally and we assume our WiFi adapter has not been compromised.

The inner workings of the algorithm is as follows. The Evil-Twin module keeps a list of devices and their positions relative to the hub. It regularly checks that these devices are in the positions they should be, and will warn the user about a potential attack if any device position is inconsistent with its previously recorded position. However, if RSSI measurements for all devices change, the module assume that the location of the smart hub itself has changed.

6

Experiments

In Section 6.1 we describe the testbed and what devices it comprises. The chapter continues in Section 6.2 by explaining what metrics and methods are used to evaluate whether the Security Supervisor fulfills the goals set up in Chapter 1. We finish up the chapter in Section 6.3 by presenting our conducted experiments in detail.

6.1 Testbed

Our testbed is composed out of various devices one can expect to find in a smart home. Devices have not been chosen on any specific criteria and similar devices of other brands will most likely work out of the box with the project as well. The devices which were used are presented below.

The main component of the test bed is a Raspberry Pi 3 B+ on which Home Assistant is running. The Raspberry Pi 3 B+ has a quad-core Cortex-A53 processor with a clock-rate of 1.4GHz and 1GB of RAM [65]. It supports 802.11b/g/n/ac WiFi protocols, Bluetooth 4.2 LE and has an Ethernet RJ45 port [65]. We have also supplied the Raspberry Pi 3 B+ with a 16GB memory card. We found this to be relatively comparable to other hubs on the markets such as Samsung SmartThings hub that has a 1GHz ARM Cortex-A9 processor with 512MB RAM, and 4GB FLASH memory [66].

The Raspberry Pi has been extended with a wireless USB network card, Panda PAU06, which supports monitor mode. Monitor mode makes it possible to listen to all traffic on a network, not only the one transmitted and received from or to the device itself. This allows Home Assistant to monitor all wireless communication as if it was located in the router. Even if the network is protected by encryption, such as WEP or WPA2-PSK, it is possible to decrypt this traffic with only the SSID and pre-shared key as a prerequisite. The Panda PAU06 requires more power than provided in standard USB ports and has therefore been connected to a USB hub with external power in order to not damage the Raspberry Pi [67].

Furthermore, a Broadlink SP3 smart plug has been connected and configured to the smart home. The Broadlink SP3 plug connects to the network via IEEE 802.11

b/g/n and operates on the 2.4GHz frequency [68]. It has the basic functionality that it can be turned on/off remotely, according to a schedule, or randomly. Furthermore, it includes a nightlight that can be turned on or off as well.

The Raspberry Pi and the Broadlink SP3 smart plug are connected to an Asus Dual-Band Wireless-AC1750 Gigabit Router. It communicates via network standards IEEE 802.11 a/b/g/n/ac, IEEE 802.3b, IPv4 and IPv6 [69]. It operates on both 2.4GHz and 5GHz frequencies and can cover a large home [69]. The router supports transfer speeds up to 450 Mbps over IEEE 802.11n and up to 1300 Mbps over 802.11ac [69]. This can be compared to the Panda PAU06 adapter which only supports 300Mbps [67].

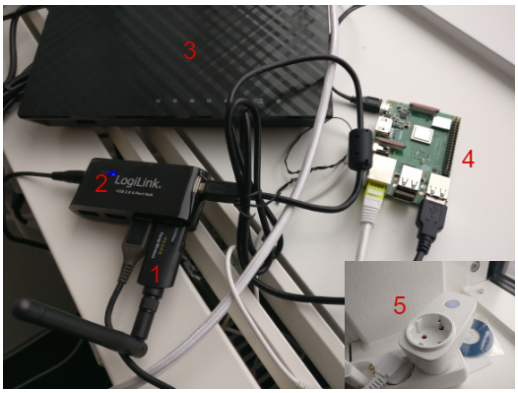


Figure 6.1: The testbed setup: 1. Panda PAU06, 2. LogiLink USB Hub, 3. ASUS AC1750, 4. Raspberry Pi 3B+, 5. Broadlink SP3 plug.

6.2 Test methodology

There are several goals the Security Supervisor needs to fulfill. To begin with, it needs to detect threats without negatively impacting the performance of the hub. In Section 6.2.1 we go into detail about what constitutes as negatively impacting the performance of the hub and what tools and techniques we used to measure these properties. Furthermore, the Security Supervisor needs to detect threats with a high accuracy and a low rate of false positives. Section 5.1.3 introduced the threats that are handled in detail in this thesis and in Section 6.2.2 we go into detail about metrics that can be used to evaluate the efficiency of each of these analysis modules.

6.2.1 Resource use

This section brings up important resources and what tools are used to evaluate the impact of the Security Supervisor on these resources. Resources which are mentioned are CPU, RAM, energy usage, and permanent storage.

6.2.1.1 Resources

One important resource to take into consideration is CPU usage. Home Assistant and the Security Supervisor should never use 100% CPU more than momentarily. As there is also an operating system on the hub, and applications do not use CPU resources evenly, there needs to be a safety margin on how much CPU the Security Supervisor can use without exhausting the system.

Many smart hubs are connected to mains power, meaning there are no direct restrictions to energy consumption. However, increased energy consumption could be both an environmental and economical issue.

Another significant resource to take into consideration is long-term storage memory. The Raspberry Pi 3b+ uses SD cards and can theoretically be supplied with any amount of memory. Huge SD cards are still very expensive though, and it is unlikely that a huge SD card will be used for Home Assistant. In addition, SD cards use flash memory that are susceptible to being worn out by write operations. While the flash memory in SD cards support a fair amount of write operations and many provide a wear leveling mechanism that distribute the write operations over the entirety of the memory to prolong the life of the memory card, it is important to keep the number of write operations to a minimum. Abusing write operations would contribute to expediting the deterioration of the SD card, which would be economically bad for the user as well as a hit to the environment.

A final resource to observe is the amount of RAM used. This is especially important to consider when dealing with network data buffers. The Security Supervisor will handle network data for multiple devices, which in turn means data rates may grow large and RAM usage with it. It is of utmost importance to keep RAM memory from maxing out, or the amount of dropped packets will radically increase.

6.2.1.2 Measurement tools

We used the well-known performance monitoring software sysstat to measuring the CPU usage on the Raspberry Pi B3+. Sysstat, first released in 1999, contains a wide range of features to measure CPU and memory usage. CPU statistics can be gathered for an entire system or each individual core, providing detailed views for users. Furthermore, sysstat keeps track of multiple properties of the state of the memory - buffer usages, amount of memory in cache and inactive memory amongst others. This data gives a good view of how the system performs under a certain load.

In order to measure the amount of energy used by the hub we used the hardware sensor Charger Doctor [70]. This hardware acts as middleware between the energy source and the device and displays real-time current usage with a resolution of 10mA. By documenting the displayed data over a period of time an estimate for energy usage can be presented.

To measure how many write operations are made in the system and how big these write operations are, we use the tool iostat [71]. This tool gives a detailed overview of the input/output operations to different parts of the system. It shows properties such as the number of reads/writes, size of reads/writes, and time the system spends waiting on I/O operations [71].

6.2.2 Analysis modules

The two security modules for botnet and Evil-Twin detection require different evaluation criteria and testing methodologies. This section presents these properties for each of the cases.

6.2.2.1 Botnet

The goal of the functionality tests for the botnet module is to find out how much malicious traffic the Security Supervisor is able to detect. We implement this in three steps. First, we collect real data from the network data stream into a file. Second, we append packets which indicate malicious use into this file. Finally, this file is injected into the network packet stream in order to simulate malicious activity and the number of detections from the module can be recorded.

The architecture of the Security Supervisor as described in Section 5.2.1 allows us to inject the malicious traffic directly into the internal packet stream of the Security Supervisor, which means we do not need to broadcast this traffic onto the actual physical network. Even though this might seem like a security issue that may open up the system to further attacks, this injection is only possible to perform with root privileges. It can be assumed the attacker does not have root access since if they do the device is already compromised and we can not defend ourselves against any internal attacks.

6.2.2.2 Evil-Twin

The goal for the functionality tests for the Evil-Twin analysis module was to find out weather the module was able to detect a difference between an attackre-controlled AP or not. Furthermore, it was important to evaluate whether there are any variability in the perceived signal strength that could interfere with the solution. Finally, we needed to take into account if inhabitants or obstructions in the home could cause significant interference to the results. The tests were performed without any obstructions and with minimal interference from inhabitants in order to increase comparability.

6.3 Experiments

This section presents specifics on experiments which were run to evaluate the performance of the Security Supervisor. We begin by presenting the experiments concerning resource use followed by details on experiments concerning individual analysis modules.

6.3.1 Resources

We began the resource tests by finding baseline values for all measurable properties when only Home Assistant was running in order to have something to compare the performance of our entire setup with. All baseline tests were performed in three sessions with periods of two hours each. The sessions were distributed over different times of the day and scheduled to run when the network was most active, ranging from 8am to 8pm.

We measured I/O statistics, CPU usage, and RAM usage including average values as well as maximum usage spikes. Data was retrieved in intervals of 10 seconds. Energy consumption, as shown on the display of the Charger Doctor, was noted every other second. Specifics of each test category are summarized in Table 6.1.

Resource	Test interval	Test period	Sessions	Total test entries
CPU	10 sec	2 hours	3 sessions	2160
RAM	10 sec	2 hours	3 sessions	2160
Storage	10 sec	2 hours	3 sessions	720
Energy	2 sec	30 min	3 sessions	2700

Table 6.1: Intervals for performing baseline tests on the system.

