DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

Hacking Into Someone's Home using Radio Waves

Ethical Hacking of Securitas' Alarm System

AXEL LINDEBERG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Hacking Into Someone's Home using Radio Waves

Ethical Hacking of Securitas' Alarm System

AXEL LINDEBERG

Degree Programme in Computer Science and Engineering
Date: June 23, 2021

Supervisor: Pontus Johnson

Examiner: Robert Lagerström
School of Electrical Engineering and Computer Science
Host organization: Försvarsmakten (Swedish Armed Forces)
Swedish title: Hacka in i Någons Hem med hjälp av Radiovågor
Swedish subtitle: Etiskt Hackning av Securitas Hemlarmsystem

Abstract

The number of IoT systems in our homes has exploded in recent years. By 2025 it is expected that the number of IoT devices will reach 38 billion. Home alarm systems are an IoT product that has increased dramatically in number and complexity in recent years. Besides triggering an alarm when an intruder tries to break in, a modern system can now control your light bulbs, lock and unlock your front door remotely, and interact with your smart speaker. They are undeniably effective in deterring physical intrusion. However, given the recent rise in complexity how well do they hold up against cyber attacks?

In this thesis, a smart home alarm system from SecuritasHome is examined. A comprehensive security analysis was performed using penetration testing techniques and threat modeling. The work focused mainly on radio frequency (RF) hacking against the systems RF communication. Among other things, a critical vulnerability was found in the proprietary RF protocol, allowing an attacker to disarm an armed system and thus completely bypass the system's functionality. The security of the system was deemed to be lacking.

Keywords

penetration testing, threat modeling, IoT, computer security, home alarm system

Sammanfattning

Antalet IoT system i våra hem har exploderat de senaste åren. Vid år 2025 förväntas antalet IoT enheter nå 38 miljarder. Hemlarmsystem är en typ av IoT-produkt som ökat dramatiskt i komplexitet på senare tid. Förutom att framkalla ett larm vid ett intrång kan ett modernt hemlamsystem numera kontrollera dina glödlampor, låsa och låsa upp din ytterdörr, samt kontrollera dina övervakningskameror. De är utan tvekan effektiva på att förhindra fysiska intrång, men hur väl står de emot cyberattacker?

I denna uppsats undersöks ett hemlarmsystem från SecuritasHome. En utförlig säkerhetsanalys gjordes av systemet med penetrationstestnings-metodiker och hotmodellering. Arbetet fokuserade mestadels på radiovågshackning (RF) mot systemets RF-kommunikation. Bland annat hittades en kritiskt sårbarhet i systemets RF-protokoll som gör det möjligt för en angripare att avlarma ett larmat system, och därmed kringå hela systemets funktionalitet. Säkerheten av systemet bedömdes vara bristfällig.

Nyckelord

penetrationstestning, hotmodellering, IoT, datasäkerhet, hemlarmsystem

Acknowledgments

I would like to thank Fredrik Heiding, PhD student within cybersecurity at KTH, for helping me procure the alarm system investigated in this thesis, as well as the HackRF SDR.

I would also like to acknowledge Professor Andreas Noack from the University of Applied Sciences Stralsund in Germany. Not only did he co-create the excellent tool *Universal Radio Hacker* which was used extensively in this thesis. He also offered up a lot of his time in personally helping me when I initially felt way out of my depth with RF hacking by answering questions about the URH tool, RF communication in general, and figuring out how to capture good signals for this system.

Next, I would like to thank my girlfriend, Caroline, who had to hear me go on and on about radio waves and RF hacking for months, handle the stressful periods, for continuously proofreading the thesis, and for putting up with this whole situation during a pandemic. The same goes for my family.

Additionally, I would like to thank Securitas AB for being reasonable, nice, and easy to deal with during the entire disclosure process of the vulnerabilities.

I would, of course, like to thank my supervisor at Försvarsmakten. They gave me invaluable insights and expertise during this entire project. All the way from selecting what type of system to explore, to sharing their knowledge during the pentesting phase, to proofreading the final version. They also lent me more than enough of their time, meeting with me every week to discuss the thesis which I really appreciated.

Above all, I would like to thank my KTH supervisor Pontus Johnson. Before even starting this thesis, his excellent course Ethical Hacking (EN2720) opened my eyes to this entire field and was easily my favorite course at KTH. He also personally helped me get in contact with and recommended me to several organizations within the security industry during the search for a place to write my thesis as well as during my job hunt after graduation. During the thesis, Pontus also gave a lot of his time, answered questions, and gave excellent feedback and encouragement.

Lastly, a special thanks to KTH for these last five years!

Stockholm, September 2021
Axel Lindeberg

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ASK Amplitude-shift keying

CSRF Cross-site request forgery

DoS Denial of Service

FSK Frequency-shift keying

IoT Internet of Things

MITM Man-in-the-middle

OSINT Open source intelligence

PSK Phase-shift keying

RF Radio-Frequency

SDR Software-defined Radio

TCP Transmission Control Protocol

URH Universal Radio Hacker

Chapter 1

Introduction

Home automation and the number of connected devices in our homes have exploded in recent years. The number of Internet of Things (IoT) devices especially has increased dramatically. It is predicted there will be about 38 billion of them by 2025 [1]. Many of IoT devices are connected to the internet and that fraction is bound to increase given the rise of 5G technology. While these devices can do amazing things, everything from smart speakers to connected refrigerators, they are unfortunately hardly famous for their security. While this is well known in the IT-security community, the general non-tech-savvy consumer is perhaps not as aware of the security considerations when bringing an IoT device into their home.

