FH | JOANNEUM
University of Applied Sciences

Design of a Honeypot for Smart Home

Master's Thesis
submitted in conformity with the requirements for the degree of
Master of Science in Engineering (MSc)
Master's degree programme **IT & Mobile Security**

FH JOANNEUM (University of Applied Sciences), Kapfenberg

Supervisor: Martin Fruhmann, Bsc MSc
Submitted by: Markus Helmut Gollmann, BSc
Personal identifier: 2010419003

May 2022

Abstract

The popularity of Smart Homes has been increasing over the last few years and this trend still carries on. They offer great comfort while causing less struggle. This concurrence is definitely a treat for homeowners. The budget-friendly Internet of Things (IoT) offers a lot of devices which are commonly used in Smart Homes. Time and again, they do not provide a decent security level. This is the point where the problems might start off. Unnoticed by the homeowner, hackers might compromise the network; in the end things can even get worse and the homeowner finds himself being held to ransom.

Although, the number one risk associated with Smart Homes are hacker attacks, only few IT security mechanisms are applied to protect the Smart Home. In this thesis an experimental setup is established to test a simple and effective security mechanism, impressively preventing hacker attacks. This experiment can probably raise the awareness of hacker attacks. The sheer number of failed attacks happening during the experiment period of time will alert homeowners; in the event that a hacker attack still turns out successful, the homeowner gets immediately informed, which raises the overall security enormously. The mechanism that can effectively boost the network security is called "Honeypot". The aim of a honeypot is to reroute the attackers from their original target and monitor the attackers' actions. Subsequently, this information is used to redesign the security concept of the Smart Home. This rerouting is performed with simulated vulnerabilities which are easy to exploit.

This thesis evaluates the different design approaches of honeypots and analyses potential attack vectors for Smart Homes. The knowledge gained from the proof-of-concept implementation is then used for further improvements of the honeypot, so that the Smart Home Honeypot is capable of protecting a variety of different Smart Homes against a variety of attackers.

In the end the evaluation of the design concepts and the results from the experimental implementation together led to a design guideline for Smart Home Honeypots. The gained knowledge should be used to create a whole range of different honeypots that are able to protect all kinds of Smart Homes. At the same time, the setup and the maintenance should be kept simple and affordable. This way, a budget-friendly and effective security mechanism is created which can be installed by any Smart Home owner. If all precautions explained in this Master Thesis are adopted appropriate honeypots can effectively protect a Smart Home against virtually all kinds of possible attackers.

Zusammenfassung

Die Beliebtheit von Smart Homes hat in den letzten Jahren zugenommen und dieser Trend hält weiter an. Der von den NutzerInnen meistgeschätzte Mehrwert ist der große Komfort und deutlich verringerte Aufwand. Das budgetfreundliche Internet der Dinge (IoT) bietet eine Vielzahl von Geräten, die häufig in Smart Homes eingesetzt werden. Immer wieder bieten sie jedoch kein angemessenes Sicherheitsniveau und genau hier können die Probleme beginnen. Unbemerkt vom Hausbesitzer können Hacker das Netzwerk kompromittieren; am Ende kann es sogar noch schlimmer werden und der Hausbesitzer findet sich in der Situation wieder, Lösegeld zahlen zu müssen.

Obwohl das größte Risiko im Zusammenhang mit Smart Homes Hackerangriffe sind, werden nur wenige IT-Sicherheitsmechanismen zum Schutz von Smart Homes eingesetzt. In dieser Arbeit wird ein Versuchsaufbau erstellt, um einen einfachen und effektiven Sicherheitsmechanismus zu testen, der Hackerangriffe eindrucksvoll verhindern soll. Dieses Experiment schärft außerdem das Bewusstsein der NutzerInnen für Hackerangriffe. Die schiere Anzahl der fehlgeschlagenen Angriffe während des Versuchszeitraums wird die HausbesitzerInnen warnen; sollte sich ein Hackerangriff dennoch als erfolgreich erweisen, werden die HausbesitzerInnen sofort informiert, was die allgemeine Sicherheit enorm steigert. Der Mechanismus, der die Netzwerksicherheit effektiv erhöhen kann, wird "Honeypot"genannt. Das Ziel eines Honeypots ist es, die Angreifer von ihrem ursprünglichen Ziel umzuleiten und die Aktionen der Angreifer zu überwachen. Anschließend werden diese Informationen genutzt, um das Sicherheitskonzept des Smart Home neu zu gestalten. Diese Umleitung wird mit simulierten Schwachstellen durchgeführt, die sich leicht ausnutzen lassen.

In dieser Arbeit werden die verschiedenen Designansätze von Honeypots bewertet und potenzielle Angriffsvektoren für Smart Homes analysiert. Die aus der Proof-of-Concept-Implementierung gewonnenen Erkenntnisse werden dann für weitere Verbesserungen des Honeypots genutzt, so dass der Smart Home Honeypot in der Lage ist, eine Vielzahl von unterschiedlichen Smart Homes gegen eine Vielzahl von Angreifern zu schützen.

Die Auswertung der Designkonzepte und die Ergebnisse aus der experimentellen Umsetzung führten schließlich zu einer Design-Richtlinie für Smart Home Honeypots. Die gewonnenen Erkenntnisse sollen genutzt werden, um eine ganze Reihe von unterschiedlichen Honeypots zu erstellen, die in der Lage sind, alle Arten von Smart Homes zu schützen. Gleichzeitig sollten die Einrichtung und Wartung einfach und kostengünstig gehalten werden. Auf diese Weise wird ein budgetfreundlicher und effektiver Sicherheitsmechanismus geschaffen, der von jedem Smart-Home-Besitzer/Besitzerin installiert werden kann. Wenn alle in dieser Masterarbeit erläuterten Vorkehrungen getroffen werden, können geeignete Honeypots ein Smart Home effektiv gegen praktisch alle Arten von möglichen Angreifern schützen.

Acknowledgement

First and foremost, I would like to express my gratitude to my supervisor Martin Fruhmann, BSc MSc for his useful comments, remarks and engagement through the development of my master thesis.

My grateful thanks are further extended to Ao. Univ.-Prof. Mag.rer.nat. Dr.techn Johann Lang for proofreading my master thesis and for always having a sympathetic ear for me and my concerns.

Finally, I am also very grateful to my family, fellow students and my friends, who had such a positive impact on my master thesis.

THANK YOU!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The initial chapter presents the basic idea of this master thesis; at the same time, it tries to clarify the objectives. It starts with an explanation why additional security concepts should be considered while designing a Smart Home and it ends with a short comparison between a honeypot and other Information Technology (IT) security devices. Moreover, the different design aspects relevant for a well-suited honeypot are addressed.

1.1 Why do we need a honeypot?

Smart home equipment offers a lot of comfort in any apartment or house; it is little wonder that such gadgetry becomes more and more popular all over. Homeowners, however, are unaware of the huge number of threats looming on them. Based on a study from Statista, the number one risks associated with smart homes are hacker attacks. Nonetheless, only few additional defense mechanisms are applied. Along with the rising popularity of the internet of things (IoT) more and more private households are attacked by criminals.

In 2019 attackers were able to hijack the smart home installation of a couple living in Milwaukee, US. They installed a camera, a doorbell and a thermostat in order to feel safe and secure and to enjoy more comfort. Unfortunately, what they achieved was the complete opposite. One day a voice was talking to them via the camera while, all of a sudden, the thermostat temperature shifted to 90°F (32.2°C). Luckily, no other smart devices were connected; otherwise, the consequences might have been even more precarious. In the end the owners were deeply troubled that the 700$ installation obviously could be misemployed as an unauthorized entrance to their home (Sears, 2019).

Based on a forecast by the company Statista the number of smart homes in Austria will double up within the next few years from 0.81 million in 2020 to 1.69 million in 2025 (Statista, 2021). This is why it is important that cybersecurity be not only considered within enterprises but also with respect to private homes. Network security appliances such as firewall or intrusion detection/prevention (IDS/IPS) systems are fairly expensive and only very few private people will own one. Another downside is that these systems have to be updated regularly and the configuration is by no means trivial. In other words, there have to be simpler solutions of counteracting attacks.

One possible device that might be suitable for a small environment are so-called 'honeypots'. A honeypot is a specially crafted server that pretends to be a typical smart home device in order to attract attackers. The huge benefit of honeypots is that they are cheap and that they do not produce any false positive alerts as do other systems. The reason for that is that authorized users will not connect to the honeypot, because they are in the know of the real system. One disadvantage of a honeypot is that it does not work proactively. So, it can only detect an attack if the attacker is already inside the network and tries to connect to the honeypot.

Before a honeypot can be used it is pivotal to analyse possible attack vectors. Without knowing the risks it is impossible to design a working security concept. Not all attack vectors can be exploited right away. Therefore, it is important to figure out how current home automation products deal with these threats. This analysis will be used for a more specific honeypot configuration.

1.2 Objectives

The implementation and the configuration of a honeypot is of key importance. Otherwise, a honeypot might even reduce the level of IT security instead of increasing it. Theoretical knowledge of the individual components of a honeypot implementation is imperative to create an accurate and efficient honeypot. Therefore, the most important components and their intended purpose will be introduced gradually; the respective information will be provided step by step.

This thesis will be addressing the following pieces of research:

- What attack vectors do exist in smart homes and how can a honeypot be used to detect an attack?
- Which configuration of a honeypot is needed to reduce the chance of being detected by an attacker?

To enhance the configuration of a honeypot a detailed analysis of potential attack vectors will be carried out. This information will be needed for creating a more specific configuration and behaviour of the honeypot. Such a specific and tailored solution can provide a higher security level compared to a generic approach.

An analysis of state-of-the-art smart home implementation will be made. The focus will be on how vendors usually deal with the already discussed attack vectors. That way, the honeypot can be included into an overall security strategy.

1.3 Methodology

In this thesis qualitative methods and an experiment are used to answer the research questions. First off, the experiment with the self-designed proof of concept honeypot is being presented. This honeypot is then exposed to the internet and is used for data gathering from real attacks. Later on, this data is analysed in order to extract the attack vectors of smart homes. The knowledge gained from this analysis is determined for subsequently creating further improvements of the honeypot.

1.4 Thesis Outline

The first section of this thesis will prepare all theoretical knowledge. This theoretical foundation is needed to make sure that the following theories and guidelines are well intelligible. All parts of a honeypot implementation will be explained in detail. The next step will be to analyse possible attack vectors of a smart home and to look at existing home automation products and how these products are usually protected against unauthorized access.

Based on the theoretical knowledge and the aforementioned attack vectors a proof-of-concept implementation of a honeypot will be built. This honeypot will pretend to be a camera, a heating control and a file share and will be hosted on a Raspberry Pi platform. Finally, the conclusion of this master thesis will outline a configuration guideline for the implementation of a honeypot as well as provide a detailed analysis of potential risks and how vendors deal with them.

Chapter 2

Related Work

The book written by The Honeynet Project (2004) explains the methodologies used by the "black- hat" hacker community. The explanations of the applied tools and the trends are considered all along this thesis in the course of the design process of the Smart Home honeypot. Additionally, the expertise provided by the authors has been used during the analysis of the recorded attacks against the honeypots.

The authors Joshi & Sardana (2011) explain the concept of honeypots from different perspectives. Their case studies of practical implementations of honeypots have been used in this thesis to design a proper implementation within the Smart Home environment. In first part of the book the most important theoretical aspects of honeypots are being explained. This information could gratefully be used in this thesis to focus on the key properties of a Smart Home honeypot in order to optimize its efficiency.

In the master thesis of Rauter (2019) the effectiveness of different honeypot implementations were evaluated. The results of this thesis influenced the positioning as well as the alarming strategy of designed Smart Home honeypot. It also showed that a combination of different honeypots was successful in an industrial environment, which positively influenced the confidence for the experimental setup in the Smart Home environment.

In the conference paper written by B. Lingenfelter & Sengupta (2020) an analysis of botnet attacks against three different honeypots was performed. This analysis showed that most of the attacks against IoT devices are originating from botnets that try to further expand. The dominant botnet payload is still the Mirai worm. The attack analysis of this thesis extends the documented analysis by other components such as the web server. This is why similarities between the two analyses can certainly be

found, which implies that the trend of expanding botnets still carries on.