When all baseline tests were done, we conducted two more rounds of resource usage tests: one with only the botnet module running and one where the Security Supervisor ran with all analysis modules active. These tests were carried out in exactly the same way as the baseline tests. The results from the baseline tests were then compared to the Security Supervisor tests in order to form an opinion on the performance impact of the Security Supervisor on the system.

6.3.2 Threat detection

This section presents details on the experiments performed to validate the implementation of the analysis modules of the Security Supervisor.

6.3.2.1 Botnet module

Testing the detection of suspicious communication from devices in the smart home was done in two stages. First, valid data was used to make sure that the evaluation module did not produce false positives. Second, forged messages which could have been part of a DDoS attack was fed into the packet stream. The evaluation of the test was a comparison between the number of malicious messages and the number of reports of malicious activity from the evaluation module.

The malicious data sets were collected pcap files that were modified using WireEdit to match the devices in the testbed. Some files were obtained from pcaper.net [72, 73, 74, 75, 76] while some were crafted from a sample packet. In most test data sets, the source IP address and source MAC address are the same as those of a device in the smart home. In one data set the source IP address is simulated to be spoofed to some random address, and in another both IP address and MAC address are simulated to be spoofed.

The test that were carried out simulated a number of different DDoS attack types to find out if the system could handle different make-ups of packets. The following attacks were tested:

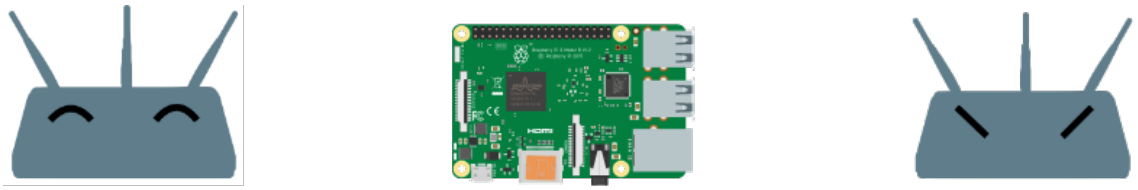
- IP fragment overlap
- Ping flood
- TCP SYN/FIN flood
- UDP DNS flood
- Smurf attack
- Smurf attack (spoofed MAC address)
- IPv6 flood

6.3.2.2 Evil-Twin module

The test of the Evil-Twin analysis module was carried out in the following way. A mobile device, in this case an Android smartphone in hotspot mode, was used to simulate a rogue AP. It used the same SSID and network password as our test network but did not spoof its MAC-address and behaviour as could be expected by a rogue AP. However, since our solution to the Evil-Twin attack does not focus on these properties, that factor is negligible. The rogue device was activated in different locations of the room and the result of the test was based on a compilation of how many times the analysis module detected the presence of the rogue device.

Three different cases where used for placement of the evil twin device. These are

(a) Case where the evil twin is placed between the real AP and the Security Supervisor.



(b) Case where the Security Supervisor is placed between the evil twin and the real AP.

(c) Case where the real AP is placed between the evil twin and the Security Supervisor.

Figure 6.2: Positioning of each device in test scenarios for the Evil-Twin detection module.

depicted in Figure 6.2. In case A, the router was placed 5m, 7m and 10m from the smart hub with 3 subcases each where the evil twin device was positioned either 1m from the router, 1m from the smart hub or in the middle between the two devices. For case B distances of 1m, 5m, 7m and 10m were used for the router with 3 subcases each where the evil twin device was placed on half the smart hub distance, equal distance and 1.5 times the distance away. For case C the distance between devices were combinations of the distances 1m, 5m, 7m and 10m. Some subcases have been ignored due to low likelihood of them ever happening in a realistic environment. A summary of all these test cases are presented in Tables 6.2a to 6.2c, where the above mentioned subcases are denoted *SC1*, *SC2* and *SC3*.

Router	Evil-Twin SC1	Evil-Twin SC2	Evil-Twin SC3
5m	-	2m	4m
7m	1m	3m	6m
10m	1m	5m	9m

(a) Positions of devices relative to the Raspberry Pi for case A.

Router	Evil-Twin SC1	Evil-Twin SC2	Evil-Twin SC3
1m	-	1m	5m
5m	2m	5m	7m
7m	3m	7m	10m
10m	5m	10m	-

(b) Positions of devices relative to the Raspberry Pi for case B.

Router	Evil-Twin +1m	Evil-Twin +5m	Evil-Twin +7m	Evil-Twin +10m
1m	1m	6m	8m	11m
5m	6m	10m	12m	15m
7m	8m	12m	14m	17m
10m	11m	15m	17m	20m

(c) Positions of devices relative to the Raspberry Pi for case C.

Table 6.2: The positioning of devices relative to each other in three test cases for the Evil-Twin module.

7

Results

This chapter presents results from the experiments on the Security Supervisor. Section 7.1 brings up results concerning resource usage whilst Section 7.2 focuses on what level of detection rate the analysis modules provide.

7.1 Resources

This section presents results from resource tests for CPU and RAM usage, permanent storage usage, and power consumption.

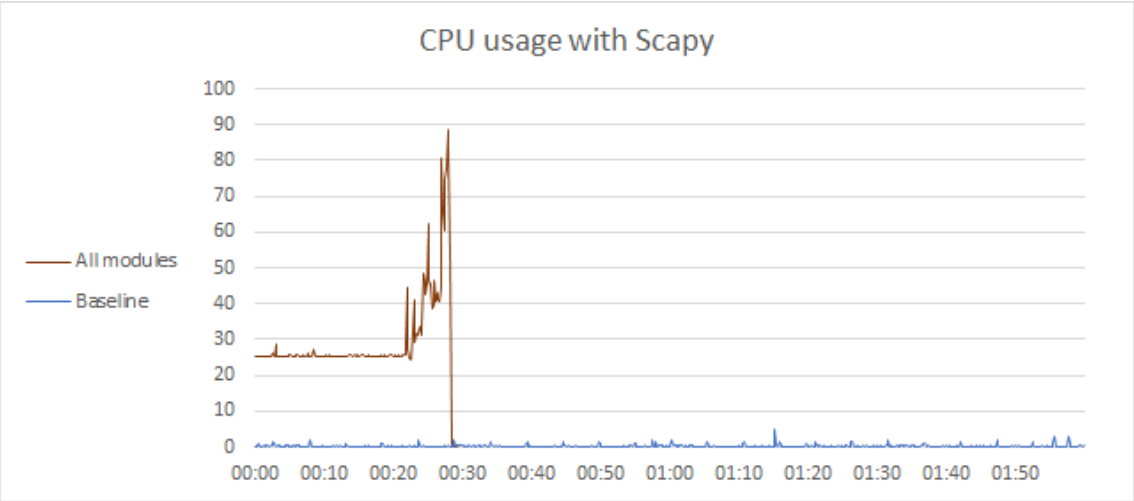
7.1.1 CPU and RAM usage

As mentioned in Section 5.2.1, two different network libraries were considered - Scapy and Pypacker. Performance in terms of CPU and RAM usage differed a lot between these two libraries. A comparison between the amount of packets each of them are able to process per second is presented in Table 7.1.

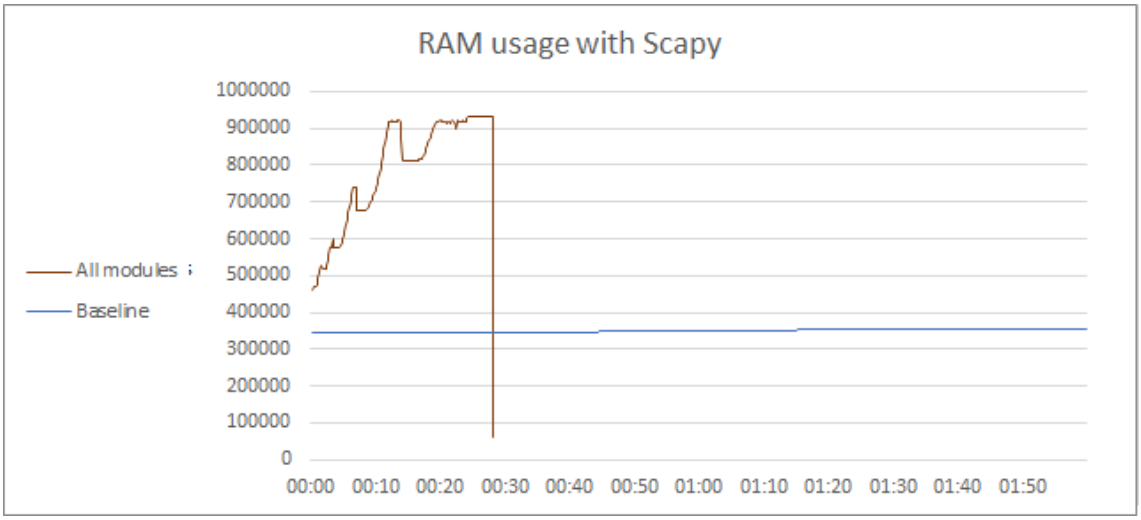
Library	Packets per second	
	Pypacker test [77]	Our testbed
Scapy	726	57
Pypacker	17 938	15 941

Table 7.1: Comparison between Scapy and Pypacker performance-wise.