A type of connected device that has become increasingly common in people's homes are smart Home Alarm Systems. In fact, the global market for smart home alarm systems is expected to grow by 20% in 2021 alone [2]. They protect your house from intruders by sounding an alarm when a suspected intrusion has occurred. Often a security central is immediately notified and security personnel sent to the site to investigate. These systems can be incredibly complex and can include multiple external peripherals like motion detectors, surveillance cameras, smoke detectors, etc. In recent years their scope and complexity has rapidly expanded even further. A modern smart home alarm system can now often control home automation systems like smart light bulbs and connected coffee machines, can interact with your smart speaker, and can even lock and unlock your door via smart locks. Additionally, they can be controlled remotely via mobile apps and web portals. These are undoubtedly useful features and undeniably these systems do provide

protection against physical intrusion. However, one might wonder, given their recent rise in complexity, how secure these systems are against cyberattacks. How much of a focus is the cybersecurity of these systems is to the companies behind them? Given the large increase of features, have the vendors thought of all new emergent cybersecurity threats?

This thesis will examine the cybersecurity of a smart home alarm system from Securitas, called SecuritasHome. There were many aspects to consider when deciding what system to investigate, which is detailed in section 3.1. The SecuritasHome system comes with features such as alarming the system using a remote keypad and a four-digit pin, smoke detection, motion detection with a corresponding camera, and control of home automation devices (see section 3). However, the main panel of the system, the *brain of the system* so to speak, is connected to the internet via a local Ethernet cable as well as 3G telecommunication. If one were to comprise the security of this system there could be devastating consequences such as disarming the alarm, allowing an intruder to enter the house without setting triggering the alarm.

1.1 Research question

This report will try and answer the following research question:

> *Is the SecuritasHome Home Alarm System secure against cyberattacks?*

In particular, this question can be broken down into two parts:

- What vulnerabilities are present in the system?
- How can the vulnerabilities be exploited?

The security analysis presented in this thesis was performed on the following firmware versions. These were the latest versions at the time of writing (spring of 2021):

- Alarm.com version: `193d`
- Climax Technology version: `HPGW-G 0.0.2.23F BG_U-ITR-F1-BD_BL.A30.20181117`

1.2 Objectives

The objective of this thesis is to assess the security of the SecuritasHome home alarm system. In essence, the objective in terms of the degree project is to assess whether or not the system can be considered secure from a computer security perspective. To achieve this a comprehensive security analysis was made of the system, to investigate which attack vectors the system is vulnerable to. Considering the large attack surface of the system in question, given its complexity and variety of features, some areas had to be delimited. More on this in section 1.4.

From the perspective of the host organization, *Försvarsmakten*, the objectives were to assess the security of home alarm systems in general, which have become increasingly common in Swedish homes. While these systems are generally considered effective against physical intrusion, less is sure about their security when it comes to cyberattacks. The host organization wanted a thorough investigation into the IT security of such a system.

1.3 Methodology

During this thesis a seven-step penetration test methodology, as presented by Weidman [3], was followed. This is detailed in section 2.1. For the threat modeling phase of the project a threat modeling technique specialized for IoT systems presented by Guzman and Gupta [4], was used. Additionally, the OWASP top ten list of vulnerabilities for IoT systems [5] was used to identify threats and used as an attack library, as well as the ETSI EN 303 645 standard [6]. The methodology is explained in more detail in chapter 2.

1.4 Delimitations

The system under consideration is very complex. It consists of many features, applications, and physical peripherals. As such, there is unfortunately not enough time within the scope of a degree project to exhaustively consider the full attack surface. Some things were also delimited due to legal reasons. As such the following major delimitations were done early in the project:

- The external cloud servers, hosted by *Alarm.com*. Legally, the security of these cannot be assessed without their permission.
- The mobile application, both the iOS and Android versions. This was delimited for two primary reasons, the major one being time and the other being the author not having easy access to an iOS device.
- The 3G wireless telecommunication. This was primarily due to the custom hardware required and the general security of this encrypted protocol Koien and Haslestad.
- The security of additional peripherals not included in the starter kit, see 3.3.1, as well as Z-wave peripherals.
- Any attack requiring *physical access* to the system. The goal of the system is to prevent physical access to the home. All components of the system are located inside so if an attacker has physical access to the system then the goal has already failed. Therefore these types of attacks were not deemed interesting to explore. Note that this does not exclude attacks that only require physical proximity, like being outside the door for example.

1.5 Structure of the thesis

This report is structured into the following chapters:

- Chapter 1 (Introduction) gives an introduction to the thesis area and research question. Additionally, delimitations of the project are listed.
- Chapter 2 (Method) gives a thorough explanation of the methodology of this thesis. First, the general penetration testing methodology applied in the project is explained and lastly, the threat modeling technique is explained.
- Chapter 3 (System Under Consideration) explains the system under consideration in detail, e.g all hardware and software components of the system, as well how the system was selected and an overview of the companies behind the system.
- Chapter 4 (Related work) lists identified related work to this thesis. This, among others, includes a security analysis of a system based on similar

hardware and partially the same firmware, as well as talks on Radio-Frequency (RF) hacking.

- Chapter 5 (Threat Model) presents a threat model of the system, more on the threat modeling technique used in section 2.2.
- Chapter 6 (Penetration testing) explains all penetration tests performed against the system, potential background information about the attack, the result of the test, as well as a discussion of the consequences and mitigations of them.
- Chapter 7 (Reported Vulnerabilities) explains which vulnerabilities were reported to the manufacturer and gives a detailed timeline of events during the responsible disclosure process, as well as the CVE ids connected to this report.
- Chapter 8 (Discussion) contains a discussion about the validity and efficiency of the methodology used, a discussion about the results from chapter 6, as well as a mandated section on the sustainability and ethics of the thesis.
- Chapter 9 (Conclusions & Future Work) concludes the thesis by presenting the conclusions of the report, to what extent the system can be considered *secure*, as well as a discussion about future work that could be done on examining the system's security.

Chapter 2

Method

The following chapter describes the method applied in this thesis. It is based on a seven-step process to penetration testing by Weidman [3]. In the first part, this method is described. That is followed by how each of these seven steps was applied in this thesis. Furthermore, for the threat modeling phase, a technique outlined by Guzman and Gupta [4] was used. This threat modeling process is also described below.