Although honeypots do not generate any false positive alerts, lots of other risks are associated with them if their implementation is flawed. The implementation guidelines written by Wendzel & Plötner (2007), were adapted for the use in a Smart Home environment. The experimental setup in this thesis has been reviewed based on the explanation of different risk analyses. This review ensured that a compromised honeypot within the experimental setup is not able to create any unwanted side effects.

A variety of different implementation options provides the required flexibility of a honeypot. If a honeypot always performs in the same way it would be too easy to be detected. The article written by H. Wafi & Bahaweres (2017) focuses on the integration of modern honeypots. Implementing wireless honeypots follows the trend of the increased number of wireless IoT devices. The knowledge about these modern honeypots has been extended with the previously mentioned sources in this thesis in order to create the modern self-designed Smart Home honeypot.

In the recent past, the number of IoT devices has increased exponentially which led to a gap between the security requirements and the implemented security features. In the article written by Iqbal et al. (2020) an in-depth analysis of the security requirements as well as a gap analysis were performed. This gap analysis has been used in the design process of Smart Home honeypots so that threats focusing on this gap can be identified and redirected to the honeypot for further analysis. This procedure helps to deal with the gap which, in turn, minimizes the risk of significant damage caused by this security gap.

The use of Artificial Intelligence (AI) in network security has been analysed by Chu & Song (2021). Although the implementation and training of AI might not be suitable for the Smart Home environment, the paper also showed key principles that need to be considered in an IoT network. These fundamentals could gratefully be used in this thesis throughout the design of the overall Smart Home security concept.

Chapter 3

Theoretical Background

This chapter provides the essential theoretical background of a Smart Home honeypot. It also contributes an in-depth explanation of the honeypot security concept as well as a description of the adopted hardware. Ultimately, a network protocol will be explained which will be needed to simulate one functionality of the target device. It continues with a focus on the overall IT security concept as it ought to be implemented in Smart Homes. Followed by an explanation of the most important network design aspects and best practice implementations.

3.1 Honeypots

Of course, at this point it is paramount to understand what honeypots are and what they are designed for. In the very basic form of a honeypot it is a simple computer server that pretends to be some different device. A honeypot does not try to hide itself, in fact it exposes itself as much as possible to attackers. So any attacker is instantly led to the idea that this 'server' is vulnerable and easy to exploit. An attacker will then focus on the honeypot and might ignore the rest of the network infrastructure. After all, the honeypot seems to be the easiest path first. This allows system administrators to analyse the attacker's conduct and adjust the security mechanism of the production servers accordingly. Based on the fact that no authorized user will connect to the honeypot, one of the major benefits of a honeypot is that it will not generate any false-positive alerts. In larger environments it might be possible to combine several individual honeypots into one 'honeynet'. These honeynets simulate not only a single device or a single service, they rather simulate a whole enterprise network. The simulated network distracts the attacker even longer, so that the administrators have more time for

their analysis. In case the attacker realizes that he is connected to a honeypot or a honeynet he will instantly stop any interactions with the systems and try to find the real production services (Joshi & Sardana, 2011).

One of the most important features in the design of a honeypot is that it has to be extremely difficult to detect its real identity. As mentioned above, an exposed honeypot is pointless. Whoever creates a honeypot ought to know the details of the device, service or network that the honeypot is meant to simulate. Moreover, the vulnerabilities presented to the attacker have to match conceivable vulnerabilities of the simulated device (Mukherjee, 2021).

A more detailed classification of honeypots based on their functionality is not worthwhile for the purpose in a small Smart Home environment and therefore will be skipped in this paper.

3.2 Hardware

The requirements for a honeypot in a Smart Home environment are completely different from honeypots in industrial enterprises. In Smart Homes it is important that a honeypot is very easy to set up and to maintain, also acquisition costs and maintenance costs play a major role during the conception phase. For small applications in a home environment a pocket computer called Raspberry Pi may be perfectly suited. The Raspberry Pi contains all you need printed on a single circuit board.

3.2.1 Costs

The acquisition price is reasonably low with the Raspberry Pi, and so is the maintenance cost. The latter can be calculated via the power consumption per hour. The maximum input current for the Raspberry Pi is 2.5 Amperes and its operating voltage is 5 Volt. The formula below calculates the power consumption per hour

$$\text{U [V]} \;\; \text{I [A]} \;\; \text{h [Hours]} = \text{P [Wh]} \tag{3.1}$$

$$5\text{V} \;\; 2.5\text{ A} \;\; 1\text{ h} = 12.5\text{ Wh} \tag{3.2}$$

Obviously, in order to arrive at the yearly power consumption the value (2.2) has to be multiplied by 8,760. Presently, the average price for one Kilowatt-hour (kWh) in Austria is 20 cents. The overall equation for the yearly maintenance charge is the

following

$$\text{P [kWh]} \;\; \text{h [Hours per Year]} \;\; \text{c [cost per kWh]} = \text{Costs [€]} \tag{3.3}$$

$$0.0125 \text{ kWh} \;\; 8{,}760 \text{ h} \;\; 0.2 \text{ €} = 21{,}9 \text{ €} \tag{3.4}$$

Assuming costs of 20 cents per kWh (E-Control, 2022) will lead to a maximum of 22 € for maintenance per year. Here we have to add that it is very unlikely that a honeypot in fact will constantly draw the maximum power of 12,5W.

3.2.2 Technical Details

Apart from the cost efficiency of the Raspberry Pi also the size of the board with it's dimensions of only 86 mm x 56 mm is suitable for a Smart Home, because it can be located anywhere.

In this master thesis a Raspberry Pi Model 3B+ is being used. The performance of this model is more than adequate for running a honeypot for a single device (Ltd, 2018). If several devices with different functionalities such as multiple cameras with individual video streams ought to be simulated, an upgrade to a more powerful hardware can well be considered. Otherwise, this model stands out by its performance to cost ratio. The following table shows the most important specifications.

Description	Spezifications
Processor	Broadcom BCM2837B0, Cortex-A53 (ARMv8) @ 1.4GHz
RAM	1GB LPDDR2 SDRAM
Wireless Interfaces	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac
Network Interface	Gigabit Ethernet over USB 2.0 (max. throughput 300 Mbps)
Power Consumption	5V/2.5A DC power

Table 3.1: Technical Details Raspberry Pi 3B+

3.3 Internet of Things

The key principle of Internet of Things (IoT) is that anything can be connected to anything and that these things share data between each other. The term IoT can be found in a variety of different patterns. Popular examples are automobiles which are equipped with sensors that automatically inform the workshop that the next service interval is reached, or a heart monitor implanted in a person's body independently contacting the ambulance if any inconsistency is recorded. In general, it can be said that any man-made object with an IP-address assigned, sharing data with other objects,

is called an ‘IoT device’. The share of data can be implemented in a variety of ways. The most common is the use of standard network communication where each device gets a unique IP-address assigned. As for Smart Homes, these smart things can be environmental sensors, IP cameras or any window contact providing the information about open or closed windows. Usually, these sensor data are exchanged with an IoT gateway which then controls the connected actors. Typically, Smart Home actors are the heating or cooling as well as the shading system. The behaviour of these systems is affected by the received sensor data. As an example, the shading system as well as the cooling can be routinely activated or stopped whenever a specific room temperature has been recorded. This smart interaction between the sensors and the actors, which both can be referred to as IoT devices, is the major benefit for homeowners and the fundamental concept of the Smart Home automation (Chu & Song, 2021).

IoT devices are often connected directly to the internet which increases the attack surface of the Smart Home or an enterprise significantly. The major concerns associated with IoT devices are privacy and IT security. Many a time it is not clear for the end user which data is gathered and sent to the IoT gateway. Due to the fact that all IoT devices in the Smart Home are closely connected, one exploitable vulnerability can be enough to take control over the Smart Home. The modified data sent by one sensor directly influences the other devices. The privacy of the IoT data should also be a matter of concern to the end user if public IoT gateways are used to connect the individual devices. The received data could potentially be used by companies who sell personal data of users (Iqbal et al., 2020).

A popular example showing the lack of implemented IT security in IoT devices is the successfully established Mirai botnet. A botnet is able to control one or more devices without the notice of their owner. This control can be used for a so-called Distributed Denial of Service (DDoS) attack, where one target is flooded with data from each member of the botnet, which results in an interruption of the service. The previously mentioned Mirai botnet was the largest IoT botnet ever recorded. In 2016 the about 145000 IoT devices were under control which resulted in a DDoS attack with a peak of 1 Tbps (B. Lingenfelter & Sengupta, 2020).

The key fundamental why it is possible to accomplish a remote control of the IoT devices is called remote code execution. The threat of remote code execution as well as the information about a real attack is explained in the Chapter 8.3.3 “Remote Code Execution”. This popular example of the lack of security was one of the impulses that eventually led to this master thesis. The idea behind a honeypot is the pressing need for a device that notifies a homeowner of an ongoing attack.

3.4 RTSP Protocol

The Real-Time Streaming Protocol (RTSP) was designed to transmit audio and video data between two endpoints with minimum latency. It was defined in the RFC2326 in 1998 and is nowadays the most frequently used protocol by IP cameras (Ruether, 2022). RTSP supports different commands, so that a client is not limited to viewing the stream, he or she is also capable of controlling the stream. The most important commands are listed below:

Command	Explanation
OPTIONS	The server respond with his capabilities of other commands
DESCRIBE	Retrieves the description of the media object
PLAY	Starts the media stream
PAUSE	Temporarily stops the media stream
SETUP	Specifies the transport mechanism to be used
TEARDOWN	Stops all media streams and terminates the session

Table 3.2: RTSP Protocol Commands

In this master thesis a simple RTSP server will be used to simulate the video stream from an original IP camera with the honeypot.

3.5 Network Security

Although the honeypot is a capable objective to enhance the security of a Smart Home, it is not recommended to solely rely on it. The security design of a Smart Home should consist out of multiple layers so even if attackers successfully bypasses one of the implemented security mechanisms, they still do not have full access to the Smart Home. One principle which is worth considering when designing the security strategy is the so-called "Defender's Dilemma". This means that the defender needs to protect all instances which could be used to compromise a network, whereas the attacker can focus on one piece of vulnerability to compromise a network. As a consequence, it is pivotal to have multiple layers because it is nearly impossible to create one single layer perfectly (Wendzel & Plötner, 2007).

The first and foremost layer in the security concept should be a network firewall. Although an enhanced configuration of a firewall is not trivial, even a simple configuration that only blocks external connection attempts can significantly increase the security level. Further protection features such as Intrusion Detection and Intrusion Prevention Systems (IDS/IPS) are nowadays implemented directly on the firewall and

do not need a further configuration to protect Smart Homes. An even more enhanced configuration of these systems is feasible, though the standard configuration is suitable for the protection purpose. Due to the automatic update features the maintenance effort is fairly low compared to the overall effect on the security (Wendzel & Plötner, 2007).

3.6 Network Segmentation

The next principle which is to be considered during the design phase of a network is the network segmentation. This means that the IT network should be split up into multiple segments. These segments allow granular control the network traffic between them: If the IP camera and the file share with private data are in different network segments, connections between them can be blocked. With this design an attacker is unable to access sensitive or private data from the file share even if he has managed to access the IP camera. This example convincingly shows that the separation of the smart home equipment from other network devices is highly recommended (Joshi & Sardana, 2011).

There are two procedures to split the network into different segments. The first one, which is not very practical, is to physically split the network. This means that there is no physical connection between the Smart Home equipment and the rest of the network. Each segment uses their own network devices such as switches or firewalls and has a separate connection to the internet. The second possibility is to use a so-called logical segmentation with Virtual Local Area Networks (Virtual Local Area Network (VLAN)s). With the use of VLANs the same network infrastructure can be used for the different network segments. This method can be easily implemented into existing networks without the need of separate cabling. This logical segmentation is performed on the network switch and the network firewall. A detailed explanation how VLANs work, is beyond the scope of this master thesis (ibid.).

The last part of the network design to be considered is the wireless network. It is highly recommended to use different Service Set Identifier (SSID)s for the IoT devices and the normal user devices. This way, also wireless Smart Home equipment is separated just like the wired equipment. The graphic below illustrates a possible segmentation of a Smart Home. In this example the red area represents the untrusted internet, the green area is the trusted network including the management of the network devices, the personal computers, a file share and of course the honeypot and the yellow area represents the IoT segment with the Smart Home equipment (H. Wafi & Bahaweres, 2017) .