Results from running the Security Supervisor with Scapy are presented in Figure 7.1. When exposed to a low network load with normal traffic from smart devices and a single AP, CPU usage averaged 14% and RAM usage did not differ from the baseline. However, when exposed to a higher network load with 6-8 nearby APs, Scapy was too slow to process data which led to a constant increase in RAM and a system crash within 29 minutes. APs generally send 10 beacon frames per second resulting in 60-80 frames per second for 6-8 APs, which exceeds the processing capacity of 57 packets per second for Scapy.



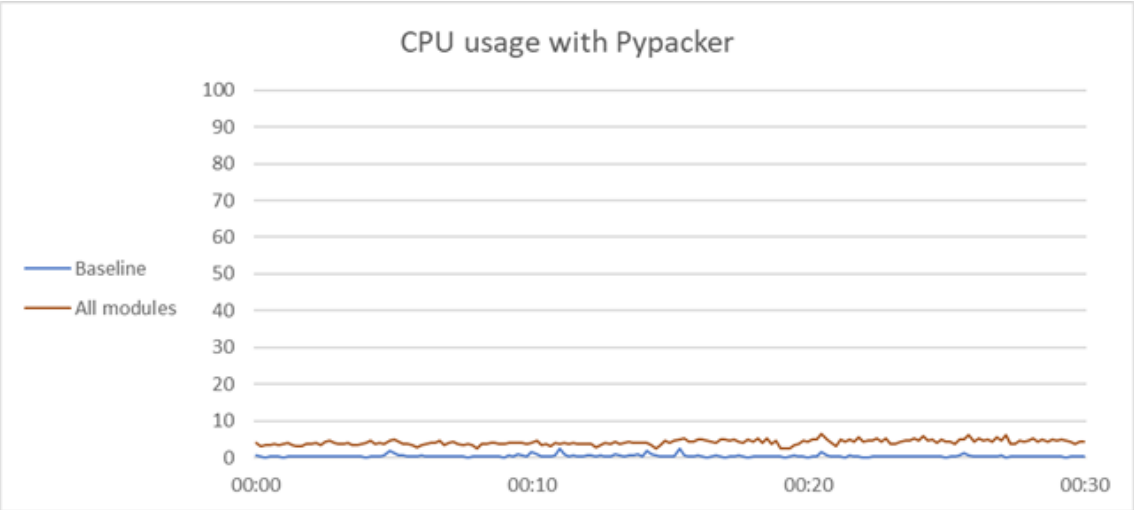
(a) CPU usage (%), high network load.



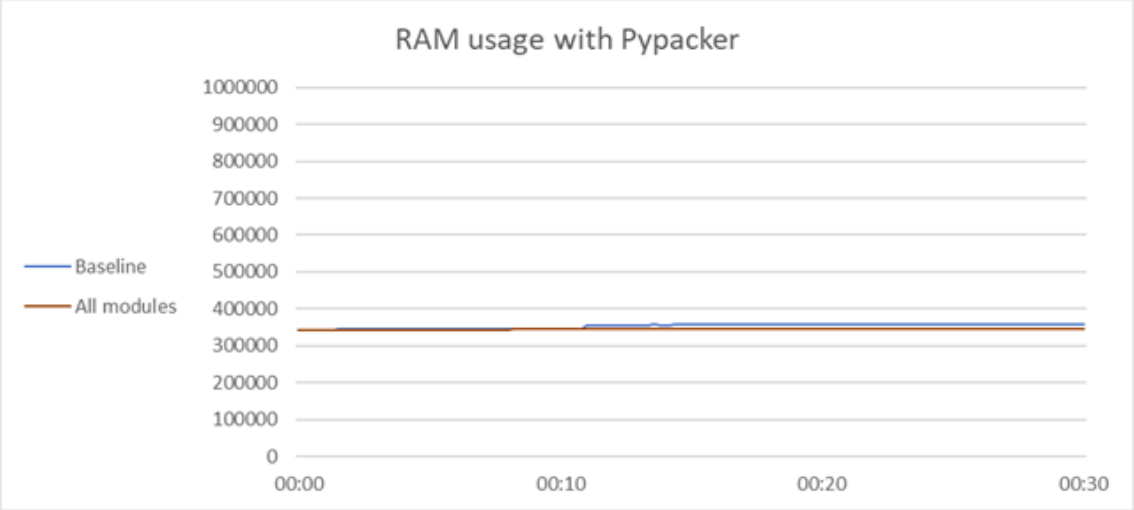
(b) RAM usage (KB), high network load.

Figure 7.1: CPU and RAM usage over time when using Security Supervisor with Scapy, each with low and high network traffic load.

Pypacker was able to handle a high load without any noticeable issues. When exposed to the higher network load, CPU usage averaged at 4% and RAM usage did not differ from the baseline. No tests were carried out for the easier case with lower network load due to the high performance of the library. Results from the Pypacker tests are presented in Figure 7.2. Note that these tests were carried out in periods of only 30 minutes instead of two hours due to two main reasons - the large expenditure of time to carry out tests and the fact that resource impacts are noticeable already early in the process.



(a) CPU usage (%), high network load.



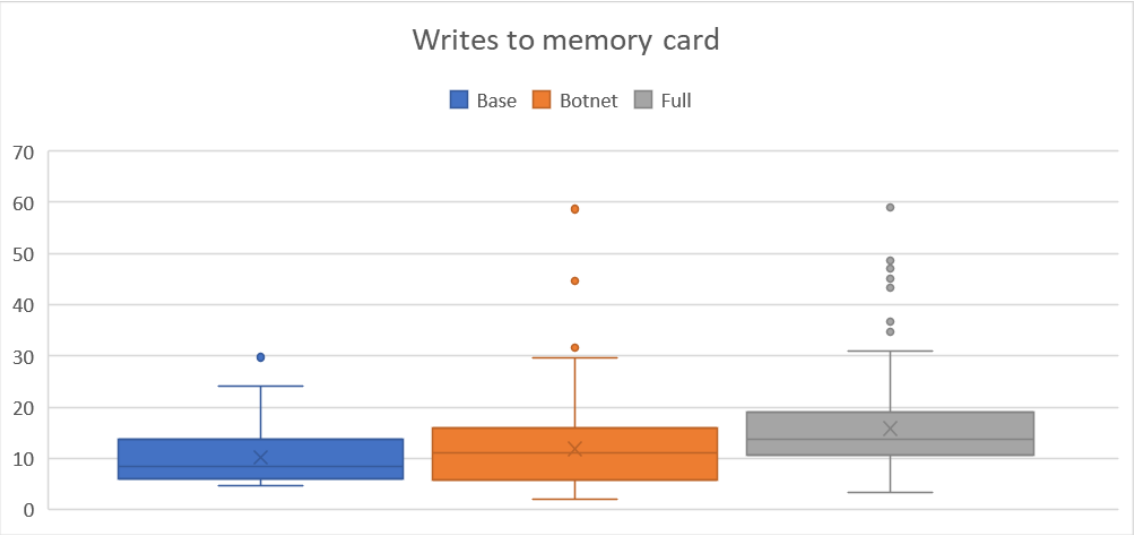
(b) RAM usage (KB), high network load.

Figure 7.2: CPU and RAM usage over time when using the Security Supervisor with Pypacker.

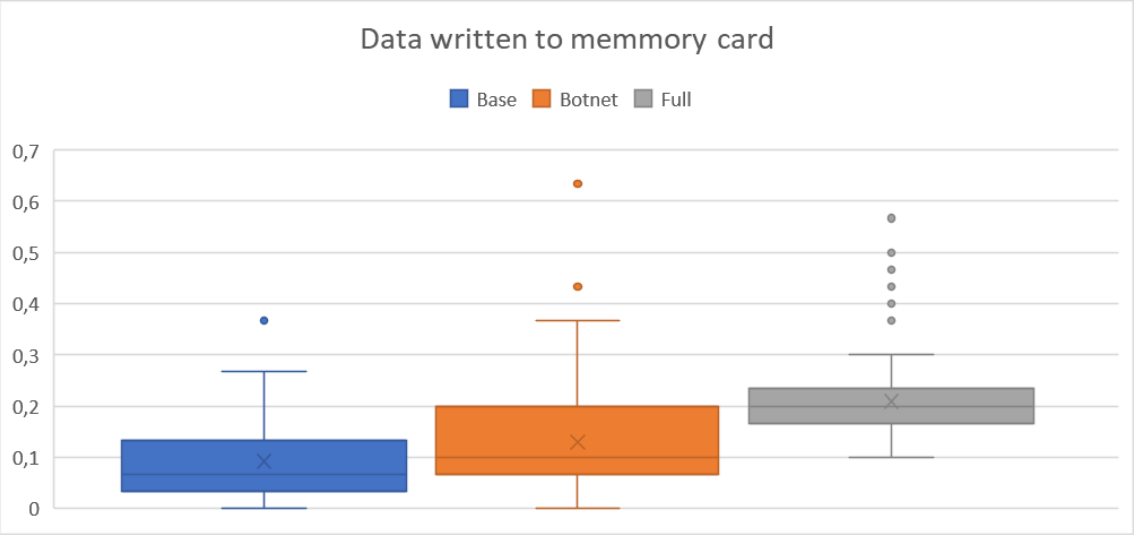
7.1.2 Permanent storage

Figure 7.3 shows an average of the results of the iostat runs for the tests. Both 7.3a and 7.3b show a comparison between the three cases of tests, namely the base version with only Home Assistant running, the botnet version where only the botnet detection module is running, and the full instance of the Security Supervisor. While all data sets show some significant variance in their values, it is clear from the charts that both the number of writes and the data written to the memory cards is the lowest in the base case and the highest in the full version.