2.1 Penetration Testing methodology

In their book *Penetration testing: a hands-on introduction to hacking*, Weidman details a seven-step process for penetration testing [3]. This section firstly gives a brief description of all seven steps and lastly outlines how each step was performed in this thesis. Included in Weidman's method for penetration testing are the following seven steps:

1. *Pre-engagement*. This step involves communicating with the party that ordered the penetration test to be done. The goal of this step is to make sure both parties are on the same page and understanding of how the tests will be done. Things to agree upon, according to Weidman, are scope, testing window, and clear authorization from the other party that you are legally allowed to assess the security of their system.
2. *Information Gathering*. This step includes what is known as Open source intelligence (OSINT). OSINT is the process of using publicly available sources of information to gather information about the system

in question, a widely used technique in computer security. These sources include search engines like Google, news articles, public government data, academic papers, etc, [8]. One might also use port scanners like *Nmap*[1] and other application scanners to gather information about the system. Additionally, one might listen in on the network traffic of the system to gain an understanding of its behavior, using tools like *WireShark*[2] for example.

3. *Threat Modeling*. This step involves mapping out the components of the system, based on the information from the previous step. From that, you think of potential attacks and vulnerabilities of the system, their potential impact, and the likelihood of success. There are many different threat modeling techniques. More on this and the technique used in this report in section 2.2.

4. *Vulnerability Analysis*. This step involves actively pentesting the system to discover vulnerabilities. This can be done for example by manually probing the system or by using vulnerability scanners like *Metasploit*[3], *Burp Suite*[4], or *Nessus*[5].

5. *Exploitation*. This step involves exploiting the vulnerabilities discovered in the previous step. By exploiting these, the goal is to perform some malicious act on the system to subvert its security.

6. *Post Exploitation*. After a successful exploit, this step involves analyzing the consequences. If the exploit involves access to a machine one might investigate the file system, look for possibilities of privilege escalation, etc. One asks how severe this successful exploit is to the overall security of the system.

7. *Reporting*. This final step involves summarizing the findings to the interested party. Crucially, if the findings are to be publicized one should adhere to the principle of responsible disclosure.

What follows is a description of how each step above was applied in this thesis.

[1] https://nmap.org/
[2] https://www.wireshark.org/
[3] https://www.metasploit.com/
[4] https://portswigger.net/burp
[5] https://www.tenable.com/products/nessus

2.1.1 Pre-engagement

According to Weidman's method, the pre-engagement step is done in collaboration with the client. In this project, however, there is no clear client except the author and perhaps KTH and Försvarsmakten. The scope and expectations were continuously discussed during the course of the project. The companies behind the system (see 3.1) were not informed of the security analysis until after the project was finalized. Securitas, the seller of the system was contacted multiple times over the phone via their customer support to verify the legality of security testing the system but were not informed of the security assessment until after the thesis was finished.

2.1.2 Information Gathering

The information-gathering phase was done in several steps, the first one being OSINT. Initially, the model number of all devices was gathered from either physical labels on the peripherals or from Securitas' website[1]. Using the search engine Google, the devices and their manufacturer *Climax Technology* were quickly identified. From their website much more information about the system could be found, such as how the peripherals communicate and their proprietary RF protocol[2]. An additional resource was finding each component's FCC ID[3], from which one can find user manuals submitted to the FCC agency, official testing documentation, and more via *fccid.io*, see chapter 3.

2.1.3 Threat Modeling

Threat modeling involves building a thorough picture of the system and identifying all possible threats to the system. There are many different threat modeling techniques. The threat modeling technique used in this thesis is one outlined in the book *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices*. In their book, authors Guzman and Gupta, describe a threat modeling technique for IoT devices, which features a six step process [4]. This threat modeling technique is applied in

[1] https://www.securitashome.se/
[2] https://www.climax.com.tw/new/f1-features-new.php
[3] https://www.fcc.gov/oet/ea/fccid

this thesis. More on this in section 2.2. The threat model is presented in chapter 5.

2.1.4 Vulnerability Analysis

The next step of the methodology involves taking the threats identified in the threat model and trying to find which of them present actual security vulnerabilities in the system. The way this was done varied greatly depending on the type of threat that was examined. For the threats relating to RF communication, for example, this involved physically capturing radio signals using a Software-defined Radio (SDR). For other threats, like investigating insecure network services, this involved using vulnerability scanners like *Nmap*[1]. Additionally, manual analysis was a big part of most of the vulnerability analysis. More about the practical penetration testing methods used in this thesis can be found in chapter 6, under the method section of each penetration test.

2.1.5 Exploitation

After a certain vulnerability in the system was confirmed, attempts to exploit the vulnerability were made. These were done to try and assess the consequences of the vulnerability and what type of results an attacker could achieve by exploiting this vulnerability.

2.1.6 Post Exploitation

Several vulnerabilities were found as a result of the previous two steps. However, none of them presented any opportunities for post-exploitation. As such, this section of the penetration testing methodology was not applied in this thesis.

2.1.7 Reporting

The identified vulnerabilities were reported to Securitas. Responsible disclosure was applied using the method of Coordinated Vulnerability Disclosure (CVD)

[1] https://nmap.org/

[9]. This is a process that aims to both handle the vulnerability responsibly and let the manufacturer produce a security patch, while at the same time making sure the vulnerability is publicly disclosed so as to increase the collective knowledge of our society within cybersecurity and make sure consumers can make an informed decision.

Securitas was contacted through their customer support, as well as their official responsible disclosure form[1]. They were given 90 days to fix and respond to the vulnerability before the report was published, as is the industry standard. However, the thesis presentation was held before that, as that was a requirement from KTH. Due to the nature of the identified vulnerabilities, withholding information on how to perform them was not possible during the presentation. Instead, participants were asked to not spread the information further until the thesis was published.