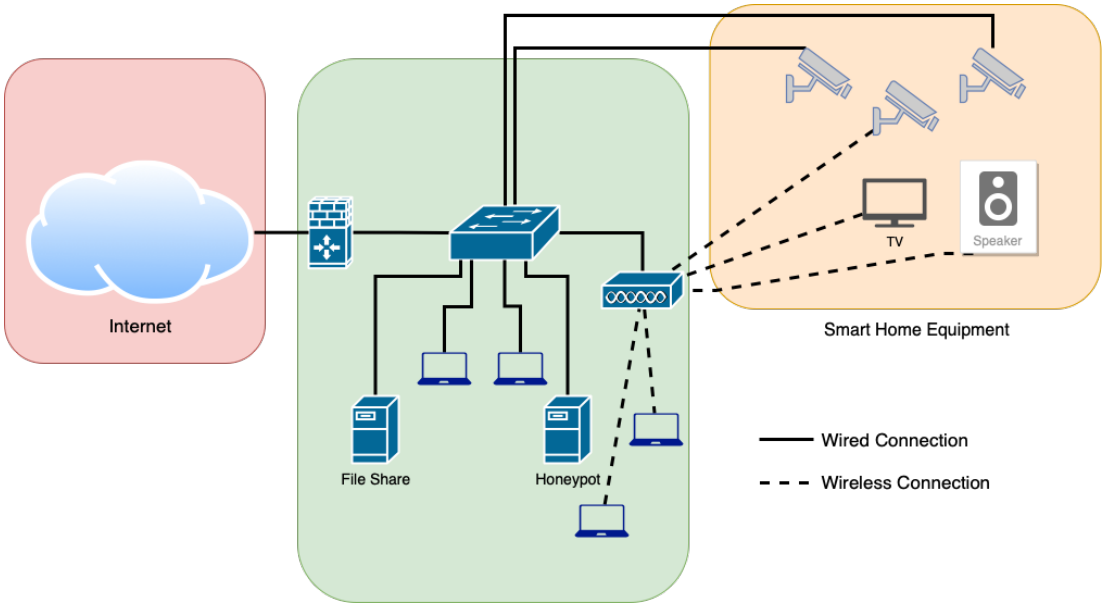

Figure 3.1: IT Network Segmentation

Commonly, personal computers use a well-tested operating system and additional security applications such as antivirus software. This means that the security level of these devices is higher than the security level of the IoT devices. The latter are most likely not as thoroughly tested and have to deal with limited resources, so additional security features might not be implemented. Therefore, a further segmentation between the notebooks and the file share is not necessary in a Smart Home network and this basic segmentation approach is sufficient.

The honeypot should be placed in the trusted network segment, so a connection to the honeypot indicates that all other security layers have failed. An attacker has gained full access to the network. In this situation the connection to the internet and the connection to the file share instantly has to be cut off (Wendzel & Plötner, 2007).

Chapter 4

Analysis of an IP Camera

As mentioned above, it is very important to thoroughly analyze the device that should be simulated. In this chapter let the device be an ordinary IP camera. The chapter covers the structure of the analysis as well as the explanation of the results that can be gathered using the different tools.

4.1 Procedure

The first step in every analysis is to get an overview of the different services which are offered from the target device. Typically, the tool Network Mapper (NMAP) is used to perform a so-called port scan. This scan shows all open network ports of the device and is also able to gather information about the running services e.g. software version numbers or the operating system. The next step during the analysis is to examine each service in detail. In this analysis the web server was analyzed without any special tool. The developer tools in the Google Chrome browser provided enough information in order to rebuild the web interface of the Internet Protocol (IP) camera. A detailed explanation of the initial scan as well as an explanation of the web server will be covered in the following two sections.

4.2 NMAP Results

NMAP is a very powerful tool for network analysis. It is not only capable of identifying open ports but also the operating system, the service version and the used protocol. For the initial scan the top 1000 Transmission Control Protocol (TCP) ports were scanned to get a first overview of the device. In order to get more information the following

additional parameters were used:

- -sC to execute the default scripts of NMAP
- -sV to determine the service/version information of open ports
- -oA to save the result for further investigation without the need of re-scanning the device

The image below shows the complete command as well as the results for the two open ports which were discovered.

```
└─# nmap -sC -sV -oA kamtron_IP-Camera_initial_scan 192.168.178.67
Starting Nmap 7.91 ( https://nmap.org ) at 2022-01-23 09:12 EST
Nmap scan report for 1jfiegbrcrfka_eth0 (192.168.178.67)
Host is up (0.0043s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE  VERSION
80/tcp   open  http?
|_http-title: IPC
8600/tcp open  asterix?
```

Figure 4.1: Initial NMAP Scan

The first open TCP port is port 80 is allowing connections to the built in web server using the Hypertext Transfer Protocol (HTTP). The further analysis of the web server will be explained in the next section.

The second network service listens to port 8600 which is used for streaming the video and audio signal. Based on the analysis performed by the tool this service could be identified as a RTSP stream. Details of the RTSP protocol have already been explained in Chapter 3.4 “RTSP Protocol”.

4.3 Web Server Results

The web server can be accessed with a web browser. The web page which can be used for changing administrative settings consists of java script application. It is worth mentioning that a lot of errors occur by accessing the web page, because resources (e.g. images) are missing on the device and therefore the browser is not able to load them. The image below shows the login screen which is visible after successfully connecting to the server.

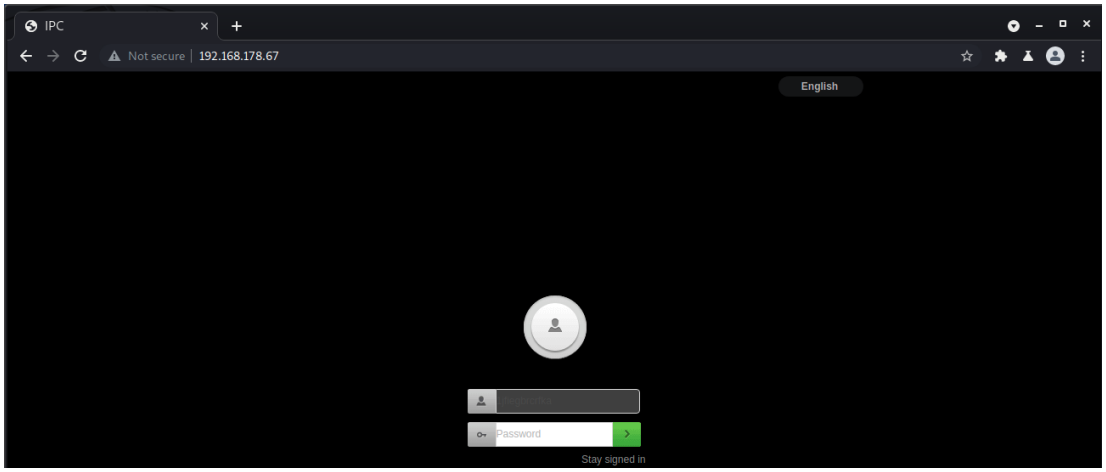

Figure 4.2: Login Screen

The java script application includes a lot of errors. This does not only relate to the missing resources that have already been mentioned earlier; the lack of proper session handling significantly tarnish the user experience. A well designed and implemented session handling allows the user to visit the web page multiple times with a single login. Typically, sessions can be terminated either by the user itself (e.g. performing a logout) or a session could be terminated by an idle timeout. For this web application the developers decided to terminate the session immediately when the page is loaded. This means that every page reload needs a new login, because there is no existing session anymore which could authenticate the user. After the successful authentication the management panel is visible. Administrative settings such as the network connection or the password can be modified. The image below displays the web interface. A detailed explanation of the settings will be provided later on, in the course of the development of the honeypot.

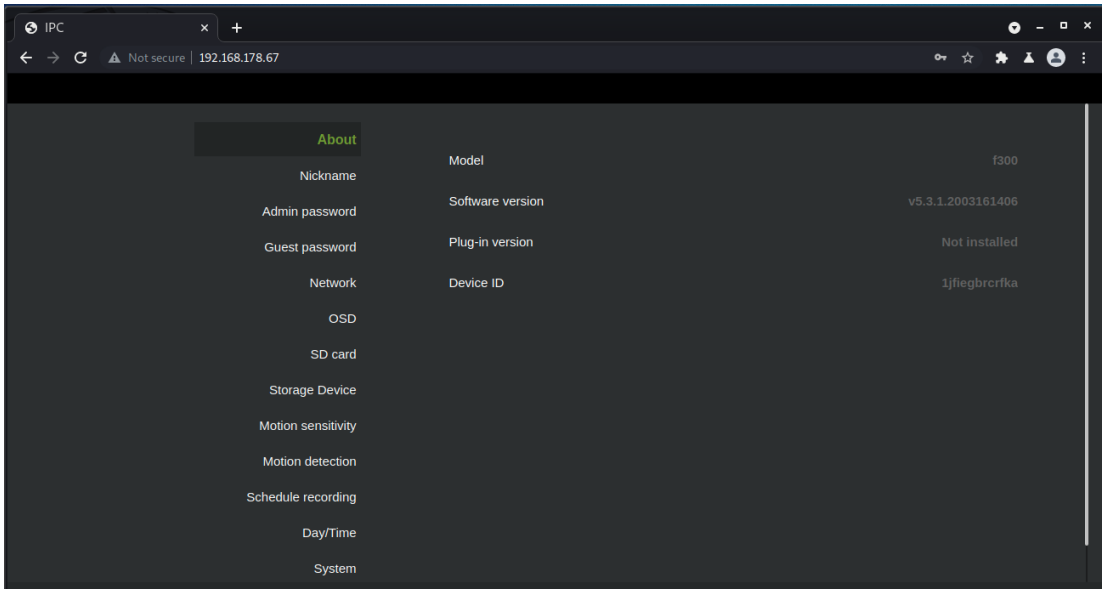

Figure 4.3: Management Interface

A detailed explanation of the settings will be provided later on, in the course of the development of the honeypot.

4.4 Privacy related findings

During the analysis of the network traffic generated by the IP camera, various packets to a public cloud were detected. Further investigations lead to the result that it is possible to access the camera through a web interface on the website https://mipcm.com. Each camera has a unique device ID which is also used as a user name cannot be changed. For the user respectively the home owner it is only possible to set a password for the device. During the setup process this connection is never mentioned, so it is very likely that home owners use a weak password as they are not aware of the risk of a remote access. Therefore, the investigated IP camera can be easily misused as a spyware.

Chapter 5

Software Components

This chapter includes a record of all necessary packages that have to be installed in order to successfully operate the honeypot. It starts off with a basic installation of the Raspberry Pi itself and moves on to the individual packages. For a detailed understanding of this chapter some basic knowledge of the Linux operating system such as changing passwords or adding a new user is a prerequisite.

5.1 Installation Raspberry Pi

The Raspberry Pi is designed to support a variety of different operating system. There are three different versions of the Raspberry Pi OS available as well as other third-party operating systems such as Ubuntu or TLXOS. The difference between the three available versions is the number of additional features which are already included in the image. In this master thesis the decision was made to use the Raspberry Pi OS in the lite version. The light version is smallest image and only includes the essentials software packages and no graphical desktop software, which means that the Raspberry Pi can only be accessed via the command line. The benefit of the lite image is the reduced size of the image, it only amounts to 463 MB large, whereas the Raspberry Pi OS with desktop and recommended software is around 3 GB large. The additional software packages required run the honeypot will be explained in the following chapters (Ltd, 2018).

For installing the Raspberry Pi OS Lite, the image can be downloaded from the official website. The next step is to transfer the image to a micro SD card and power on the Raspberry Pi. It will automatically boot from the Secure Digital (SD) card and is ready to use. It is recommended to change the default credentials of the standard

user "Pi" immediately as well as to allow SSH connections to facilitate remote logins; no monitor or keyboard is required. It is paramount to choose a strong and reliable password for all users, otherwise attackers will actively connect to the Raspberry Pi and compromise the whole operation.

5.2 Installation Apache, MariaDB

The basic installation being performed, the first two packages needed are a web server and database server. These two are needed for the simulation of web interfaces as well as for storing data gathered during an attack. This data is pivotal for a subsequent analysis to understand the anatomy of an attack.