From the values collected during the tests, we can make a rough estimate of the expected lifetime of the memory card in the different versions tested. To make this estimate, we need to establish some values.



(a) Memory writes over time.



(b) Data written to memory over time.

Figure 7.3: The amount of writes to memory over time, as well as the amount of data written to memory over time, for three different cases.

We use a 16GiB memory card. Out of these 16 GiB, 4GiB is taken up by the card's internal functionality, as well as the operative system of the Rasberry Pi. As such, there are 12GiB available for Home Assistant and the Security Supervisor. The file system write block size (wbs) of our operating system is 4KiB, and our memory card has an erase block size (ebs) of 4MiB. This gives us 3146 sectors (ws) available to use in the card. Assuming up to 3000 write cycles (wc) [78] per sector we can theoretically do $9.4 * 10^6$ 4Kib writes in total to our memory card.

From the iostat data, we discovered the average number of writes per second (wps), as well as the average size of these writes (s). Using this data we can construct a formula that gives us the lifetime of the card in years. The results of this formula

for each test case can be found in Table 7.2. The derivation of this formula can be seen below.

Find the amount of write blocks per average write:

$$\left(\frac{wb}{w}\right) = \begin{cases} \dfrac{s}{wbs} & \text{if } s \bmod wbs = 0 \\ \dfrac{s}{wbs} + 1 & \text{otherwise} \end{cases} \tag{7.1}$$

Find the amount of writes per write sector:

$$wpws = \frac{\frac{wb}{w}}{\frac{ebs}{wbs \times \frac{wb}{w}}} = \frac{wbs \times \left(\frac{wb}{w}\right)^2}{ebs} \tag{7.2}$$

Find the amount of total available writes to the memory card:

$$tw = wpws \times ws \tag{7.3}$$

Find expected lifetime for the memory card (in years):

$$years = \frac{tw}{wps} \times (60 \times 60 \times 24 \times 365)^{-1} \tag{7.4}$$

As the results show in the year column of Table 7.2 the theoretical lifetime of the card decreases with 49 years due to the additional load of the Security Supervisor.

Version	w/s	KiB/w	wb/w	w/sector	w	y
Only Home Assistant	1.1	9.8	3	333	$3.1e^9$	91
Only botnet module	1.2	11.4	3	333	$3.1e^9$	81
Full Security Supervisor	1.8	14.0	4	250	$2.4e^9$	42

Table 7.2: Estimated lifetime for the memory card given the three test cases. The average writes per second from our test data can be found in the w/s column, while the KiB/w denotes the measured average size per write.

7.1.3 Power consumption

Power consumption in the setup is affected by two factors: current draw by the wireless USB network card and the current draw by the Raspberry Pi itself. Figures

7.4 and 7.5 present how the energy consumption of these two devices varied over time. The energy consumption of the wireless network card was not affected by the Security Supervisor at all, whilst the Raspberry Pi experienced a small increase in power consumption. The Raspberry Pi used slightly more energy with the Security Supervisor running than when only Home Assistant core modules were active.

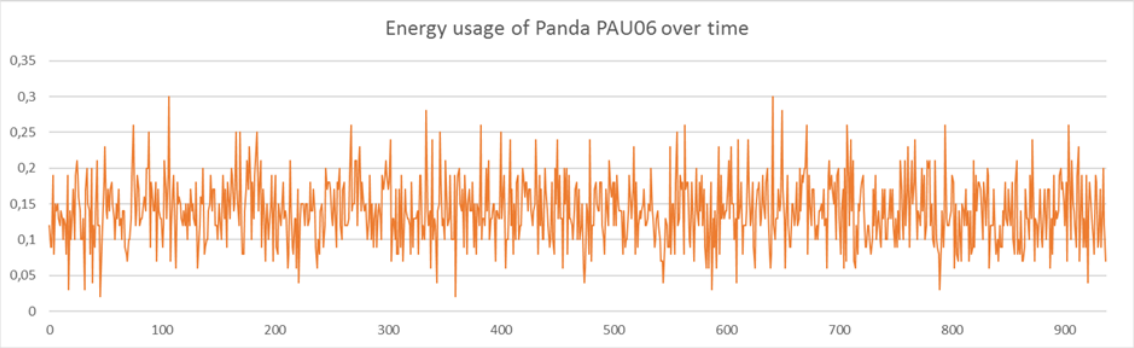


Figure 7.4: Energy usage over time of wireless network card Panda PAU06 in terms of current, measured in ampere.

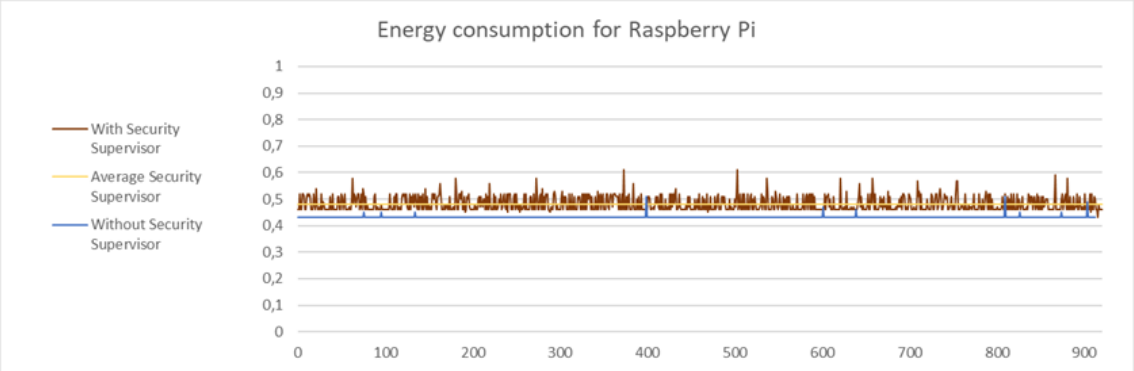


Figure 7.5: Energy usage over time of Raspberry Pi running Home Assistant in terms of current, measured in ampere.

Device	Baseline	With Sec. Sup.	Increased usage		
			Current	Cost/month*	CO^2/month
Raspberry Pi	0.43A	0.48A	0.05A	0.12SEK	0.133kg
Panda PAU06	0A	0.14A	0.14A	0.36SEK	0.385kg
Total	0.43A	0.62A	0.19A	0.60SEK	0.518kg

* Cost is based on 0.70SEK/kWh according to statistics from Vattenfall [79] and a voltage of 5V.

Table 7.3: Energy usage without and with the Security Supervisor running.

Since the voltage for both the Raspberry Pi as well as the wireless network card was 5V at all times, we can compare energy usage based on the amount of current used. Average current usage for the Raspberry Pi increased from 0.43A to 0.48A when using the Security Supervisor. The wireless network card averaged at 0.14A. Table 7.3 summarizes the energy usage measurements for both of these devices.

7.2 Analysis modules

This section presents results from the tests for the implemented security modules of the Security Supervisor and their detection rates.

7.2.1 Botnet module

Table 7.4 shows the results from the tests on the botnet detection module. As can be seen in the first two rows of the table, the module did not give any false positives. In the case where the source IP and MAC addresses were that of the legit device, the module caught all instances of packets where the outgoing address was one not already profiled. In the row for Smurf-type attacks, we can see that the module can even catch the cases where the IP address of the source is spoofed to not be that of the legit devices. However, as seen for the entry *Smurf (with spoofed MAC)*, the module cannot detect attacks where both IP address and MAC address are spoofed.

DDoS attack	**Total**	**Normal**	**Malicious**	**Detected**
Legit traffic	300	300	0	0
UDP flood	29	16	13	13
IP fragment overlap	9	0	9	9
Ping flood	10	0	10	10
Smurf	7	0	7	7
Smurf (spoofed MAC)	7	0	7	0
TCP SYN/FIN flood	9	0	9	9
UDP DNS flood	8	0	8	8
IPv6 UDP flood	8	0	8	8
Sum	**387**	**71**	**316**	**64**

Table 7.4: Results from running the botnet analysis tests.

7.2.2 Evil-Twin module

For the Evil-Twin module, we performed base case tests to measure the amount of false positives and to use for comparisons with attack data. Figure 7.6 presents the variation among base cases for distances of one, five, seven and ten meters. There are two samples for each represented distance with the exception of one meter which features three samples.