Additionally, two vulnerabilities were reported to MITRE and two CVEs were requested.

2.2 Threat modeling for IoT devices

Threat modeling is a wide field. According to authors Xiong and Lagerström the field lacks a common ground [10]. There are competing definitions of the term, and they conclude that it is used in many different and sometimes competing ways. However, the definition used in this report is the following, as suggested in their report: "threat modeling is a process that can be used to analyze potential attacks or threats, and can also be supported by threat libraries or attack taxonomies". The authors also note that there are two categories of threat models, *manual/automatic* and *formal/graphical* threat modeling. The former category is applied in this thesis. Lastly, some threat models aim to be general and others more specific to the type of system examined Xiong and Lagerström. This thesis uses a threat modeling technique of the latter category, which is specified towards IoT systems.

In their book *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices*, authors Guzman and Gupta, outline a six-step process for threat modeling IoT devices. This is the threat modeling technique applied in this thesis, the result of which can be found in chapter 5. There are

[1] https://www.securitas.com/en/about-us/responsible-disclosure/

many different threat modeling techniques and this one was chosen due to it being customized specifically for IoT devices. Here are the six steps of the process listed and described:

1. *Identify IoT assets*. This initial step involves identifying all assets of the system you have an interest in protecting. Essentially, this list should include anything that could be a target or something negatively affected by an attack. This aids in identifying what an attacker might focus on when crafting an attack or pentesting the system. Can be presented as a simple table describing each asset.

2. *Create an IoT Device Architecture Overview*. Once all assets have been identified, the next step involves creating an architecture overview. In their method, Guzman and Gupta includes three components of this breakdown. First is a list of all use cases of the system. This details what a regular user can do with the system and the steps of each. Second is a diagram of all components of the system, and how they communicate (what protocol for example). Components can include everything from external cloud servers, network equipment, and individual processes running on a machine. This is usually presented visually, in a diagram depicting the system. Lastly, a list of all identified technologies used in the system is included. This list should include everything from operating systems, network protocols, and known applications. Any additional information about each, such as version number, for example, should be included if known.

3. *Decompose the IoT Device*. This step includes analyzing the application and protocol data flow of the system. Using this, one aims to locate and detail all potentially vulnerable entry points of the system, either into physical devices or client applications. Additionally, entry points of higher privilege access should be noted. An entry point could for example be a hosted web server on an IoT device or an open TCP port, the firmware of the device, or a mobile application.

4. *Identify Threats*. In this phase, the threats to the system are identified. The documented data flows aid you in identifying potential threats. One can use a model of threats like STRIDE [11] to identify threats of different characteristics, see section 2.3. In their method, Guzman and Gupta proposes adding two categories to the STRIDE model for IoT-specific threats. The first one is *Physical Security Bypass*, which involves vulnerabilities caused by the attacker having physical access to

the device for a limited period of time. The second category is *Supply Chain Issues*, which includes threats to the various technologies that the system relies on. This could be documented vulnerabilities in the hardware platform for example. Note that due to the delimitations of this report (see section 1.4) the former introduced category is not included in this thesis.

5. *Document Threats*. A few of the identified threat use cases will be documented in this step. For each, the target of the threat, the attack technique, and the potential countermeasures should be noted.
6. *Rate the Threats*. For each documented threat, one rates its severity. The authors propose using a system like DREAD to give a score to each threat, based on a few different criteria like damage potential and reproducibility.

The authors suggest as a final step rating the threats using the DREAD model. However, due to the time constraints of this project that was deemed excessive. There were more promising identified threats than there was time to explore them all within the scope of this thesis. Therefore, intuition was relied up on to determine which threats to explore by answering the following questions:

- How likely is it that the threat can be successfully exploited?
- What is the potential impact of the system being vulnerable to the threat?

2.3 The STRIDE model

STRIDE is a model of threats used to identify and categorize threats to the cybersecurity of an IT system [11]. It was initially developed by Praerit Garg and Loren Kohnfelder at Microsoft as part of their threat modeling technique. It is a widely used mnemonic in the security industry to aid in recognizing threats[1]. The following are the six properties that make up the acronym and a description of each:

- **Spoofing Identity**. This means impersonating the identity of another user or component of the system. One example is obtaining a user's

[1] https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats

login credentials and posing as that user, or performing a *Man in the middle* attack and posing as an external trusted server.

- **Tampering with Data**. This means modifying data in the system in a malicious manner that you were perhaps not meant to modify. An example is unauthorized modifications to data stored in a database.
- **Repudiation**. This means being able to claim you did not perform a certain action. An example would be to somehow be able to claim a transaction did not go through, thereby illegally receiving additional payment, or deleting log files.
- **Information Disclosure**. This means exposing information or data to users who are not meant to have access to it. An example is a database leak.
- **Denial of Service**. This means denying access to a service. An example is making a web server unavailable to users by hitting it with heavy traffic, perhaps via a distributed Denial of Service (DoS) attack.
- **Elevation of Privilege**. This means an unprivileged user gaining privileged access to the system, allowing them access to features of the system they were not meant to. An example is exploiting a vulnerability in a program or kernel to get access as a more privileged user on the machine like `root`.

In conjunction with the OWASP top ten list of vulnerabilities in IoT systems and the ETSI EN 303 645 standard (see sections 4.1 and 4.2), this model was used to identify and categorize the threats against the system. This is presented in section 5.4.

Chapter 3

System Under Consideration

The following chapter gives a thorough explanation of the system under consideration in this thesis. Firstly, the process of selecting what system to investigate is explained. The system in question is the *SecuritasHome startpaket*[1], which is a home alarm system from Securitas. This starter kit includes multiple hardware components, as well as access to software portals and mobile applications to control the system.