The Rasperry Pi OS is based on a Debian Linux, which means that the basic packet manager is already installed. This packet manager will be used for installing all of the additional packages. The major benefit of the packet manager is, that it takes care of all dependencies a package might have. Otherwise, these would have to be resolved manually, which might be a very time-consuming process.

The Apache web server was chosen in this master thesis as it is very simple to configure and one of the most frequently used web servers. MariaDB was chosen as the database server, because it is the open-source version of the MySQL server and therefore free of charge.

The two software packages are installed with the following two commands:

```
root@raspberrypi:~/$ apt-get install apache2
root@raspberrypi:~/$ apt-get install mariadb-server
```

After the installation of the mariadb-server it is recommended to execute the secure installation script which is included in the installation with the following command.

```
root@raspberrypi:~/$ mysql_secure_installation
```

This script increases the security of the installed database server by automatically performing the following tasks:

- Setting the password for the root user
- Switching to the unix_socket authentication
- Removing the anonymous user
- Disabling the remote login for the root user

- Removing the default database "test"

5.3 Install PIP3

The next program that has to be executed in the of the honeypot setup is the python3 package installer pip. This program installs the additional python3 packages such as the requests package which is needed for the communication with external web servers. PIP3 can easily be installed using the following command:

```
root@raspberrypi:~/$ apt-get install python3-pip
```

The following python3 packages are essential in order to successfully run the honeypot:

```
root@raspberrypi:~/$ pip3 install requests
root@raspberrypi:~/$ pip3 install telegram_send
root@raspberrypi:~/$ pip3 install mariadb
root@raspberrypi:~/$ sudo apt-get install python3-opencv
```

5.4 Install Port Knocking

Port knocking describes a technique for hiding services. A service is only available for the user if a specific pattern of connection attempts is performed. To give an example: port 22 for the SSH service is only open if the user previously tried to connect to the ports 1001,1002,1003, otherwise the port is closed and no SSH connection can be established. Port knocking can be used as an additional security feature, because without knowing the exact pattern no connection to a service can be established. As for a honeypot this technique is used to hide the SSH port, so a possible attacker does not recognize a difference between the target device and the honeypot during a port scan. The port knocking daemon can be easily installed with the following command:

```
root@raspberrypi:~/$ apt-get install knockd
```

5.5 Install MAC Changer

The next software utility that is needed to hide the honeypot is called Media Access Control (MAC) changer. This utility allows to change the MAC address of network interfaces to a specific ad- dress or to a random address. In this use case we go for a specific address; this way, the manufacturer of the honeypot will not be disclosed.

A MAC address is typically a hardware bound address which originally could not be changed. This address consists of 48 bits, whereas the first 24 bits are a unique vendor identifier and the remaining bits are used for consecutive numbering. The goal is to change the MAC address so that the vendor identifier matches with the target device. The installation and the configuration of the MAC changer can be achieved using the following two commands:

```
user@raspberrypi:~/$ apt install macchanger
user@raspberrypi:~/$ macchanger -m=ae:ca:06:22:89:55 eth0
```

The configuration command takes two parameters. The first one starting with `-m` is the new MAC address of the interface and the second parameter is the name of the interface which is meant to be changed. The original MAC address can simply be restored with the parameter `-p`. Please note that this change is only temporary, if the honeypot reboots the change has to be re-performed. Therefore, it is advisable to automatically perform it whenever the boot process is finished.

5.6 RTSP-Simple Server

The last software module that needs to be installed is a simple RTSP server. This server will be used to stream the video of the simulated IP camera. Due to the complexity of such a server an external project called "RTSP Simple Server" developed from "aler9" is used. The pre-compiled binary of the server can be downloaded from code repository and then started from the command line. In order to mimic the target camera as close as possible a few additional configurations were applied. All these configurations can be done by editing the supplied configuration file `rtsp-simple-server.yml`. The first one is to change the RTSP port from 8554 to 8600. Next other supported protocols such as RTMP or HLS were disabled (Aler9, 2022).

Chapter 6

Implementation Honeypot

In this chapter additional configuration steps of the different previously installed software modules will be explained. These steps are needed to successfully start and operate the smart home honeypot. This chapter also includes a basic explanation of the implemented detection and reporting mechanism.

6.1 Basic Concepts and Detection Mechanism

Honeypots are a reactive tool by design, which means that they are not designed to proactively inform about potential upcoming attacks; this would be the job of vulnerability scanners. A honeypot is a passive device waiting for an attacker to establish a connection before generating an alarm. In the design of the smart home honeypot the log files of the Apache web server play a major role in the detection process of an attack. The access log file will be read by the honeypot process in regular intervals and if a new entry is detected a notification will be sent to the home owner. A message in the Telegram messenger will be used to inform the home owner about the newly established connection. Based on the fact that an authorized user or device will never establish a connection to the honeypot, the home owner can be sure that a malicious activity has taken place.

The detection mechanism and the alarming mechanism are programmed in modules. This simplifies further development, because the alarming process can be re-used even if the design of detection mechanism changes or additional mechanisms are included. One possible enhancement of the detection mechanism would be to also monitor the SSH logins and look for unused usernames to detect connection attempts to the Raspberry Pi itself. Login attempts with non-existing usernames are an indication for brute

force attacks where the most common username/password combinations are tried out.

6.2 Configuration Apache + MariaDB

Once the basic configuration and installation tasks (see Chapter 4 Installation Fundamentals) are performed, there are a few additional steps to be taken before the honeypot is ready. The first adaption which has to be made to the Apache web server is a change of permissions to the access log which is written by the server. This file contains information about every connection which has been established between a client and a server. This file will regularly be parsed by a python script; a procedure which is usually repeated every 30 seconds. In case of irregularities the home owner will be alarmed. Obviously, it is necessary that this file can be read by another process. The default permission of the file is 'read and write' for the user 'root' and 'read' for members of group 'adm'. Other users have no permissions whatsoever to the file. The permissions have to be set in the settings of the log rotation deamon. However, changing the permission of the file itself is not sufficient as a new access log file is being created. The image below shows the log rotation settings of the Apache web server.

```
root@raspberrypi:/etc/logrotate.d# cat apache2
/var/log/apache2/*.log {
        daily
        missingok
        rotate 14
        compress
        delaycompress
        notifempty
        create 644 root adm
        sharedscripts
        postrotate
                if invoke-rc.d apache2 status > /dev/null 2>&1; then \
                    invoke-rc.d apache2 reload > /dev/null 2>&1; \
                fi;
        endscript
        prerotate
                if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
                        run-parts /etc/logrotate.d/httpd-prerotate; \
                fi; \
        endscript
}
root@raspberrypi:/etc/logrotate.d#
```

Figure 6.1: Apache log rotate settings

By changing the numerical permission to 644 (read/write for the owner, read for the group and read for all others) instead of 640 read permissions to all other users are granted. The image below shows the changed permissions to the access log files. All

other log files such as the error log have the default permission, without any read access for other users.

```
root@raspberrypi:/var/log/apache2# ls -l
total 32
-rw-r--r-- 1 root adm  232 Nov 19 07:21 access.log
-rw-r--r-- 1 root adm 1730 Nov 18 13:32 access.log.1
-rw-r----- 1 root adm  269 Nov 19 00:00 error.log
-rw-r----- 1 root adm 1227 Nov 19 00:00 error.log.1
-rw-r----- 1 root adm  359 Nov 18 00:00 error.log.2.gz
-rw-r----- 1 root adm  356 Nov 17 00:00 error.log.3.gz
-rw-r----- 1 root adm  358 Nov 16 00:00 error.log.4.gz
-rw-r----- 1 root adm  359 Nov 15 00:00 error.log.5.gz
-rw-r----- 1 root adm    0 Nov 13 10:33 other_vhosts_access.log
root@raspberrypi:/var/log/apache2#
```

Figure 6.2: Apache log files

6.3 Configuration of the alarming process

During the design process of the honeypot the decision was made to use the Telegram messenger for alarming the home owner. Telegram provides an Application Programming Interface (API) which can be used to automatically send messages. Apart from this, a python3 package is available handling the communication between the honeypot and the Telegram servers. The installation of the package has already been covered in the previous chapter. The package needs to be configured once, following the commands listed below:

```
markus@raspberrypi:~ $ /home/markus/.local/bin/telegram-send --configure
Talk with the BotFather on Telegram (https://telegram.me/BotFather), create a bot and insert the token
> [redacted]
Connected with SmartHomeHoney_bot.

Please add SmartHomeHoney_bot on Telegram (https://telegram.me/SmartHomeHoney_bot)
and send it the password: [redacted]

Congratulations markus_aut97!
telegram-send is now ready for use!
markus@raspberrypi:~ $
```

Figure 6.3: Telegram Send configuration

The creation of the telegram bot itself will be covered in the next section. It goes without saying that the password and the token itself should not be shared with anyone, because everybody who knows the token of the bot can send messages on behalf of the bot. After the successful configuration of the python3 package a message can be easily sent using the following code:

```
1 import telegram_send
2
3 telegram_send.send(messages=["*ALERT*:\nConnection found from IP:{ip}".
    ‹! format(ip=ip)],parse_mode="Markdown")
```

Listing 6.1: Code snippet for sending the alert message

The function `telegram_send.send()` takes an array of messages, so it is possible to send multiple messages in one go. The parameter parse_mode can be used for further formatting of the message. The screenshot below shows what the message looks like on the mobile phone of the home owner.

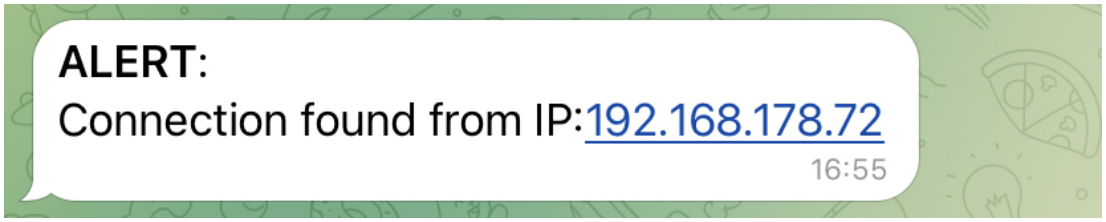

Figure 6.4: Telegram Message

6.4 Creating a Telegram Bot

In this section the process of creating a Telegram bot will be explained. This bot will be used for sending the alarm messages. For creating or editing an existing bot the Telegram service called "BotFather" has to be contacted. With the command `/start` sent to the BotFather, a reply with all possible commands will be received. The image below shows this process.