Attack tests for the module comprised different combinations of positions of the router and evil twin device in relation to the Raspberry Pi. Test cases were split into three parts: one where the router was in the middle, one where the Raspberry Pi was in the middle and one where the evil twin device was in the middle. Each

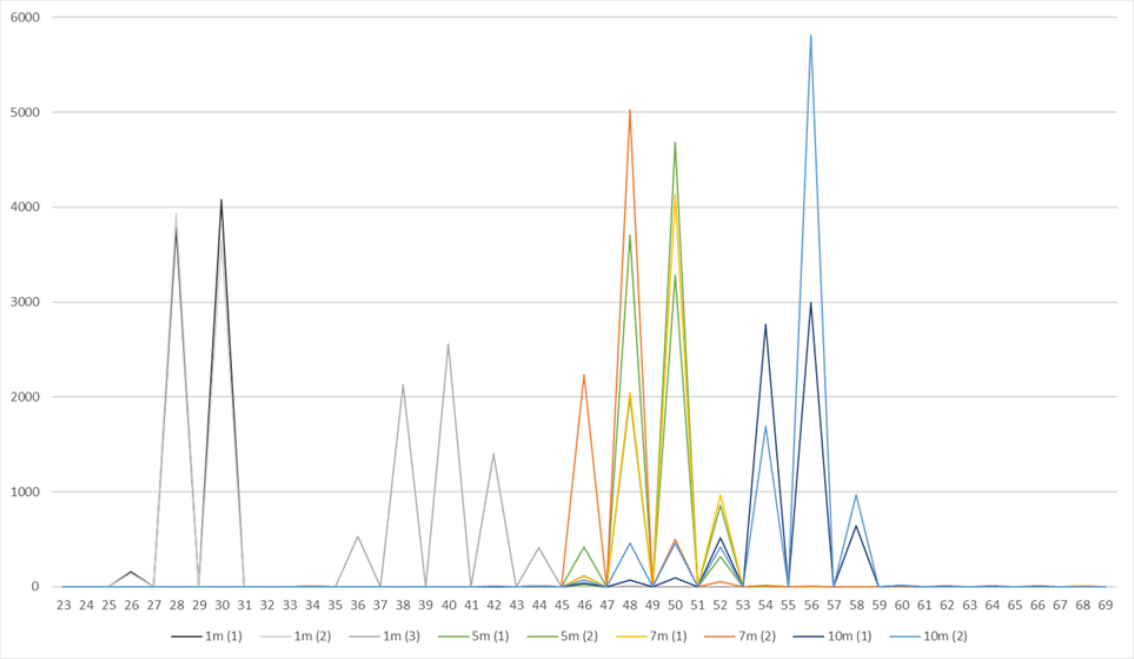


Figure 7.6: Comparison of base test signal strengths for the Evil-Twin module.

of these cases are presented in Table 7.5, 7.6 and 7.7 where the statistical model described in Section 5.2.2 is used.

Router	Basecase 2*	Basecase 3*	Evil-Twin 1		Evil-Twin 2		Evil-Twin 3	
			Dist.	Result	Dist.	Result	Dist.	Result
5 meters	210	75	-		*2m*	14	*4m*	742
7 meters	135	0	*1m*	14	*3m*	14	*6m*	2
10 meters	1605	2130	*1m*	182	*5m*	127	*9m*	5

* Calculated off runtime with an algorithm without timing data of packets.

Table 7.5: Detections reported from Evil-Twin analysis tests with the statistical model for case A, where the evil twin device is between the router and the Raspberry Pi. Numbers in comparison to basecase 1.

Table 7.8, 7.9 and 7.10 present the number of detections for the three cases when an alternative detection model for the Evil-Twin module was evaluated as a comparison. This model uses a simple max-min approach to regulate accepted signal strength levels, where values outside the maximum and minimum recorded values during the profiling phase is considered potential threats.

Router	Basecase 2*	Basecase 3*	Evil-Twin 1		Evil-Twin 2		Evil-Twin 3	
			Dist.	Result	Dist.	Result	Dist.	Result
1 meters	165	150	-		*1m*	0	*5m*	9
5 meters	210	75	*2m*	595	*5m*	1400	*7m*	602
7 meters	135	0	*3m*	65	*7m*	1136	*10m*	973
10 meters	1605	2130	*5m*	289	*10m*	1002	-	

* Calculated off runtime with an algorithm without timing data of packets.

Table 7.6: Detections reported from the Evil-Twin analysis tests with the statistical model for case B, where the Raspberry Pi is between the router and the evil twin device. Numbers in comparison to basecase 1.

Router	Basecase 2*	Basecase 3*	ET 1m	ET 5m	ET 7m	ET 10m
1 meters	165	150	6	1	3	7
5 meters	75	30	115	755	865	678
7 meters	45	1110	3	8	7	2
10 meters	3555	4920	9	14	8	12

* Calculated off runtime with an algorithm without timing data of packets.

Table 7.7: Detections reported from the Evil-Twin analysis tests with the statistical model for case C, where the router is between the Raspberry Pi and the evil twin device. Numbers in comparison to basecase 1.

Router	Basecase 2	Basecase 3	Evil-Twin 1		Evil-Twin 2		Evil-Twin 3	
			Dist.	Result	Dist.	Result	Dist.	Result
5 meters	4	6	-		*2m*	13	*4m*	22
7 meters	0	0	*1m*	2	*3m*	2	*6m*	0
10 meters	1	5	*1m*	3	*5m*	0	*9m*	1

Table 7.8: Detections reported from the Evil-Twin analysis tests with the max-min-model for case A, where the evil twin device is between the router and the Raspberry Pi. Numbers in comparison to basecase 1.

Router	Basecase 2	Basecase 3	Evil-Twin 1		Evil-Twin 2		Evil-Twin 3	
			Dist.	Result	Dist.	Result	Dist.	Result
1 meters	8033	7732	-		*1m*	1	*5m*	0
5 meters	4	6	*2m*	6	*5m*	11	*7m*	17
7 meters	0	0	*3m*	7	*7m*	1	*10m*	6
10 meters	1	5	*5m*	6	*10m*	6	-	

Table 7.9: Detections reported from the Evil-Twin analysis tests with the max-min-model for case B, where the Raspberry Pi is between the router and the evil twin device. Numbers in comparison to basecase 1.

Router	Basecase 2	Basecase 3	ET 1m	ET 5m	ET 7m	ET 10m
1 meters	8033	7732	0	3	0	1
5 meters	0	0	0	0	1	0
7 meters	3	5	0	1	0	0
10 meters	0	0	0	0	0	0

Table 7.10: Detections reported from the Evil-Twin analysis tests with the max-min-model for case C, where the router is between the Raspberry Pi and the evil twin device. Numbers in comparison to basecase 1.

8

Discussion

In this chapter we will discuss both the results from our tests and some points pertinent to this work. We begin the chapter with Section 8.1 in which we discuss the results presented in Chapter 7. Section 8.2 discusses the viability to run an IDS on a smart hub based on performance tests as well as internal security concerns, followed by Section 8.3 which brings up the subject of sustainability and ethics. The chapter is concluded in Section 8.4 where we propose some directions for further research on the topic of detecting threats within smart homes.

8.1 Evaluation of test results

In this section we evaluate the test results from Chapter 7. Section 8.1.1 evaluates and discusses tests on resources whilst Section 8.1.2 discusses how well the analysis modules detect threats.

8.1.1 Resource usage

In this subsection we discuss the results from the resource usage test and how they affect the viability of the Security Supervisor. We begin with the CPU usage, continue with RAM usage and finish with permanent storage memory usage.

8.1.1.1 CPU usage

As can be seen in Figure 7.1 and Figure 7.2, Home Assistant only uses very little of the Raspberry Pi 3B+'s CPU resources, spanning between 0.1% - 2.6% and averaging at 0.4%. As such, more than 90% of the CPU is not utilized while running Home Assistant. The Home Assistant running on the test setup is fairly minimal, and as such it is likely that some more CPU resources may be needed for Home Assistant to function properly in a less minimal setup, e.g. with an active voice assistant or security camera. To allow for some additional CPU resources to be

assigned to Home Assistant, as well as allow for some unexpected usage spikes, the CPU usage limit of the Security Supervisor is set to 30%.

The scaling potential of the Security Supervisor is affected by two factors - number of security modules which are supposed to process the data and amount of network traffic going in to the system. Based on our test results performance in terms of security modules are promising, as required processing power is not noticeably affected by turning on or off any existing modules. However, network traffic has a bigger impact on the system. When an increased amount of network traffic is encountered, the CPU usage can be seen to increase as well. This is especially significant when using the Scapy network library as shown in Figure 7.1, where the entire Home Assistant process crashes when too much data is encountered. Pypacker, the other network library used in this thesis, is able to handle much more traffic and our data suggests it will be powerful enough for a regular smart home. Figure 7.2 shows that our limit of maximum 30% CPU usage is easily met by Pypacker, even in a scenario with high network traffic load.

8.1.1.2 RAM usage

RAM usage of the Security Supervisor is rather low compared to the capacity of the entire system. In both of the cases in Figure 7.1 and 7.2 where it is able to process all data before new data is received, the increase of RAM maxes out at 20MB. However, when the Security Supervisor is unable to process packets sufficiently fast the RAM usage becomes a large concern. Packets which has not been processed stacks in memory and will in time force the module to shut down.

An alternative solution to keep an excessive amount of unprocessed packets in memory would be to drop packets the Security Supervisor can not process. Dropping packets can be done in two ways, either silently, or with notifying the user of the event. However, neither way is without fault. Silently dropping packets has a major downside in that the user is not aware that it has occurred, which could mean that the user is unaware of a security issue. On the other hand, notifying the user of dropped packets may overwhelm the user with notifications if the network is often overloaded. Given that the performance with Pypacker has been satisfactory, and that neither method is obviously superior to the other we have decided to not drop packets at all, and have not explored these ideas further.

8.1.1.3 Memory usage

The increase in writes to memory and the data written to memory significantly lowers the lifetime of the memory card from 91 years to 42 years. The largest decrease in lifetime is due to the Evil-Twin module, most likely due to the large increase in traffic that needs to be evaluated. As such, we can conclude that a large increase in network traffic has an adverse result on the lifetime expectancy of the

memory card.