3.1 Selection of the system

When starting this project, there were many aspects to consider regarding what system to investigate. The first aspect was *impact*. If the security was compromised what consequences would that have? Hacking a doorbell might lead to annoyances, while hacking a smart pacemaker could have lethal consequences. Smart home alarm systems have the potential for a huge impact. Gaining physical access to a house without any alarm system is not difficult. It is as easy as breaking a window. People, therefore, rely on alarm systems to deter criminals and to immediately notify security personnel of a breach. The worst consequence of a cybersecurity vulnerability in such a system would be to completely disarm an armed system, without authorization, thereby granting easy access to a property without, potentially, getting caught.

Another aspect to consider is *vulnerability*. What is the likelihood to find a vulnerability in the system? There are several things to consider to answer

[1] https://www.securitashome.se/product.html/securitashome

that question. One is the reputation of the manufacturer. Are there many previously reported vulnerabilities on their systems? Are they known for caring about security and producing secure products? Another is how complex the system is. Does it have many features and components? How large is the system's attack surface? Smart home alarm systems have become increasingly complex, leading to a relatively large attack surface.

Due to the reasons mentioned above, smart home alarm systems were deemed an excellent target for this thesis. The last thing to consider was therefore how feasible it is to procure the system, and importantly if one has the legal rights to security test it. In Sweden, the two largest companies in the industry are *Verisure*[1] and *Sector Alarm*[2]. Systems from these companies were considered in the early phase of the project and both were contacted through their customer support to assess the feasibility to procure the system and if it would be legal to do a security evaluation. Both companies, unfortunately, failed in this criteria. In Sweden, the law regarding cybersecurity evaluations, in simplified terms, says you have to have the owner's permission to security test a system [12]. Both companies stated firmly that they would continue to own the physical system and require that their technicians install the system on the premises. Both seemed unwilling to let you buy out the system, and were therefore ruled out. In the case of Securitas, on the other hand, you own the physical system and pay a monthly fee to access their software platform and security personnel in case of a breach. This, along with the aspects described above, made Securitas home alarm system a good target for security analysis.

3.2 The companies behind the system

This section covers the structure of the three major companies behind the SecuritasHome Home Alarm System.

While the system is sold and branded by Securitas, they actually have little to do with the hardware and software components of the platform, see figure 3.1. The hardware, related firmware, and proprietary radio wave protocol for communication between the components are manufactured and produced by a Taiwanese company called *Climax Technology*[3]. They are a major manufacturer of wireless home security systems and produce hardware for

[1] https://www.verisure.se/
[2] https://www.sectoralarm.se/
[3] https://www.climax.com.tw/

home consumer security. They design everything from smart home alarm systems, and smart garage door openers, to smart medical accessories for seniors. The software, like the web portal and mobile applications as well as some additional firmware, is developed by an American company called *Alarm.com*[1]. They are strictly a B2B (business to business) company, meaning they do not sell or advertise their product directly to the end consumer. Instead, they outsource the sale and advertisement of the system to partner companies, one of them being Securitas. Securitas sell, advertise, and put their brand on the product. Their main contribution to the system is in terms of real-time response to an alarm, customer service, technical support, and sending security personnel to respond to an active alarm breach. Consequently, when considering the cybersecurity of the system, Securitas is not highly relevant. The two relevant parties are *Alarm.com* and, considering the focus of this thesis, especially *Climax Technology*.

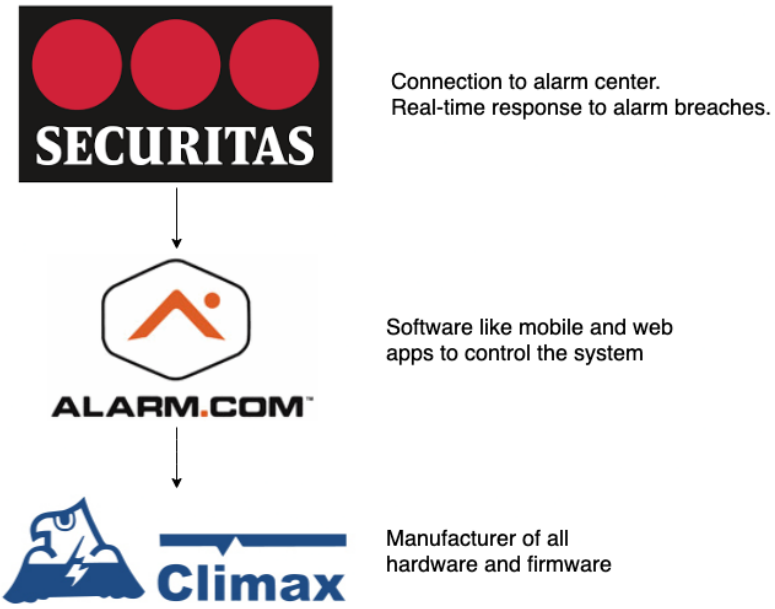


Figure 3.1 – The companies behind the system.

[1] https://alarm.com/

3.3 Components and Software

This section describes all the components and software of the system. Initially, all hardware components are described and their functionality. Lastly, all software components of the system are described.

3.3.1 Hardware components

The SecuritasHome starter kit contains five hardware components, see figure 3.2. These are described below. Note that the system supports many additional components, like smart locks for example.

(a) Main Panel (b) Remote Keypad (c) Motion Detection Camera

(d) Door Contact Sensor (e) Smoke Detector

Figure 3.2 – The hardware components of the system.

Main Panel

Model number: HSGW-G8-3G/LTE-ZW-F1 433/868
FCCID: GX9HSGWF1919
The main panel, see figure 3.2a, is the *"brains"* of the system so to speak. It handles communication with all other hardware devices as well as external servers. Through radio wave communication it communicates to all the other hardware peripherals of the system. It uses 3G telecommunication to communicate with external servers. It also has an Ethernet port to connect to the local network, the purpose of which is unclear but according an FAQ on SecuritasHome's webpage[1] it is required during the installation process. The panel also has a tamper sensor, which triggers if the plastic back panel is taken off, to protect against physical attacks. Lastly, the panel is powered through a cable but also features a backup battery. This allows the system to continue functioning for several hours in the event of a power outage.