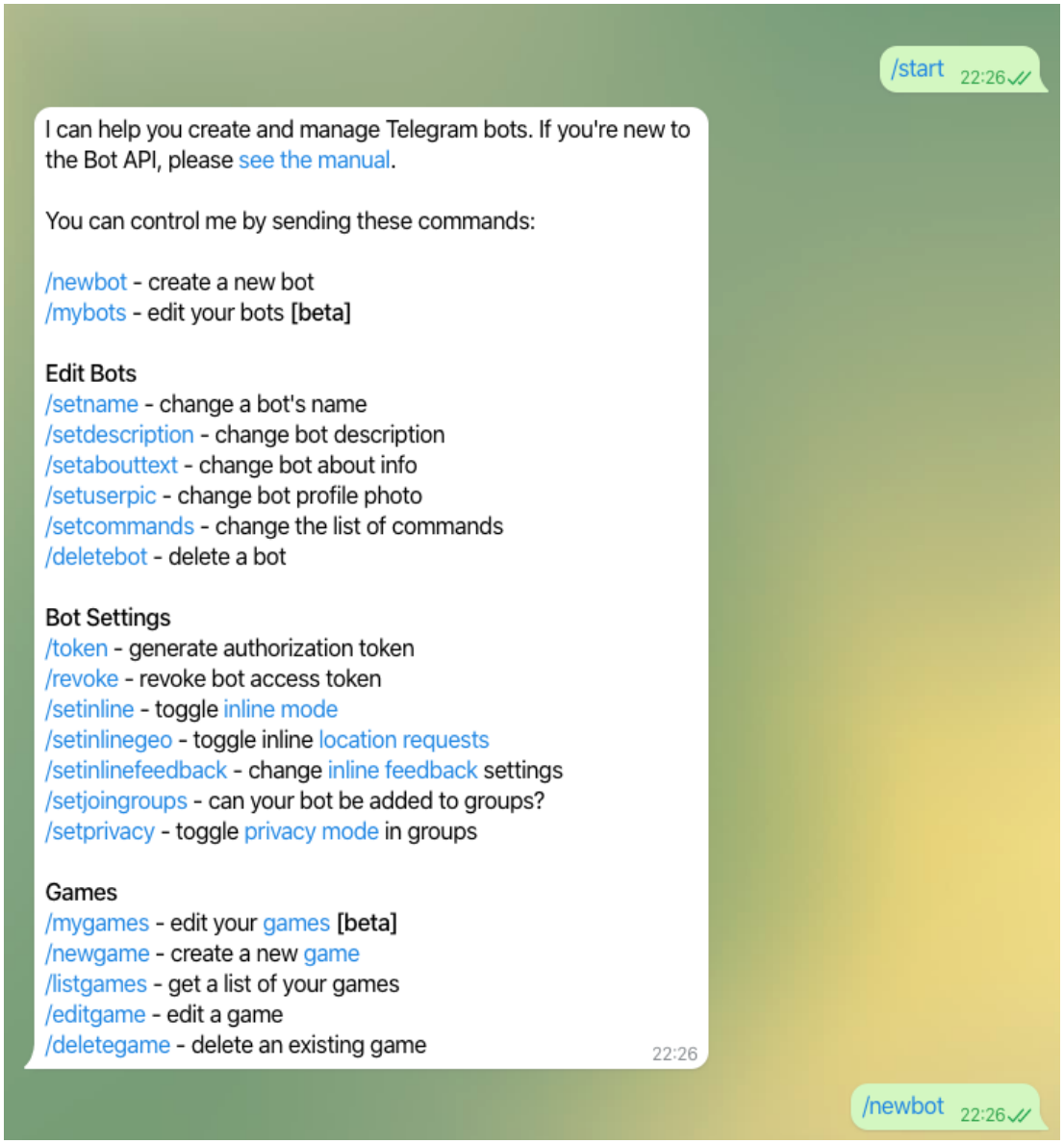

Figure 6.5: Creating a Telegram Bot Part 1

Next the command `/newbot` is needed to create a new bot. For this master thesis the name "SmartHomeHoney" has been chosen, but it can be any name that has not been chosen before by any other Telegram Bot. Once the bot has been successfully created, the API key is included in the reply from the BotFather. This token is then needed for configuring the telegram_send python package.

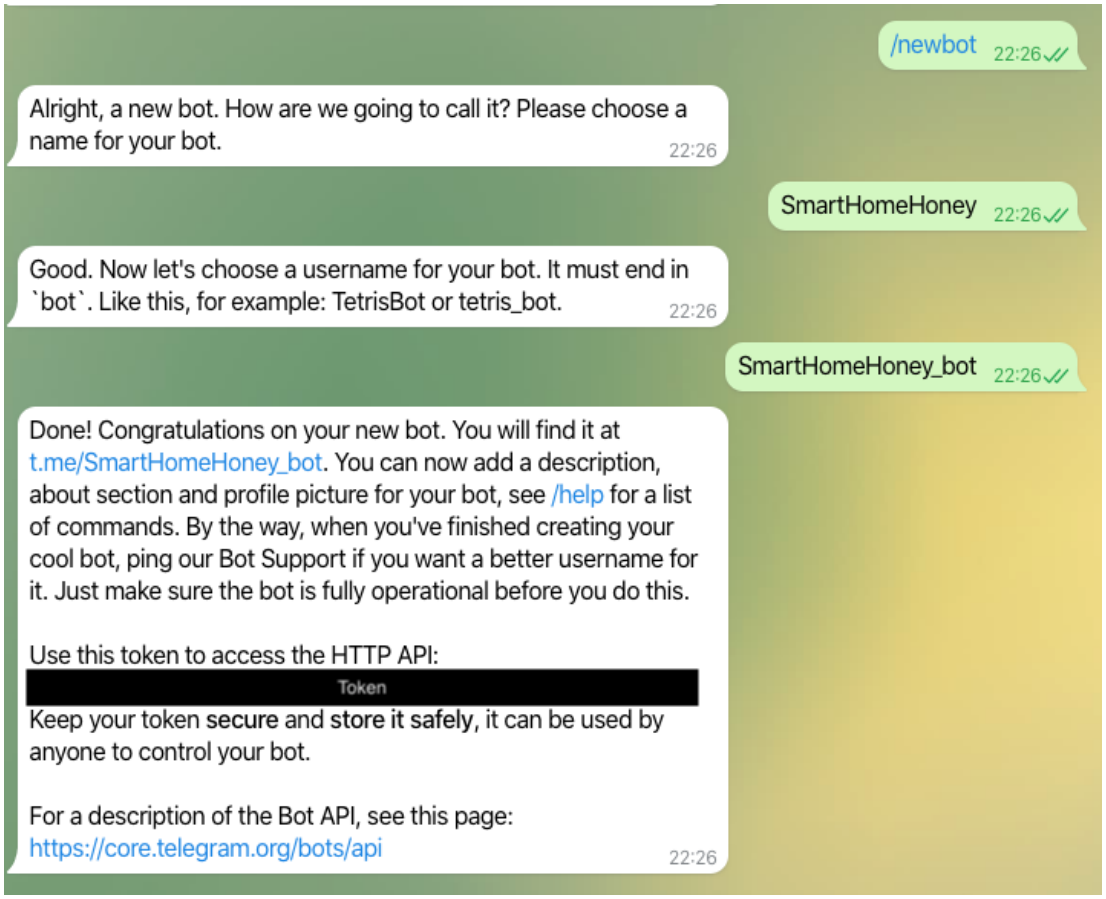

Figure 6.6: Creating a Telegram Bot Part 2

6.5 Cron jobs

The last section of this chapter focuses on the configuration of the cron jobs. Cron is a time-based scheduler that can be used to periodically execute tasks. The smart home honeypot uses this built-in Unix feature to periodically check the access log file on the Apache web server for new connections. The following command allows to edit the cron jobs of the user:

```
user@raspberrypi:~/$ crontab -e
```

The structure of the Linux crontab consists out of six columns. In the following table each column of the crontab is contained in one row:

Crontab column	Name	Explanation
m	Minute	A value in this field means that the command will be executed in the specified minute of the hour. e.g. 10 means 10th minute of the hour
h	Hour	This field specifies the hour of the day in a 24 hour format. e.g. 09 means 9am
dom	Day of Month	Executes the command only at a specific day of the month. e.g. 15 means on the 15th day
mon	Month	Specifies the month in numeric value 1 to 12
dow	Day of Week	Specifies a specific day of the week. This value ranges from 0 to 6
command	Command	Path to the command which should be executed.

Table 6.1: Crontab explaination

An * in any of the columns means that it is executed every time. E.g. * in the HOUR field means that the command is executed every hour and not only at a specific hour. The smallest interval for executing a command is one minute. In order to reduce the detection time of an attack two cron jobs are needed for the honeypot. To list all cron jobs for a specific user the command `crontab -l` can be applied. The image below shows the configuration which is recommended for the smart home honeypot.

```
markus@raspberrypi:~ $ crontab -l
# m h  dom mon dow   command
* * * * * /home/markus/SmartHomeHoneyPot/python/log_parser.py
* * * * * sleep 30; /home/markus/SmartHomeHoneyPot/python/log_parser.py
```

Figure 6.7: Crontab

As can be seen, both commands use the asterisk (*) so that they are executed every minute, on each day of the year. The second command employs a sleep 30; it instructs the system to wait for 30 seconds before the log parser script is executed. With this trick it is possible that the script "log_parser.py" is executed every 30 seconds. This guarantees that the home owner gets notified within a maximum time of 30 seconds whenever a new connection is being established. If a even smaller reaction time is needed it would be possible to create four cron jobs and adjust the sleep time to 15, 30, 45 seconds, so that the script is executed every 15 seconds instead of every 30 seconds.

6.6 Configure Port knocking

The configuration of the knockd service can be made by editing the configuration file located at `/etc/knockd.conf`. In this file the sequence for opening and closing a

port can be configured. It is very important that this sequence be kept secret; otherwise anybody would be able to open the port to the service. Therefore, it is highly recommended to change the initial sequence, because it is publicly available. The image below shows the example configuration which has been used in this master thesis.

```
[openSSH]
        sequence    = 123,1234,12345
        seq_timeout = 5
        command     = /sbin/iptables -A INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn

[closeSSH]
        sequence    = 12345,1234,123
        seq_timeout = 5
        command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 22 -j ACCEPT
        tcpflags    = syn
```

Figure 6.8: Knockd configuration

The knocking sequence for opening the SSH port 22 is set to the TCP ports 123, 1234 and 12345. Connection attempts to these ports have to be made within five seconds in order to successfully open the port. The closing sequence is configured with the same ports, but in reversed order.

6.7 Automated Honeypot Deployment

As described in the previous chapters the process of analyzing a Smart Home device and programming a honeypot is a pretty time-consuming process and does not provide good scalability. On the other hand, an automated deployment of a more generic honeypot is not as accurate as an individually designed honeypot. This is why it has to be well evaluated if such an ambiguity can be tolerated or not. The advantage of an automated honeypot is, that the deployment is very fast, and it can be easily deployed in multiple network segments, therefore this deployment method is often used in larger networks (Rauter, 2019).

One framework which can be used for an easy deployment is called "OpenCanary". This framework allows the user to configure multiple services where each service is an individual honeypot. The list below shows a subset of the supported services which can be useful for Smart Homes. These services can be configured in simple configuration files which are then read by OpenCanary (Canary, 2022).

- SSH: a Secure Shell server alerting on login attempts
- File Transfer Protocol (FTP): a File Transfer Protocol server alterting on login attempts

- GIT: a Git protocol alerting on repository cloning
- HTTP: an HTTP web server alerting on login attempts
- MySQL: a MySQL server altering on login attempts

To compare the generic honeypot with the individual Smart Home honeypot a default configuration of the HTTP honeypot service is listed below. The OpenCanary services only provide a login prompt and alerts are triggered if an attacker tries to login to the service. There are no further web pages available. This HTTP service provides a basic Hypertext Markup Language (HTML) login page or a login form for a Synology Network Attached Storage (NAS) out of the box. Another login screen can be configured manually (Canary, 2022).

```
{
    "http.banner": "Apache/2.2.22 (Ubuntu)",
    "http.enabled": true,
    "http.port": 80,
    "http.skin": "nasLogin",
    "http.skin.list": [
        {
            "desc": "Plain HTML Login",
            "name": "basicLogin"
        },
        {
            "desc": "Synology NAS Login",
            "name": "nasLogin"
        }
    ],
}
```

One of the major differences between the generic HTTP honeypot and the self-designed one, is the number of features supported. The generic honeypot will be identified immediately after the login because there are no further features available. The self-designed honeypot will not be detected as easily as it supports all features from the real device.

Chapter 7

Results

This chapter provides a detailed overview of the implemented features of the honeypot and a comparison between the honeypot and the original IP surveillance camera. It starts off with an explanation of the simulated web service followed by an explanation of the video stream. The comparison of the two devices will track the anatomy of a typical attack.

7.1 Impressions from the Proof of Concept Honeypot

In this first part the implemented features will be explained one by one. The first one of these features is the web service. A short explanation of the web service as already been provided in Chapter 4.3 “Web Server Results”. An in-depth analysis of the web application reveals that it consists of two parts. The first one is a client side JavaScript application communicating with the back-end system. The image below shows some of the requests between the front-end and the back-end. The marked request is used to get the first basic information from the device such as the username or the nickname.