A way to decrease the load on the memory card could be to read network traffic in a RAM buffer instead of to and from a file. This would increase the load on RAM, but decrease the load of the memory card. Which option for network traffic handling to use can in this instance be seen as a trade-off between RAM and memory card lifetime. While the decrease in lifetime is significant, it is also worth noting that expected lifetime of the card is over 40 years, much longer than the expected lifetime of most home electronics on the market today. In fact, it is likely that the memory card would fail for some other reason well before this. One could thus argue that as long as the expected lifetime is long enough, the decrease as such is less important.

8.1.2 Threat detection

In this section we discuss the functionality tests of the security modules in the Security Supervisor. We begin by discussing the results from the botnet module and finish with a discussion on the Evil-Twin module.

8.1.2.1 Botnet

Performance of the botnet module is good and it is able to detect most of the malicious cases. The simplicity of the module itself makes it a good showcase of the capabilities of the Security Supervisor framework - common DoS attacks without spoofed MAC addresses that originate from devices in smart homes can be detected with 100% detection rate by adding less than 100 lines of code.

The botnet module also showcase the shortcomings of the Security Supervisor. Base premises of the infrastructure makes it immensely vulnerable to spoofing attacks since only traffic to and from smart home devices are being processed. Attacks that spoof the source addresses are thus impossible as the traffic they produce would not be considered to be within the smart home. It would be possible to change the infrastructure in such a way that unexpected network traffic on the network as a whole is also included in the analysis, however it would come with additional issues for suitable user interfaces and might also increase the amount of network traffic the Security Supervisor has to handle. As mentioned in Section 8.1 this is very undesirable.

Furthermore, the botnet module is particularly vulnerable to DNS spoofing attacks. As mentioned in Section 5.2.2, this module depends on registered DNS responses to make correct judgements on whether a device talks to a known entity or not. Consequently, the module needs to be able to trust any received DNS data to be able to make correct decisions for all cases. It is therefore important to extend the Security Supervisor to be able to detect DNS spoofing attacks to guarantee that the botnet module functions correctly.

8.1.2.2 Evil-Twin

Results from the Evil-Twin tests show a lack of success for both of the detection models used. Figure 7.6, where a comparison between base tests are shown, may contain an explanation to this matter. Recorded values for 5 meters and 7 meters show a large overlap. Even though there is a significant difference between signal strengths for 7 meters and 10 meters, entries for 1 meters displays a large disunion where two samples overlap almost identically whilst the third does not overlap at all. This data suggests that while signal strength do change over distance, it is not a reliable measurement and cannot be expected to be precise.

Tests for the statistical model show both success and failure. For case A, the number of false positives are similar or even worse than the number of true positives, and hence an attack is not detectable. In case B there is a large difference in magnitude between the number of false positives and true positives for distances of 5 and 7 meters, which suggests indications of a working detection process. However, samples for one and ten meters show a lack of precision. Finally, case C shows both significant detection of malicious data and significant detection of data which in fact is not malicious. Conclusively, as can be seen from these cases, a detection in the model cannot be trusted to be correct.

The max-min-model gives detections of an entirely different magnitude than the statistical model. Even though the analysis was active for 15 minutes with an evil twin constantly broadcasting beacon frames, the maximum amount of malicious reports were 22. This represents less than 0.3% of the received beacon frames. Furthermore, none of the cases show any significant difference between false positives and true positives. Additionally, a flat profiling phase can cause the number of detections to increase rapidly as shown for reports on one meter. Therefore, this model is not a well-functioning solution either.

Ultimately, our test data indicates that detection of an Evil-Twin attack with the help of signal strength is not possible with our current models. However, some result entries look promising and might be useful if a different model or different hardware is used. More antennas would allow for triangulation and hence a more accurate detection. If a sufficient model is provided, the Security Supervisor will be able to support detection of Evil-Twin attacks.

8.2 Evaluation of the Security Supervisor

In this section we evaluate the viability of the Security Supervisor and discuss it on a detailed level. Section 8.2.1 discusses if the smart hub is a viable placement for an IDS, followed by Section 8.2.2 which states current security concerns within the Security Supervisor implementation. Section 8.2.3 completes the section with a discussion on the current state of the Security Supervisor in relation to a potential

full production-ready solution.

8.2.1 Placement of the Security Supervisor

One of the key questions for this thesis was to find out if a smart hub is a viable placement for threat detection. An initial issue was how to get access to the network data to be processed. We needed to introduce an external network card with monitoring capabilities to be able to retrieve this network data. This network card is not equipped by default on a Raspberry Pi 3B+ and such capabilities are not generally available. However, our solution shows that it is not outside the capabilities of a hub of similar performance to handle such functionality.

Results in Chapter 7 and the discussion in Section 8.1.1 show that the hub can handle the load performance-wise. However, CPU and RAM usages increases substantially when network traffic increases which might indicate, as mentioned above, that the solution does not scale well in traffic heavy networks. One reason behind this could be the amount of steps taken to process a single data packet as all data is captured, written to a file, read from the file, parsed and then processed. If data were to be transferred immediately to the Security Supervisor, two of these intermediate steps could be eliminated and performance might increase at the cost of modularity and ease-of-use.

It could be argued that the threat detection should be placed in the router or another peripheral as Batalla et al. do in [23]. However, in these cases there is a risk for system lock in where information about the devices and their automatons cannot be retrieved. Our model takes advantage of the positioning and uses information from the smart hub configuration to filter out network traffic to and from smart home devices and hence ignores irrelevant data for the threat detection mechanism. Furthermore, it is possible to take advantage of knowledge of states to check whether a device performs as expected, e.g. if your light reports an active 'on' state and the light sensor reports darkness there might be an issue in the system. Currently no security module in our implementation checks these states but the overall framework support interaction with smart hub data.

An additional issue with placing threat detection for smart homes in the router is that while smart homes may use a variety of network technologies, such as Zigbee or Z-Wave, home routers usually only support Wi-Fi and Ethernet technologies. With threat detection placed in the smart hub, raw data from devices using smart home network technologies can be monitored and analyzed to find more threats.

8.2.2 Security concerns within the Security Supervisor

There are a few security issues in the current implementation of the Security Supervisor. These comprise both internal vulnerabilities and threats towards detection

algorithms.

Firstly, the reading of network data is something that requires root access on the Raspberry Pi. Home Assistant does not have root access to the system for security reasons. In the presented solution network data collection is handled by a stand-alone script outside of Home Assistant in order to get around this issue, but it could have been solved with an SUID binary as well. When well designed, none of the above mentioned solutions lets a malicious intruder access root privileges in case of intrusion into Home Assistant. However, they do allow an intruder to observe network data.

Secondly, the profiling phase opens up the Security Supervisor to security breaches. During the profiling, the Security Supervisor is vulnerable to profiling incorrect behaviour. An attacker could potentially time their attack to coincide with the profiling and thus have attack behaviour be considered legit behaviour by the Security Supervisor.

Finally, as the current solution stands, one of the details of the Evil-Twin detection process is open to being tricked. In the cross-referencing phase, where movement of the hub is to be detected, there is a potential for an attacker simulate a smart hub movement. If the attacker introduces numerous new APs and waits for the profiling end, they can then move these APs in order to trick the Evil-Twin detection module that the hub itself has moved.

8.2.3 The Security Supervisor in a larger picture

It is our opinion that it is viable to run a small-scale IDS on a smart hub resource wise. However, much work is needed to transform this prototype into a system ready for production.

In its current form, the Security Supervisor lacks coverage for adequate intrusion detection. Only two analysis modules covering select attacks are available. To get a satisfactory coverage more analysis modules that handle additional threats are needed. From the data that we have, we believe that the Security Supervisor could support many modules, as the majority of the resource expenditure currently comes from collecting network traffic and not the detection process itself.

The current setup is portable enough to be implemented in other systems with moderate modification. Care would have to be taken to not tie the system too hard to the platform in further development. Portability is key to allow widespread adoption and thus benefiting as many users as possible.

8.3 Sustainability and ethics

The risks of security breaches in smart homes can be very high for the individual. An insecure smart home can thus be considered a danger to the user. The Security Supervisor is an attempt at mitigating the risks to the user, making the smart home a safer environment. However, these security benefits have to be weighted against the environmental impacts caused by the Security Supervisor.

The Security Supervisor definitively increases the energy usage compared to only running Home Assistant, as we can see in Table 7.3. Much of this increase comes from the network card in monitoring mode, a device that is unfortunately needed for the Security Supervisor to function. This might be an additional reason for placing intrusion detection in the router, as discussed earlier in section 8.2.1. Reducing the number of high-energy consuming antennas would have a positive effect on energy usage. While the energy increase is definitive, it is not particularly big - energy usage of the Security Supervisor during an entire month is equal to driving a car 2km in terms of CO^2 emissions [80]. It is our opinion that this energy expenditure could be justifiable if it can provide a solution to mitigate security risks for smart homes.

Furthermore, it is apparent from Table 7.2 that the lifetime of the memory card is significantly shortened by the implementation of the Security Supervisor. A decrease in memory card lifetime could mean an increase in demand on memory cards as well as more waste in the form of run out memory cards. However, the lifetime of the memory card with the Security Supervisor is still significantly long, especially considering the field of electronics. One can even say that the lifetime is long enough that it will probably outlive its use-case by several years. It is our opinion that while the amount of writes to the card is higher than desired, and work should be done to rectify this, the current lifetime of the card is not a major environmental or economical issue for the Security Supervisor. This is due to the fact that the current lifetime is long enough that the memory card will probably be replaced due to other reasons than being worn out due to the amounts of writes it has had to sustain.