Remote Keypad

Model number: KPT-23-EL-F1
FCCID: GX9KPF1
The remote keypad is a 16 button keypad used to arm and disarm the system using a personal 4 digit pin. See figure 3.2b. This device communicates with the main panel over radio wave communication. It features no tamper sensors.

Motion Detection Camera

Model number: VST-862-F1
FCCID: GX9862
This device, see figure 3.2c, has an infra-red sensor to detect motion, and a camera to survey the location. When triggered the device takes two pictures which are sent to the main panel. It is not a surveillance camera, meaning it does not continuously take pictures. The camera is only active when motion is detected and the alarm is triggered, presumably to save power. The camera also features a tamper sensor, which triggers if it is not properly attached against the wall.

[1] https://www.securitashome.se/faq.html

Door Contact Sensor

Model number: DC-23-F1
FCCID: GX9DC23
This device, see figure 3.2d, senses when a door or window is opened. A small external magnet is placed on the door/window close to the device. When these are separated the device is triggered and communicates with the main panel over radio wave communication. Like the camera, it also features a tamper sensor, which triggers if it is not properly attached against the wall.

Smoke Detector

Model number: SD-8EL
FCCID: GX9SD8ELF1919
This device is a smoke detector, see figure 3.2e. It communicates with the main panel over radio waves and also includes a siren that triggers when it detects smoke.

Additional peripherals

The system supports many additional devices not included in the *SecuritasHome startpaket*. These include for example water leakage sensors[1], temperature sensors[2], IP cameras over both WiFi[3] and PoE[4], and smart locks.

Additionally, the system supports controlling devices over the Z-Wave protocol[5]. This allows a whole plethora of devices to be connected to and controlled by the system such as smart light bulbs for example. This includes devices from completely different manufacturers.

Note that the devices covered in this section are delimited from this project and thus not covered in the report.

[1] https://www.securitashome.se/product.html/vattendetektor-2
[2] https://www.securitashome.se/product.html/temperatursensor
[3] https://www.securitashome.se/product.html/ip-kamera-inne-wifi
[4] https://www.securitashome.se/product.html/ip-kamera-ute
[5] https://en.wikipedia.org/wiki/Z-Wave

3.3.2 Software 1: Web portal

The web portal is a web page created by the American company *Alarm.com*, see figure 3.1, hosted at `https://www.alarm.com/web/system/`. From the landing page, see figure 3.3, the user can do the following:

- See if the system has any issues. This can be seen in the *System OK* box in figure 3.3.
- See the state of each sensor of the system, like the door contact sensor (see figure 3.2d).
- See the arm/disarm state of the system.
- See the latest photograph taken by the motion detection camera (see figure 3.2c).
- Arm and disarm the system remotely.
- Request a photo from the camera to be taken either directly or on the next detected movement.

Crucially, from the landing page, the user can easily arm or disarm the system, see figure 3.4. Beyond this, the user can also see a list of recent activity in the system, change the user's personal four-digit pin codes, and create new users.

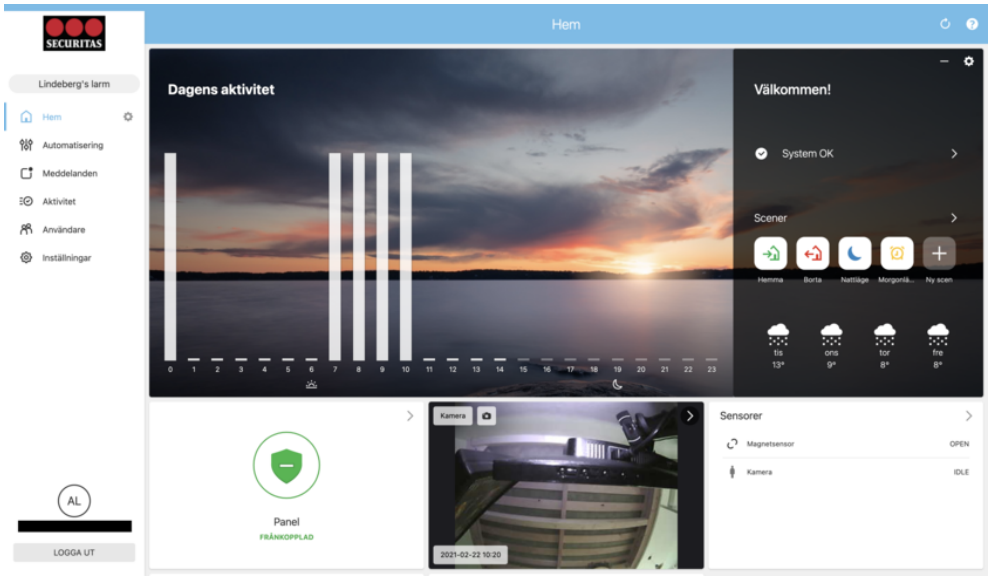


Figure 3.3 – The web portal landing page.