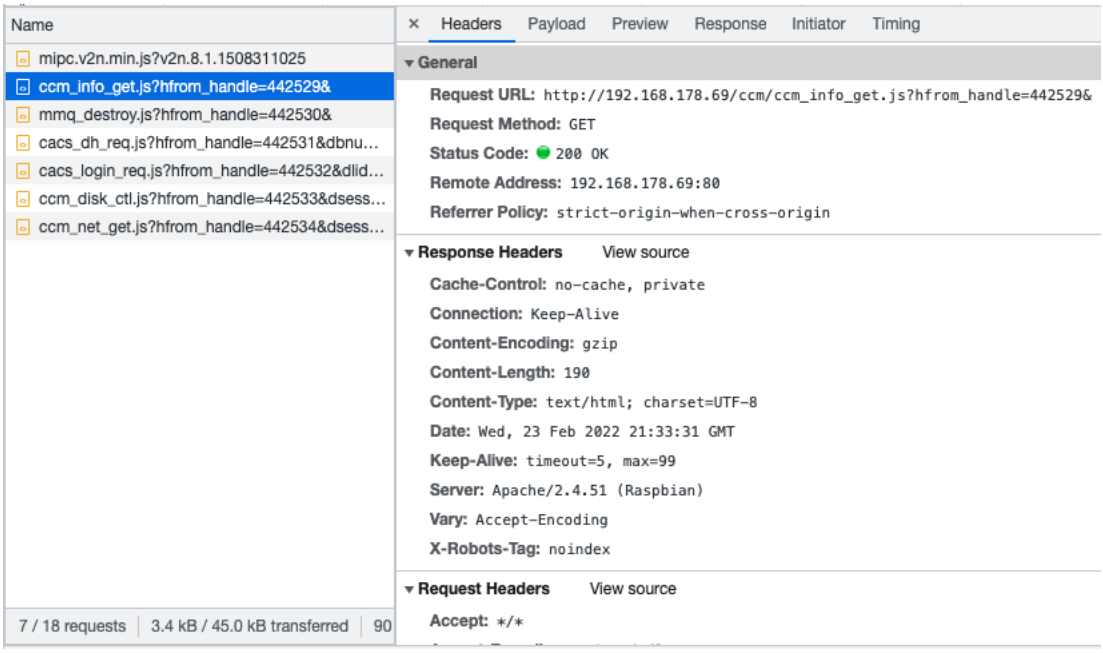

Figure 7.1: Web service Requests

Due to the fact that the JavaScript application has been minified it would be very complicated and time-consuming to reverse engineer the application in order to build a identical version for the honeypot. This is why the decision was made to use the original front-end application and only create a back-end that responds to the requests from the front-end. One of the challenges by simulating the back-end was that each request from the front-end contains a unique number. This number is generated by the JavaScript application and has to be included in the response from the back-end otherwise the application cannot handle the response and will halt. The back-end has been created using the programming language recursive initialism PHP: Hypertext Preprocessor (PHP) and the framework Symfony. This framework is capable of defining multiple so-called controllers where each controller listens to a specific path. The code snippet below shows such a controller together with the configured path for this controller.

```
1 /**
2 * @Route("/ccm/ccm_info_get.js")
3 */
4 public function ccm_info(): Response
5 {
6     $response = 'message({type:"ccm_info_get_ack",from:542113792,
      ↪ from_handle:859,to_handle:'.$_GET["hfrom_handle"].',data:{ret:{
      ↪ code:"",sub:"",reason:"",desc:""},type:"IPC",sn:"1jfiegbrcrfka",
      ↪ nick:"MasterThesis",ver:"v5.3.1.2003161406",spv:"v1"}});';
7     return new Response($response);
8 }
```

Listing 7.1: Code snippet of the info controller

This example shows the back-end implementation of the previously mentioned info request. This controller-based design provides three major benefits. The first one is, that it is easily possible to extract data from the request e.g. get the unique number with `$_GET["hfrom_handle"]` so that the front-end is able to parse the request. The second advantage is that the request can be stored in a database for further analysis of the attacker's demeanor. The code below shows the implementation of the login controller storing the entered password, the username and the respective time stamp in the local database.

```
1 /**
2 * @Route("/ccm/cacs_login_req.js")
3 */
4 public function login_req(ManagerRegistry $doctrine): Response
5 {
6     $entityManager = $doctrine->getManager();
7     $kamtron = new Kamtron();
8     $kamtron->setRequest('pass:'.$_GET["dpass"].'|duser:'.$_GET["duser"]);
9     $kamtron->setTime(date("Y-m-d H:i:s"));
10    $kamtron->setFile("cacs_login_req.js");
11    $entityManager->persist($kamtron);
12    $entityManager->flush();
13    $response = 'message({type:"cacs_login_ack",from:542113792,from_handle
      ↪ :1155,to_handle:'.$_GET["hfrom_handle"].',data:{result:"",sid:"0
      ↪ x2",seq:1089,addr:"192.168.178.65",guest:0,vreq:"",uid:"0x0",lid
      ↪ :"0x0",lkey:""}})';
14
15    return new Response($response);
16 }
```

Listing 7.2: Code snippet of the login controller

For persisting the request in a database the Object-Relational Mapping (ORM) mapper doctrine is used. This mapper takes care of the database connection as well as the error handling. So there is no need to type the SQL statements manually. The additional advantage is that this design provides a very good scalability. If new features are discovered or a new additional device ought to be simulated, it is easily possible to add new controllers serving the new requests.

The next feature that has been implement is the video stream. Just like the IP camera the honeypot is able to stream a video to a client using the RTSP protocol. The implementation of this feature consists of two parts. The first one is the RTSP simple server which has already been explained in Chapter 5.6 “RTSP-Simple Server”. The second part is a python script which reads a stored video on the disks and sends it to the RTSP server. Clients can now connect to that server to access the video stream. The video processing in python was achieved by the use of the OpenCV library in combination the ffmpeg command line utility.

```
1  import cv2
2  import subprocess as sp
3  if __name__ == "__main__":
4      rtsp_server = 'rtsp://localhost:8600/mystream' # push server (output
           ‘! server)
5      cap = cv2.VideoCapture("/demo-video/demo.mp4")
6
7      sizeStr = str(int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))) + \
8          'x' + str(int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))
9      fps = int(cap.get(cv2.CAP_PROP_FPS))
10     command = ['ffmpeg', # ... Built the ffmpeg command...
11                rtsp_server]
12
13     process = sp.Popen(command, stdin=sp.PIPE)
14     while(cap.isOpened()):
15         ret, frame = cap.read()
16         ret2, frame2 = cv2.imencode('.png', frame)
17         process.stdin.write(frame2.tobytes())
```

Listing 7.3: Code snippet of the python RTSP stream

7.2 Comparison Camera and Honeypot

We observe the IP camera and, on the other hand, the simulated services on the honeypot. Each of the implemented features is compared on its own. First off, the initial information gathering process is correlated. When the whole project was set up, one

question arose: Was it possible to create a honeypot that is virtually indistinguishable from the original camera? But for all that, the result of the proof-of-concept implementation was surprisingly affirmative. The two code segments below show the port scan results of the two systems. The first one is from the original IP camera and the second is from the honeypot.

```
nmap -sV -oA kamtron_scan 192.168.178.67
Starting Nmap 7.91 ( https://nmap.org ) at 2022-02-24 05:01 EST
Nmap scan report for 1jfiegbrcrfka_eth0 (192.168.178.67)
Host is up (0.0018s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE  VERSION
80/tcp   open  http?
8600/tcp open  asterix?
1 service unrecognized despite returning data. If you know the service/
    version, please submit the following fingerprint at https://nmap.org
    /cgi-bin/submit.cgi?new-service :
SF-Port8600-TCP:V=7.91%I=7%D=2/24%Time=62175786%P=x86_64-pc-linux-gnu%r(NU
SF:LL,6C,"8\0\0\0l\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\xae\xd2\xcc\[\0\0
SF:\0\0\xe4\xb7\xd1v\0\0\0\0\xf5\x8f\x05Tmrmt_hello\0\0\0\0\0\0\0\0\0\0\0\
SF:0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\x0e\0\0\0\x08\xb8\xd1v\0\0\0\x001jfiegbrc
...
```

Listing 7.4: NMAP Scan of the target device

```
nmap -sV -oA honeypot_scan 192.168.178.76
Starting Nmap 7.91 ( https://nmap.org ) at 2022-02-24 05:11 EST
Nmap scan report for 2jfiegbrcrfka_eth0 (192.168.178.76)
Host is up (0.00054s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE  VERSION
80/tcp   open  http     MWS 0.01
8600/tcp open  asterix?
1 service unrecognized despite returning data. If you know the service/
    version, please submit the following fingerprint at https://nmap.org
    /cgi-bin/submit.cgi?new-service :
SF-Port80-TCP:V=7.91%I=7%D=2/24%Time=621759FA%P=x86_64-pc-linux-gnu%r(GetR
SF:equest,37B,"HTTP/1\.1\x20200\x20OK\r\nDate:\x20Thu,\x2024\x20Feb\x20202
SF:2\x2010:12:10\x20GMT\r\nServer:\x20MWS\x200\.01\r\nLast-Modified:\x20We
SF:d,\x2016\x20Feb\x202022\x2018:17:27\x20GMT\r\nETag:\x20\"27a-5d826a982c
SF:6ce\"\r\nAccept-Ranges:\x20bytes\r\nContent-Length:\x20634\r\nVary:\x20
...
```

Listing 7.5: NMAP Scan of the honeypot

As can be seen the only difference between the two devices is the type of data returned in the RTSP stream. This difference is negligible, because it is very unlikely that an attacker will actually notice the difference.

After the first comparison the next feature, the web application, is addressed. Based on the fact that it was possible to use the original front-end JavaScript application it is not virtually impossible to notice any differences between the two devices. Even a detailed analysis of the sent and received requests does not reveal any perceivable differences. The only differences found are revealed in the two images below.

Figure 7.2: IP Camera Web Request

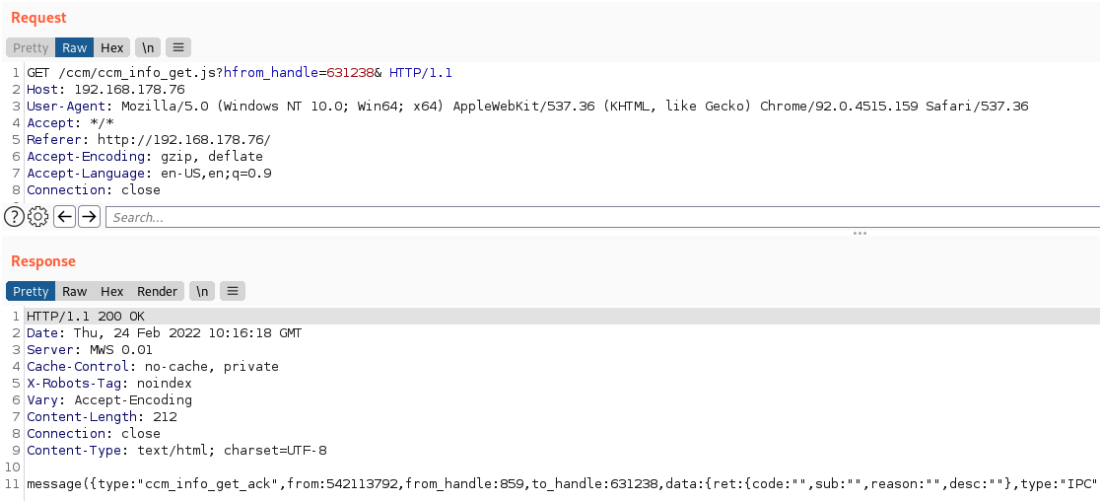

Figure 7.3: Honeypot web request

The minor differences between the two requests are the returned content type and a few additional headers. With additional fine-tuning of the Apache server configuration it would even be possible to eliminate them altogether. Unassumingly, we can see that this proof of concept implementation of the honeypot is conclusive.

We could prove that it is possible to simulate a device on the Raspberry Pi hardware platform in such a way that it is very hard to detect the honeypot. In the first overview both devices have the same behaviour and the same look and feel for the user/attacker. The marginal differences can only be detected in an in-depth analysis of both devices side by side. This kind of comparison, however, is inaccessible to any attacker due to the absence of further information on the simulated type of IP camera.

Chapter 8

Attack Analysis

In this chapter some detailed analysis of attacks against two honeypots in use will be presented. At first, attacks against the self-designed Smart Home Honeypot will be evaluated. This evaluation will be followed by the analysis of attacks against an SSH honeypot. The protection against the identified attack vectors will be presented in the last part of this chapter.

8.1 Test Setup

The implementation of a honeypot has to be well-prepared, otherwise the honeypot is potentially a weak spot of the network. In order to safely record and analyze real attacks against the honeypots, a separate network segment has been created. This segment is called Demilitarized Zone (DMZ). One of the key properties of a DMZ is, that connections from the untrusted internet as well as connections from the trusted office network are allowed. A more detailed explanation of network segmentation has already been provided in ensuing Chapter 3.6 “Network Segmentation”. The enforcement of the security rules which allow or block specific network traffic is done by the firewall. A graphical implementation of the network design can be seen below.