8.4 Future work

This section presents several possible directions on how to extend this work, covering both continuation of the Security Supervisor and alternative approaches to intrusion detection in smart homes based on our experiences during the thesis work.

8.4.1 Improve user interaction

In this work we have only described methods to detect issues inside the smart home and not how this information is best presented to the user. This is an important step in threat detection and there has been a lot of research going into this [81, 82, 83]. Further work in how to present the data to the user is needed in order to achieve trust and understanding from end users.

8.4.2 Adapt a conventional IDS

One issue with the approach used to build the Security Supervisor is that it is implemented from scratch. Consequently, a multitude of modules need to be implemented to reach a satisfactory level of detection. A more time-effective approach could instead be to re-purpose a conventional IDS.

A conventional IDS comes with the advantages that it is well-tested, has models for handling known attacks and already has a wide user-base. However, a full-fledged IDS is not suitable for the smart hub, as devices with comparable resources to a smart hub lacks the resources to run a full IDS [25, 26]. These performance issues would likely be exacerbated with the added functionality needed to be able to handle the new protocols and communication techniques unique to the smart home environment.

While a conventional IDS is not suitable for the smart hub as is, it could be worthwhile to investigate if it could be used as a base to develop a solution for a smart hub.

8.4.3 Extend the current solution

The Security Supervisor needs to be extended with more functionality before it can be viewed as a full-fledged IDS. This work is twofold. States of smart home devices need to be incorporated into detection solutions to take full advantage of the positioning of the detection mechanism. Furthermore, several more security modules need to be added in order to support more attacks. Implementation for additional technologies is needed as well.

9

Conclusion

It is very clear that smart homes are, and will continue to be an appealing target in cybercrime. Security breaches in smart homes can be used as steppingstones for cybercrime that can have far-reaching and severe consequences for both society and individuals. As shown in Chapter 4 smart homes with their multitude of devices are an appealing target for obtaining DDoS slaves. These DDoS attacks could potentially target important societal infrastructures, causing severe damage. Smart homes also makes it possible for attackers to collect sensitive information about individuals, which could cause severe economical damage or even physical harm to the individual.

In this thesis work we have focused on developing and evaluating a security solution for the smart home that can mitigate security threats against individuals and society. Solutions to mitigate two specific attacks have been implemented. The attacks in question are the use of smart home devices in botnets, and the Evil-Twin attack, where an adversary tries to steal devices from an access point to steal information.

The current Security Supervisor is not able to be deployed at this time. Further improvement is necessary. This is because there are significant weaknesses in the functionality of the security modules. A large amount of additional security modules needs to be developed as well to offer a wider security solution.

From a resource usage perspective we believe that it is possible to place an IDS for the smart home on a smart hub. It is our opinion that a smart hub of a similar capacity to the Raspberry Pi has enough hardware and processing capability to handle a limited IDS while still being able to perform its function as a smart hub. We encourage further research in the area and look forward to find out where this research field may evolve.

Bibliography

[1] A. Jacobsson, M. Boldt, and B. Carlsson, "A risk analysis of a smart home automation system," *Future Generation Computer Systems*, vol. 56, no. Supplement C, pp. 719 – 733, 2016.

[2] Home Assistant, "Components architecture," accessed: 2017-06-13. [Online]. Available: https://developers.home-assistant.io/docs/en/architecture_components.html

[3] "List of network protocols (OSI model)," accessed: 2018-09-03. [Online]. Available: https://en.wikipedia.org/wiki/List_of_network_protocols_(OSI_model)

[4] OWASP, "Top IoT vulnerabilities - OWASP," accessed: 2018-08-22. [Online]. Available: https://www.owasp.org/index.php/Top_IoT_Vulnerabilities

[5] T. Denning, T. Kohno, and H. M. Levy, "Computer security and the modern home," *Commun. ACM*, vol. 56, no. 1, pp. 94–103, Jan. 2013. [Online]. Available: http://doi.acm.org/10.1145/2398356.2398377

[6] IDATE DigiWorld, "Digiworld yearbook 2016," June 2016.

[7] Gartner, Inc., "Gartner says 4.9 billion connected "things" will be in use in 2015," November 2014, accessed 2018-05-03. [Online]. Available: https://www.gartner.com/newsroom/id/2905717

[8] ——, "Gartner says a typical family home could contain more than 500 smart devices by 2022," September 2014, accessed 2018-08-13. [Online]. Available: https://www.gartner.com/newsroom/id/2839717

[9] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *2016 IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 636–654.

[10] O. Shwartz, Y. Mathov, M. Bohadana, Y. Elovici, and Y. Oren, "Opening Pandora's Box: Effective Techniques for Reverse Engineering IoT Devices," in *Smart Card Research and Advanced Applications*, T. Eisenbarth and Y. Teglia, Eds. Cham: Springer International Publishing, 2018, pp. 1–21.

[11] J. Best, "Wake up baby: Man hacks into 10-month-old's baby monitor to watch sleeping infant," April 2014, accessed 2018-05-03. [Online]. Available: https://www.mirror.co.uk/news/world-news/man-hacks-10-month-olds-baby-monitor-3468827

[12] A. Charlton, "How a thermostat in the lobby fish tank let hackers steal a casino's high-roller database," April 2018, accessed 2018-05-03. [Online]. Available: https://www.gearbrain.com/iot-hack-on-casino-aquarium-2560513466.html

[13] L. Mathews, "Hackers use DDoS attack to cut heat to apartments," November 2016, accessed 2018-08-23. [Online]. Available: https://www.forbes.com/sites/leemathews/2016/11/07/ddos-attack-leaves-finnish-apartments-without-heat/#6abaa9111a09

[14] M. M. Hossain, M. Fotouhi, and R. Hasan, "Towards an analysis of security issues, challenges, and open problems in the internet of things," in *2015 IEEE World Congress on Services*, June 2015, pp. 21–28.

[15] S. Raza, H. Shafagh, K. Hewage, R. Hummen, T. Voigt *et al.*, "Lithe: Lightweight secure CoAP for the Internet of Things," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3711–3720, 2013.

[16] J. Granjal, E. Monteiro, and J. S. Silva, "Security for the Internet of Things: A survey of existing protocols and open research issues," *IEEE Communications Surveys Tutorials*, vol. 17, no. 3, pp. 1294–1312, Jan 2015.

[17] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, Oct 2017.

[18] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661 – 2674, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870513001005

[19] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman, and R. Boreli, "An experimental study of security and privacy risks with emerging household appliances," in *2014 IEEE Conference on Communications and Network Security*, Oct 2014, pp. 79–84.

[20] D. H. Summerville, K. M. Zach, and Y. Chen, "Ultra-lightweight deep packet anomaly detection for Internet of Things devices," in *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*, Dec 2015, pp. 1–8.

[21] J. R. C. Nurse, S. Creese, and D. D. Roure, "Security risk assessment in Internet of Things systems," *IT Professional*, vol. 19, no. 5, pp. 20–26, October 2017.

[22] V. Sivaraman, H. H. Gharakheili, A. Vishwanath, R. Boreli, and O. Mehani, "Network-level security and privacy control for smart-home IoT devices," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2015, pp. 163–167.

[23] J. M. Batalla, A. Vasilakos, and M. Gajewski, "Secure smart homes: Opportunities and challenges," *ACM Comput. Surv.*, vol. 50, no. 5, pp. 75:1–75:32, Sep. 2017. [Online]. Available: http://doi.acm.org/10.1145/3122816

[24] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *Journal of Network and Computer Applications*, vol. 84, pp. 25 – 37, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804517300802

[25] A. Sforzin, F. G. Mármol, M. Conti, and J. Bohli, "RPiDS: Raspberry Pi IDS — A fruitful intrusion detection system for IoT," in *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, July 2016, pp. 440–448.

[26] A. K. Kyaw, Y. Chen, and J. Joseph, "Pi-IDS: Evaluation of open-source intrusion detection systems on Raspberry Pi 2," in *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, Nov 2015, pp. 165–170.

[27] D. Sung, "Smart home visions through the ages: The history of home automation," January 2018, accessed: 2018-08-14. [Online]. Available: https://www.the-ambient.com/features/visions-through-the-ages-history-of-home-automation-178

[28] N. King, "Smart home - a definition," September 2003, accessed: 2018-07-20. [Online]. Available: https://www.housinglin.org.uk/_assets/Resources/Housing/Housing_advice/Smart_Home_-_A_definition_September_2003.pdf

[29] SmartHomeUSA, "What is a smart home," accessed: 2018-08-14. [Online]. Available: https://www.smarthomeusa.com/smarthome/

[30] R. J. Robles and T.-h. Kim, "Applications, systems and methods in smart home technology: A review," *Int. Journal of Advanced Science And Technology*, vol. 15, 2010.

[31] G. Sandström, *Smart Homes and User Values Long-term evaluation of IT-services in Residential and Single Family Dwellings.* Stockholm: KTH, 2009.

[32] K. A. McKay, L. Bassham, M. S. Turan, and N. Mouha, *Report on lightweight cryptography.* US Department of Commerce, National Institute of Standards

and Technology, 2017.

[33] K. R. Fall and W. R. Stevens, *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[34] N. Vidgren, K. Haataja, J. L. Patiño-Andres, J. J. Ramírez-Sanchis, and P. Toivanen, "Security threats in Zigbee-enabled systems: Vulnerability evaluation, practical experiments, countermeasures, and lessons learned," in *2013 46th Hawaii International Conference on System Sciences*, Jan 2013, pp. 5132–5138.