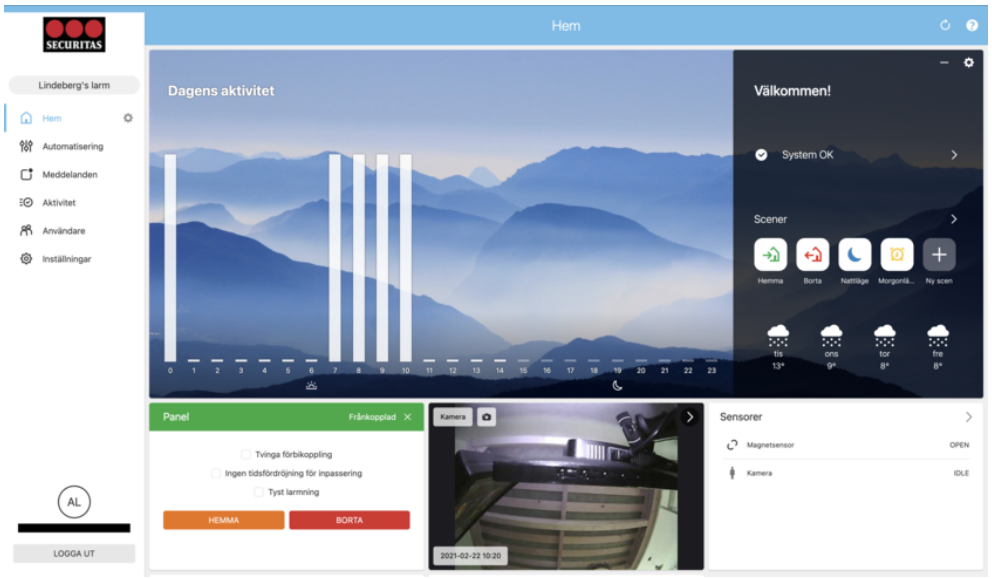


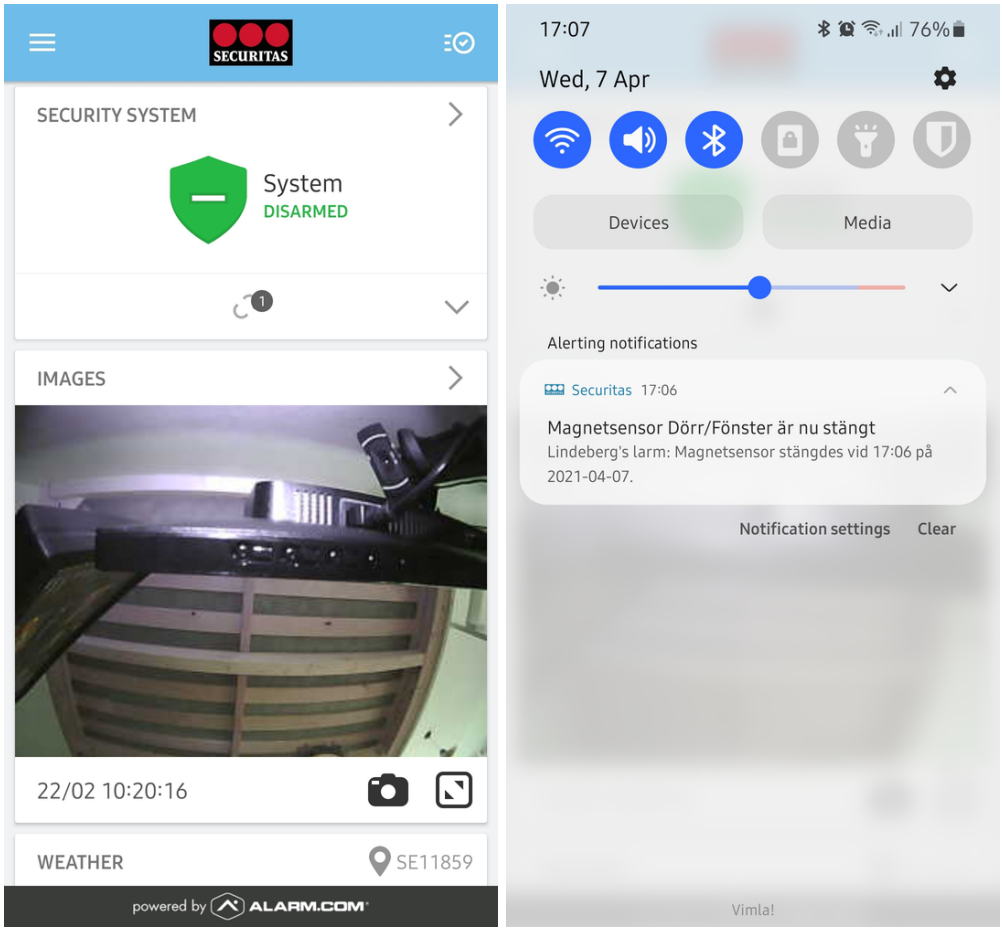
Figure 3.4 – Arming the alarm from the web portal.

3.3.3 Software 2: Mobile application

The system can also be controlled and administrated via a mobile application (see figure 3.5), free to download via the *Google Play Store*, called *Securitas Connect*[1], and from the *App Store*, for iOS devices, under the same name[2]. As explained previously, while the application is branded by Securitas, it is developed by *Alarm.com*. The interface very closely resembles the web portal, see figure 3.5a, and offers identical functionality. The only additional feature is real-time notifications of any changes in the system, done via push notifications to your mobile device, as shown in figure 3.5b. This could be for example the door contact sensor triggering, the main panel losing external power due to a power outage, or the alarm being armed/disarmed. These are also sent as an email to the user.

[1] https://play.google.com/store/apps/details?id=com.alarm.alarmmobile.android.securitas

[2] https://apps.apple.com/se/app/securitas-connect/id1111700213



(a) The mobile app's landing page.

(b) A notification triggered by the door contact sensor.

Figure 3.5 – The Securitas Connect mobile app.

3.3.4 Software 3: Local web admin page

Beyond the two applications created by *Alarm.com* described above, the main panel (see figure 3.2a) hosts a web server on the local network. This feature is mostly undocumented and not presented to the user during the installation process. It is presumably not meant to be used or found by the regular, non-tech-savvy consumer. The page is not hosted on any domain name, as far as the author is aware, and instead has to be accessed directly via the main panel's local IP address on port 80. The landing page of this web server, see figure 3.6, is quite simple and shows some basic debug information about the system such as the MAC address, IMEI number of the cellular communication, etc.

Beyond that, the site only has two actions the user can do. One is to perform a "*Phone Test*", to presumably test the connection to the mobile 3G network, and the other is a "*Network Scan*". Once the network scan has completed, the page shows a list of all reachable telecommunication towers, see figure 3.7.