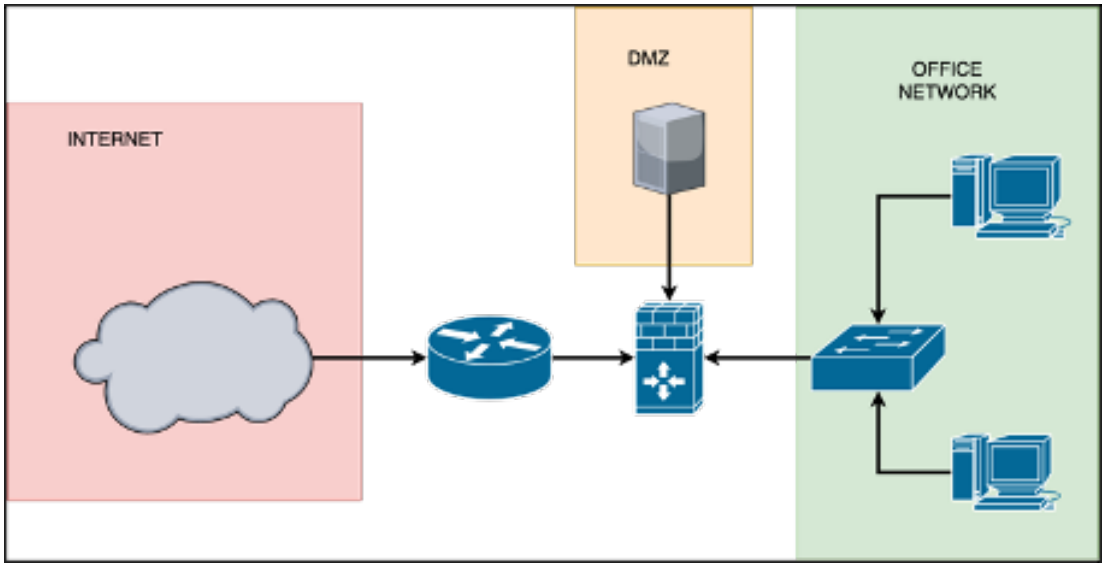

Figure 8.1: IT Infrastructure Design

Security policies have to be applied between the different security zones, so that the potential damage is limited if the honeypot gets compromised by an attacker. The following security rules were enforced to restrict the network traffic to the DMZ or originating from the DMZ. With these restrictions in place an attacker is not capable of successfully establishing connections to the office network originating either from the internet nor from the honeypot. The risk of a compromised network and also the potential damage is significantly reduced.

- Traffic from the internet to the DMZ using port 22 Secure Shell (SSH), port 80 (HTTP), port 8600 (RTSP) are forwarded to the honeypot.
- Traffic from the internet to the DMZ on any other port is blocked.
- Traffic from the DMZ to the internet on any port is allowed.
- Traffic from the DMZ to the office network is blocked.
- Traffic from one management PC to the DMZ is allowed.
- Any other traffic from the office network to the DMZ is blocked.

The experimental setup with the honeypots were online for four weeks in order to gather significant data.

8.2 Data Analysis

The aim of the experimental setup was to gain knowledge about real attacks against Smart Home devices. Each of the applied honeypots focuses on a different attack vector. The first one, the self-designed Smart Home Honeypot, homes in on web-based attacks. These attacks focus on the configured web servers which are running on the devices as well as on the web applications themselves. The honeypot stores information about all web requests made by an attacker. The second honeypot is used to simulate an accessible system shell of a Smart Home Device. These shells are commonly used for the configuration of the devices and are typically protected by usernames and passwords. To attract attackers, every username and password combination is eligible for a 'successful' login. During an attack the used username and password combination as well as the used commands are recorded for later analysis. Although both honeypots focus on different attack vectors, the structured way of storing the information is the same for both implementations. The information gathered by the two honeypots is stored either in text files using the JavaScript Object Notation (JSON) syntax or the Structured Query Language (SQL) database. Due to the huge number of attacks caught by each of the honeypots the analysis is assisted by Python scripts. These scripts are used to extract the most important parts and create a top five list of the interesting parameters. A detailed explanation of the results is provided in the two sections below.

8.2.1 Smart Home Honeypot

The first one to be examined within this analysis was the self-designed smart home honeypot. Information has been gathered from the log files of the web server and from the log files of the Real Time Streaming Protocol (RTSP) simple server. Unfortunately, there were no connections established to the RTSP server within the four weeks of the investigation period. One reason for the lack of connection might the use of a high port number. All ports from zero up to 1,024 are referred to as well-known ports. In this port range most of the widely used applications have a dedicated port, e.g., HTTP on port 80, HTTPS on port 443. In order to find the open RTSP port 8600 a larger and therefore more time-consuming port scan would have been needed. Most of the automated attacks skip the port scan and try to connect directly to the ports of interest. With this procedure the efficiency of an attack can be increased (Project, 2004). The access log of the web server shows the complete opposite: Nearly 600 different IP addresses were connected to the honeypot generating around 2,000 lines of log messages.

As already mentioned, the analysis of these huge number of lines is only possible with

scripted assistance. To extract the interesting information out of the Apache access log the following function has been used.

```
1  def parseApacheLogLine(line):
2      regex = '([(\d\.)]+) - - \[(.*)\] "(.*?)" (\d+) (\d+) "(.*?)" "(.*?)"'
3      # Structure Parsed Line:
4      # 0: IP
5      # 1: Timestamp
6      # 2: HTTP method + Path
7      # 3: Status code
8      try:
9          parsed_line = re.match(regex,line).groups()
10         return parsed_line[0], parsed_line[2], parsed_line[3]
11     except:
12         return "", "", ""
```

Listing 8.1: Parse Apache Access Log

This function is embedded in a script which opens the log file and reads it line by line. Each line is processed with the function. The heart of the function is the regular expression in line 2 which processes the input based on the pattern. The log message is then stored in an array where each item of the array holds a specific information. In the end the IP address as well as the HTTP method + path and the HTTP status code is returned to the rest of the program.

By extracting the essential information is could be found out that the attacks focused primarily on the web server itself and not on the simulated device. Only two connections passed the login screen, although every password has been accepted. That's a clear indicator that automated attacks were used with no manual interaction. Although nearly no attack focused on the simulated device, it does not mean that the honeypot did not work as expected. An exploited vulnerability of the web server itself is as dangerous as an exploited vulnerability of the device. Therefore, also attacks against the web server have to be notified and analysed. Most of the attacks tried to access the `/.env` file or the `/.robots.txt` file. The first one is used in NodeJS environments to store sensitive data, whereas the second file can be used to get further information about the structure of the website. Accessing these files is done during the information gathering phase of an attack. A more detailed explanation of such an attack that happened during the test period, is provided in Chapter 8.3 "Attack Scenarios".

Although the source of an attack usually cannot be identified by the victim, it was very interesting to see which one was the last server contacting the honeypot. Successful attackers try their best to hide their IP address. One of the techniques used to achieve

this goal is the use of proxy servers. Proxy servers can be utilized to redirect connections. The attacker connects to the proxy and initiates a new connection towards the victim originating from the proxy. Without access to the proxy server, the victim has no chance to identify the origin of the attack as it only sees the connection to the proxy server while the connection between the attacker and the proxy server is concealed. Of course, the proxy server can be chained which makes it even harder to identify the source of the attack.

The analysis revealed the top 5 most used IP addresses:

1. IP: 45.155.204.146 : 112 times (Russia)
2. IP: 188.215.235.120 : 100 times (Cambodia)
3. IP: 2.57.121.51 : 83 times (Romania)
4. IP: 45.148.10.81 : 44 times (Netherlands)
5. IP: 157.230.216.203 : 38 times (United States)

8.2.2 SSH Honeypot

The second honeypot which has been used for data gathering is the SSH honeypot called "Cowrie". Cowrie is an open source SSH honeypot with lots of features already built into it. The source code of the project can be found on a public GitHub repository (Oosterhof, 2022). This honeypot can be installed within minutes and the placement in the network is the same as the one for the first honeypot. The only difference between the two honeypots is that the SSH honeypot requires a different port mapping than the Smart Home honeypot. The firewall security policy has to be adjusted accordingly. Cowrie provides a variety of different logging mechanisms. In this setup the MariaDB database as well as JSON log files have been used. Although the honeypot was only online for a short period of around two weeks, more than 344,000 lines of text have been written to the log file. It would be impossible to analyze the data by hand, so a simple Python script has been created. This script consists of three parts. The first one reads the content of the file and creates a JSON object out of the string. Each line of the file is a single event such as login, logout, or an executed command. In the next step each line is processed individually, and the key indicators were extracted. For this experiment the used login credentials, the executed commands and the origin of the attack were extracted from the gathered information. The last step of Python script counts the events and provides the results in an easily readable way.

At first the login credentials were analyzed. The list below shows the top 5 most used

username and password combination

1. user: **root**, password: **admin** (23,302 times)
2. user: **root**, password: **111111** (1,199 times)
3. user: **user**, password: **1** (394 times)
4. user: **root**, password: **11111111** (349 times)
5. user: **root**, password: **123123** (178 times)

As can be seen from the results above, it is very important that strong passwords are used all the time. Even if the device is "only" accessible from the internal network. The next interesting information which could be gathered are the executed commands. The most frequently executed commands are used to download execute shell scripts. These scripts are either a backdoor to the system which should ensure reliable access or crypto miners which make use of the hardware resources from the server. The executed commands contain sensitive information such as script parameters or Uniform Resource Locator (URL) addresses where scripts are loaded, therefore it is not possible to print the commands into this thesis. Other commands which were not intended to load any scripts were used for information gathering. As an example the command `uname -s -v -n -r -m` were used to display information about the kernel version and the operating system in general. A detailed explanation of the identified attack will be provided in the next section.

8.3 Attack Scenarios

This section is used to explain attacks that have been recorded by one of the used honeypots. Two of the attacks tried to use the vulnerabilities to gain access to the device and one attack tried to install a so-called crypto miner. The crypto miner attack has been recorded on the cowrie SSH honeypot, whereas the other two attacks have been recorded on the Smart Home Honeypot.

8.3.1 Cryto Miner

The first attack which is analysed was found in the following log message:

```
1 {"eventid":"cowrie.command.failed","input":"curl -s -L http://<censored
    ↪ data>/setup_c3pool_miner.sh | LC_ALL=en_US.UTF-8 bash -s <command
    ↪ shortend>","sensor":"raspberrypi","timestamp":"2022-04-25T10
    ↪ :23:11.499849Z","src_ip":"<private Data>","session":"24747365e06b"}
```

Listing 8.2: Crypto Miner example

The attacker tried to execute two commands connected with a pipe. The first one `curl -s -L` was used to download the miner to the victim. The second command sets a charset and tries to execute the install script which has previously been downloaded. A complete analysis of the payload is beyond the scope of this thesis.

Although these crypto miners do not harm the network directly nor do they try to steal private information. The installation should be avoided because they still have an impact to the network. The aim of such miners is to use the system resources for generating crypto currency. The impact to the network is reduced performance due to the unauthorized employment of hardware resources.

An effective countermeasure to this attack is the use of a firewall in combination with an Intrusion Detection System (IDS)/Intrusion Prevention System (IPS) which is able to track the signature of such payloads and blocks the connection immediately. Downloads cannot be completed anymore and therefore the installation process will fail.

8.3.2 Password leakage

The aim of the next attack is to leverage passwords with the use of Common Vulnerabilities and Exposures (CVE) CVE-2012-707. This attack has been identified with the following log message from the Apache access log file.

```
1 <censored IP> - - [01/Apr/2022:16:57:19 +0100] "GET /?a=fetch&content=<php>
    ↪ die(@md5(HelloThinkCMF))</php> HTTP/1.1" 200 600 "-" "Mozilla/5.0 (
    ↪ Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    ↪ Chrome/78.0.3904.108 Safari/537.36"
```

Listing 8.3: Password leakage

This attack uses a window of vulnerability in the WordPress Core 3.4 to leak passwords. Word- Press used a weak hashing algorithm, which allowed attackers to gain access to the plain-text passwords. The impact of this attack depends on the strength

of the password. The other thing that raises the vulnerability is if the same password is being used for different services.