[35] O. Olawumi, K. Haataja, M. Asikainen, N. Vidgren, and P. Toivanen, "Three practical attacks against Zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned," in *2014 14th International Conference on Hybrid Intelligent Systems*, Dec 2014, pp. 199–206.

[36] T. Zillner and S. Strobl, "Zigbee exploited – the good, the bad and the ugly," in *In Depth Security – Proceedings of the DeepSec Conferences*, vol. 2, 2016, pp. 699–704.

[37] Zigbee Alliance, "Zigbee 3.0," accessed: 2018-08-16. [Online]. Available: https://www.zigbee.org/zigbee-for-developers/zigbee-3-0/

[38] Z. Shelby, K. Hartke, and C. Bormann, *The Constrained Application Protocol (CoAP)*, IETF Std. RFC7252, June 2014.

[39] R. A. Light, "Mosquitto: server and client implementation of the MQTT protocol," *Journal of Open Source Software*, vol. 2, no. 13, 2017.

[40] C. Wilson, T. Hargreaves, and R. Hauxwell-Baldwin, "Smart homes and their users: a systematic analysis and key challenges," *Personal and Ubiquitous Computing*, vol. 19, no. 2, pp. 463–476, 2015.

[41] M. Pragnell, L. Spence, and R. Moore, *The market potential for Smart Homes*. YPS for the Joseph Rowntree Foundation York, November 2000.

[42] STATISTA, "Smart home," accessed: 2018-08-23. [Online]. Available: https://www.statista.com/outlook/279/100/smart-home/worldwide

[43] J. F. Coughlin, L. A. D'Ambrosio, B. Reimer, and M. R. Pratt, "Older adult perceptions of smart home technologies: implications for research, policy & market innovations in healthcare," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*. IEEE, 2007, pp. 1810–1815.

[44] A. McLean, "Ethical frontiers of ICT and older users: cultural, pragmatic and ethical issues," *Ethics and information technology*, vol. 13, no. 4, pp. 313–326, 2011.

[45] H. Rohracher, "The role of users in the social shaping of environmental tech-

nologies," *Innovation: the european journal of social science research*, vol. 16, no. 2, pp. 177–192, 2003.

[46] C. Tweed and G. Quigley, "The design and technological feasibility of home systems for the elderly," *The Queens University, Belfast*, 2000.

[47] Threat Analysis Group, LLC, "Threat, vulnerability, risk – commonly mixed up terms," accessed: 2018-07-20. [Online]. Available: https://www.threatanalysis.com/2010/05/03/threat-vulnerability-risk-commonly-mixed-up-terms/

[48] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. Fu, "Security vulnerabilities of internet of things: A case study of the smart plug system," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1899–1909, 2017.

[49] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart Nest thermostat: a smart spy in your home," in *Black Hat USA 2014*, 2014.

[50] X. Lei, G. Tu, A. X. Liu, C. Li, and T. Xie, "The insecurity of home digital voice assistants - Amazon Alexa as a case study," *CoRR*, vol. abs/1712.03327, 2017. [Online]. Available: http://arxiv.org/abs/1712.03327

[51] Z. Tang, Y. Zhao, L. Yang, S. Qi, D. Fang, X. Chen, X. Gong, and Z. Wang, "Exploiting wireless received signal strength indicators to detect evil-twin attacks in smart homes," *Mobile Information Systems*, vol. 2017, pp. 1–14, 2017.

[52] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, July 2017.

[53] A. Amin, I. ul Haq, and M. Nazir, "Two factor authentication," *International Journal of Computer Science and Mobile Computing*, vol. 6, pp. 5–8, July 2017.

[54] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," Internet Requests for Comments, RFC Editor, RFC 6347, January 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6347.txt

[55] S. Gerdes, O. Bergmann, C. Borrman, G. Selander, and L. Seitz, "Datagram transport layer security (DTLS) profile for authentication and authorization for constrained environments (ACE)," Internet Engineering Task Force, IETF, IETF Internet-Draft, June 2017. [Online]. Available: https://tools.ietf.org/id/draft-ietf-ace-dtls-authorize-00.html

[56] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, April 2015, pp. 746–751.

[57] P. R. Kennedy, T. G. Hall, and W. C. Yip, "Radio telecommunication device and method of authenticating a user with a voice authentication token," Jul. 4 2000, uS Patent 6,084,967.

[58] Nuance Communications, Inc, "Authenticate customers with biometrics," accessed: 2018-09-05. [Online]. Available: https://www.nuance.com/omni-channel-customer-engagement/security/identification-and-verification.html

[59] VoicePIN.com, "Voice authentication, voice biometrics, voice recognition - VoicePin," accessed: 2018-09-05. [Online]. Available: http://voicepin.com/

[60] Google, "Voice Match and media on Google Home," accessed: 2018-09-05. [Online]. Available: https://support.google.com/googlehome/answer/7342711?hl=en

[61] L. Zhang, S. Tan, and J. Yang, "Hearing your voice is not enough: An articulatory gesture based liveness detection for voice authentication," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 57–71.

[62] Home Assistant, "Home Assistant," accessed: 2017-12-11. [Online]. Available: https://home-assistant.io

[63] M. Murugan and D. H. Du, "Rejuvenator: A static wear leveling algorithm for nand flash memory with minimized overhead," in *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on*. IEEE, 2011, pp. 1–12.

[64] Radiotap, "Radiotap," accessed: 2018-09-11. [Online]. Available: https://www.radiotap.org/

[65] raspberrypi.org, "Raspberry Pi 3B+ specs and bencchmarks," accessed: 2018-07-13. [Online]. Available: https://www.raspberrypi.org/magpi/raspberry-pi-specs-benchmarks/

[66] "Meet the SmartThings Hub hardware," accessed: 2018-07-13. [Online]. Available: https://support.smartthings.com/hc/en-us/articles/205956900-Meet-the-SmartThings-Hub-hardware

[67] Panda Wireless, Inc., "Panda PAU06 user's manual v2.6," accessed: 2018-09-11. [Online]. Available: http://www.pandawireless.com/download/PWUsersManual_V2.6_14.10.40.pdf

[68] Clas Ohlsson AB, "WiFi smart plug with night light Clas Ohlson Home," 2018, accessed: 2018-09-07. [Online]. Available: https://images.clasohlson.com/medias/sys_master/9590387703838.pdf

[69] ASUSTeK Computer Inc, "RT-AC1750 specifications," NA, accessed: 2018-09-07. [Online]. Available: https://www.asus.com/us/Networking/RT-AC1750/specifications/

[70] Adafruit, "USB Charger Doctor," accessed: 2018-09-12. [Online]. Available: https://www.adafruit.com/product/1852

[71] "iostat - Linux man page," accessed: 2018-11-14. [Online]. Available: https://linux.die.net/man/1/iostat

[72] bwilkerson, "IP fragment overlapper," December 2008, accessed: 2018-10-29. [Online]. Available: http://www.pcapr.net/view/bwilkerson/2008/11/4/16/ip_fragment_overlapper.pcap.html

[73] ——, "IPv6 UDP flood," December 2008, accessed: 2018-10-29. [Online]. Available: http://www.pcapr.net/view/bwilkerson/2008/11/4/16/ipv6_udp_flood.pcap.html

[74] ——, "TCP SYN/FIN flood," December 2008, accessed: 2018-10-29. [Online]. Available: http://www.pcapr.net/view/bwilkerson/2008/11/4/16/tcp_syn_fin_flood.pcap.html

[75] ——, "UDP DNS flood," December 2008, accessed: 2018-10-29. [Online]. Available: http://www.pcapr.net/view/bwilkerson/2008/11/4/16/udp_dns_flood.pcap.html

[76] wingo, "Ping flood," August 2014, accessed: 2018-10-29. [Online]. Available: https://github.com/Igalia/pflua-bench/blob/master/savefiles/ping-flood.pcap

[77] pypacker, "Pypacker," accessed: 2018-10-29. [Online]. Available: https://gitlab.com/mike01/pypacker

[78] Kingston Technology Corporation, "Flash memory guide," 2015, accessed: 2018-10-29. [Online]. Available: https://media.kingston.com/pdfs/MKF_283.1_Flash_Memory_Guide_EN.pdf

[79] Vattenfall AB, "Rörligt elpris - Vattenfall," accessed: 2018-10-31. [Online]. Available: https://www.vattenfall.se/elavtal/elpriser/rorligt-elpris/

[80] Environmental Protection Agency (EPA), USA, "Greenhouse Gas Equivalences Calculator," accessed: 2018-10-31. [Online]. Available: https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator

[81] G. P. Spathoulas and S. K. Katsikas, "Enhancing IDS performance through comprehensive alert post-processing," *Computers /& Security*, vol. 37, pp. 176 – 196, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404813000503

[82] F. Mansmann, F. Fischer, D. A. Keim, and S. C. North, "Visual support for analyzing network traffic and intrusion detection events using TreeMap and Graph representations," in *Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology*, ser. CHiMiT '09. New York, NY, USA: ACM, 2009, pp. 3:19–3:28. [Online]. Available: http://doi.acm.org/10.1145/1641587.1641590

[83] Y. Livnat, J. Agutter, S. Moon, R. F. Erbacher, and S. Foresti, "A visualization

paradigm for network intrusion detection," in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, June 2005, pp. 92–99.