Mitigating such attacks with network protection devices can be very tricky. It is only possible if the signature is already present to firewalls or IDS/IPS systems. Otherwise, the most effective mitigation techniques are regular updates of all software components and a strong password policy with a separate password for each service.

8.3.3 Remote Code Execution

The last attack and – at the same time - one of the most dangerous recorded attacks belongs to the category ‘remote code execution’. This category allows an attacker to execute arbitrary code on the victim’s machine. The attack has been found in the following log message

```
1 <private IP> - - [02/Apr/2022:21:41:06 +0100] "POST /cgi-bin/.%2e/.%2e/.%2e
    ↪ /.%2e/bin/sh HTTP/1.1" 400 459 "-" "Mozilla/5.0 (Windows NT 10.0;
    ↪ Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
    ↪ /78.0.3904.108 Safari/537.36"
```

Listing 8.4: Remote Code Execution

The attacker tried to use a so-called path traversal vulnerability from the web server to execute shell commands. Also, a simple Web Application Firewall (WAF) bypass has been used, by using the URL encoded form of a dot which is `%2e`. WAF’s try to detect special URL pattern to identify an attack. The pattern `../` will be recognized as a path traversal attempt, whereas `.%2e/` might not be detected, although it has the same meaning for the web server. The URL decoding takes place after the WAF by the web server itself. With this vulnerability the attacker is able to take over the whole device. The compromised device can be used as so-called "jumphost" for further navigation through the internal network (Project, 2004).

The associated risk with such a vulnerability is most daunting. One potential damage could be the encryption of files by ransomware. The other one is the potential theft of files. Possible mitigation for such attacks are the same as for the previous attacks: the use of network security devices and a regular update of the software.

The most effective strategy to limit the potential impact of an attack of this kind is the network segmentation. With network segmentation in place, the attacker will not be able to travel through the complete network and gain access to other devices or files.

As the remote code execution is doubtlessly the most dangerous attack, the graphic

below illustrates the different steps of a network infiltration and where the attack can be stopped.

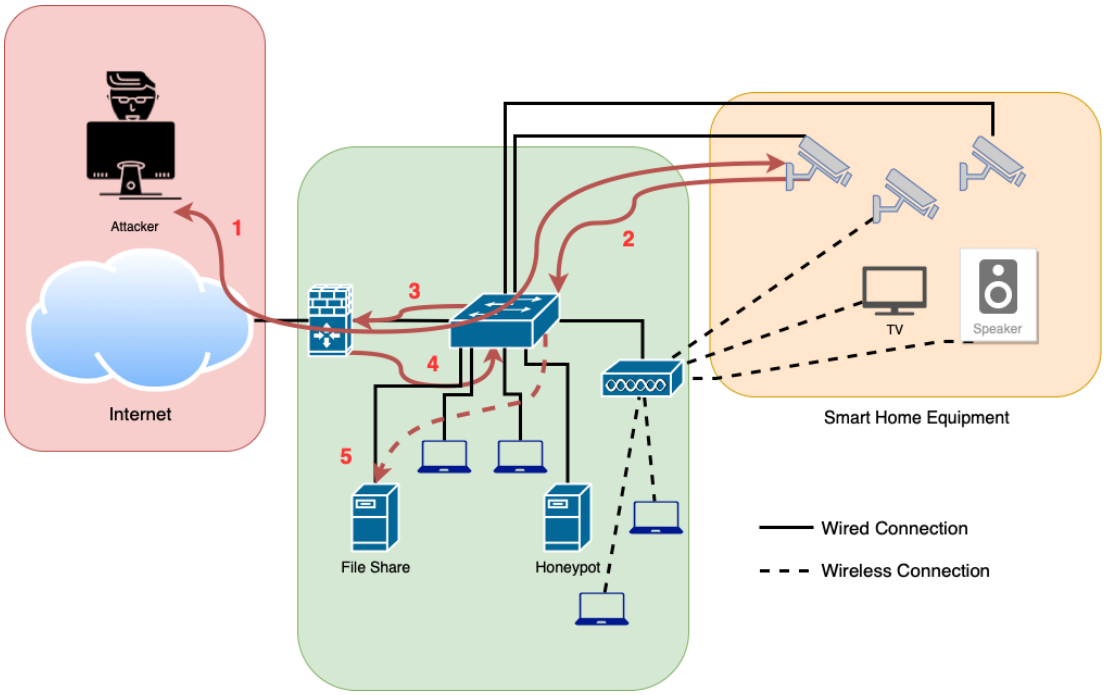

Figure 8.2: IT Network Attack

1. Step: A connection between the IP camera and the attacker has been successfully established. The attacker has persistent access to the device.
2. Step: The attacker tries to move inside the network. If no network segmentation is used, he/she can now directly connect to the file share.
3. Step: In this example with network segmentation, the traffic is redirected to the firewall.
4. Step: Without a strict security policy the traffic would be forwarded by the firewall, to the file share.
5. Step: The connection between the IP camera and the file share has been successfully established.

This scenario highlights the importance of the firewall device with regard to the overall network security. The firewall can stop the attack at multiple points. In Step 1 the firewall is capable of blocking the remote-control connection between the attacker and the IP camera. In Step 3 the firewall is capable of checking if the connection attempt between the file share and the camera is legitimate.

Chapter 9

Conclusion and Outlook

When the topic of this master thesis was raised for the first time it was not a foregone conclusion that it would be possible to be accomplished. The safety analysis in the beginning uncovered major security issues of smart home devices; this, in turn, raised the awareness for IT security in a smart home environment.

The causes of risk were addressed in detail. In order to examine the effectiveness of the use of honeypots an experiment was thoroughly prepared and conducted. Two honeypots of very different descent have been used: The first one was self-designed while the second was ready-made (cowrie SSH) offered by public GitHub repository.

The self-designed honeypot performed even better than anticipated. It was almost impossible for attackers to notice any difference between the honeypot and a real device that it was meant to simulate. Overall, it was on par with the object of comparison, the cowrie SSH honeypot. The reason for this surprising outcome was the bespoke nature of that self-designed example, made-to-measure for the particular purpose. The cowrie SSH honeypot, on the other side, had its virtues as well.

Of course, the security concepts mentioned in the past few chapters have to be allowed for. This being said, the application of honeypots can increase the security level of a smart home by a wide margin. The initial expectations for the increase of smart home security by the use of honeypots were even exceeded.

What might be the next stride to even promote the effectiveness of honeypots?

The first step could be an improvement in the deployment and update process of honeypots. This way, the scalability of the system would be enhanced. An automated deployment of smart home honeypots could close the gap between the generic frameworks and self-designed honeypots in terms of ease and comfort.

The second objective could be the integration of external Security Information and Event Management (SIEM) analysis tools which enhances and simplifies the analysis of the log files.

At the time being, for the proof of concept only a single Smart Home device has been simulated. Even though a commonly used device has been chosen, this honeypot might not be suitable for each and every smart home application.

Therefore, the ensuing step should be to extend the list of supported devices so that the honeypot is capable of simulating a range of devices if needed. With this next step the possible coverage of smart homes can be significantly improved. It would be favourable to implement different alerting mechanisms, avoiding the obligation to install the messenger app Telegram or the like.

Acronyms

AI	Artificial Intelligence
API	Application Programming Interface
CVE	Common Vulnerabilities and Exposures
DDoS	Distributed Denial of Service
DMZ	Demilitarized Zone
FTP	File Transfer Protocol
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
IPS	Intrusion Prevention System
IT	Information Technology
JSON	JavaScript Object Notation
MAC	Media Access Control
NAS	Network Attached Storage
NMAP	Network Mapper
ORM	Object-Relational Mapping
PHP	recursive initialism PHP: Hypertext Preprocessor
RTSP	Real Time Streaming Protocol
SD	Secure Digital
SIEM	Security Information and Event Management
SQL	Structured Query Language

SSH Secure Shell

SSID Service Set Identifier

TCP Transmission Control Protocol

URL Uniform Resource Locator

VLAN Virtual Local Area Network

WAF Web Application Firewall

Bibliography

Aler9 (2022). *RtspSimpleServer*. Available from: https://github.com/aler9/rtsp-simple-server [Jan. 25, 2022] (cit. on p. 20).

B. Lingenfelter, I. V. & S. Sengupta (2020). "Analyzing Variation Among IoT Botnets Using Medium Interaction Honeypots". In: *10th Annual Computing and Communication Workshop and Conference (CCWC)* 10, pp. 0761–0767. DOI: 10.1109/CCWC47524.2020.9031234 (cit. on pp. 4, 9).

Canary, T. (2022). *OpenCanary*. Available from: https://opencanary.readthedocs.io/en/latest/starting/opencanary.html [May 20, 2022] (cit. on pp. 28, 29).

Chu, M. & Y. Song (2021). "Analysis of network security and privacy security based on AI in IOT environment". In: *2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE)*, pp. 390–393. DOI: 10.1109/ICISCAE52414.2021.9590786 (cit. on pp. 5, 9).

E-Control (Jan. 1, 2022). *Was kostet eine kWh Strom*. Available from: https://www.e-control.at/konsumenten/strom/strompreis/was-kostet-eine-kwh [Jan. 25, 2022] (cit. on p. 8).

H. Wafi A. Fiade, N. H. & R. B. Bahaweres (2017). "Implementation of a modern security systems honeypot Honey Network on wireless networks". In: *International Young Engineers Forum (YEF-ECE)*, pp. 91–96. DOI: 10.1109/YEF-ECE.2017.7935647 (cit. on pp. 5, 11).

Iqbal, W., H. Abbas, M. Daneshmand, B. Rauf & Y. A. Bangash (2020). "An In-Depth Analysis of IoT Security Requirements, Challenges, and Their Countermeasures via Software-Defined Security". In: *IEEE Internet of Things Journal* 7.10, pp. 10250–10276. DOI: 10.1109/JIOT.2020.2997651 (cit. on pp. 5, 9).

Joshi, R. & A. Sardana (2011). *Honeypots: a new paradigm to information security*. 1st ed. CRC Press. ISBN 1578087082 (cit. on pp. 4, 7, 11).

Ltd, R. P. (Mar. 14, 2018). *Raspberry Pi 3 Model B+*. Available from: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/ [Dec. 20, 2021] (cit. on pp. 8, 17).

Mukherjee, L. (Dec. 30, 2021). *What Is a Honeypot in Network Security? Definition, Types Uses*. Available from: https://sectigostore.com/blog/what-is-a-honeypot-in-network-security-definition-types-uses/ [Jan. 25, 2022] (cit. on p. 7).

Oosterhof, M. (2022). *Cowrie*. Available from: https://github.com/cowrie/cowrie [Apr. 1, 2022] (cit. on p. 41).

Project, T. H. (2004). *Know your enemy: learning about security threats*. 2nd ed. Addison-Wesley Longman, Amsterdam. ISBN 0321166469 (cit. on pp. 4, 39, 44).

Rauter, T. (2019). *Improving security in industrial control systems using a honeypot/honeynet-based approach* (cit. on pp. 4, 28).

Ruether, T. (Jan. 6, 2022). *RTSP: The Real-Time Streaming Protocol Explained (Update)*. Available from: https://www.wowza.com/blog/rtsp-the-real-time-streaming-protocol-explained [Jan. 25, 2022] (cit. on p. 10).

Sears, A. (Sept. 23, 2019). *'Felt so violated:' Milwaukee couple warns hackers are outsmarting smart homes*. Available from: https://www.fox6now.com/news/felt-so-violated-milwaukee-couple-warns-hackers-are-outsmarting-smart-homes [Feb. 25, 2022] (cit. on p. 1).

Statista (June 2021). *Prognose zur Anzahl der Smart Home Haushalte nach Segmenten in Österreich für die Jahre 2017 bis 2025*. Available from: https://de.statista.com/prognosen/801569/anzahl-der-smart-home-haushalte-nach-segmenten-in-oesterreich [Feb. 25, 2022] (cit. on p. 2).

Wendzel, S. & J. Plötner (2007). *Praxisbuch Netzwerk-Sicherheit : Risikoanalyse, Methoden und Umsetzung ; [optimale Netzwerk- und Serverabsicherung ; für Unix/Linux- und Windows-Systeme ; VPN, OpenVPN, IT-Grundschutz, Penetration Testing, Viren, Würmer und Trojaner]*. Galileo Press. ISBN 9783898428286 (cit. on pp. 5, 10–12).