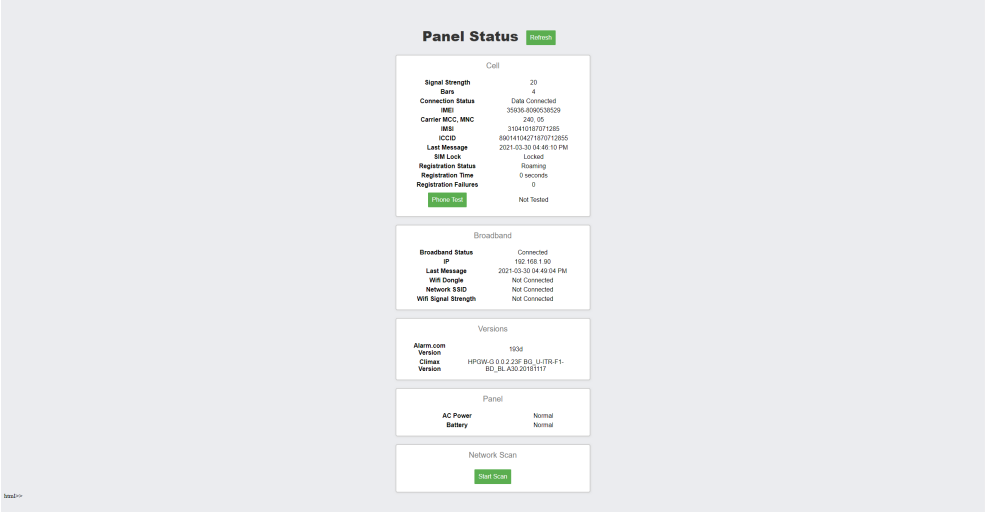


Figure 3.6 – The local web server's landing page.

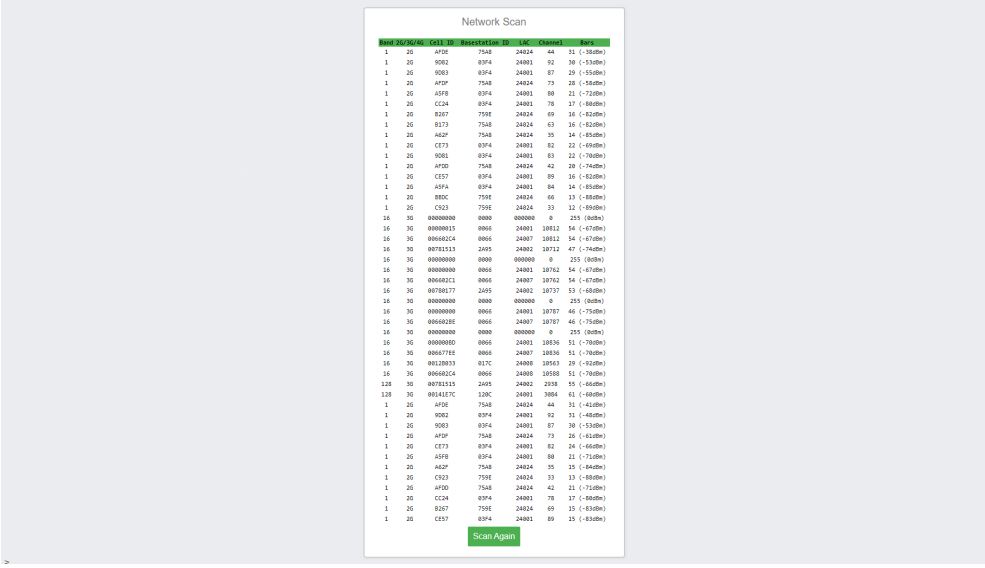


Figure 3.7 – A mobile network scan from the local web server.

Additionally, the web page features an undocumented login page on the path `/action/login`, see figure 3.8. This page lets the user authenticate using *HTTP Basic auth*. The credentials here are not tied to the user's account on the aforementioned web portal or mobile applications. Presumably, this is purely meant as a backdoor for the manufacturer for debugging purposes, and not meant to be used by the user of the system. If wrong credentials are entered the user is presented with a static message saying "Access Denied: Wrong Password!".

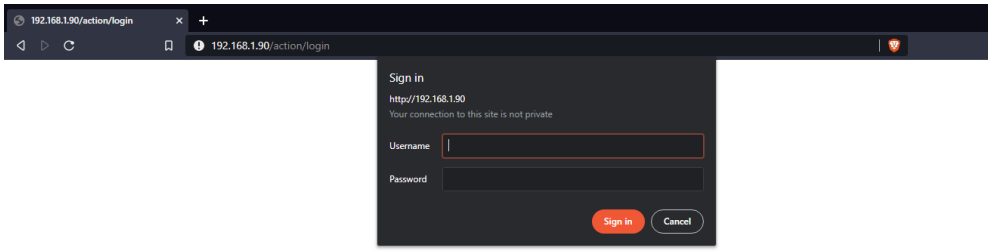


Figure 3.8 – The local web server's HTTP Basic Auth login page.

Chapter 4

Related work

This chapter describes identified related works done in the same area and specifically against the hardware used in the system. The OWASP IoT Top 10 is described as well as its relevance to this report. Additionally, the ETSI EN 303 645 standard for the development of secure IoT products is covered. Regarding the system under consideration, very little has been published regarding the cybersecurity of this specific IoT system, as far as the author is aware. However, two sources that explore systems based on the hardware from the same manufacturer, Climax Technology, have been identified. Additionally, a student thesis was done on a similar system but from a different manufacturer. These are presented below. Lastly, two talks on RF hacking are presented. These present common techniques in RF hacking and what types of vulnerabilities are common in RF communication.

4.1 OWASP IoT Top 10

The OWASP top 10 for web vulnerabilities is one of the most used sources of the most common web vulnerabilities [13]. The OWASP Foundation, the publisher of the report, claims it is "globally recognized by developers as the first step towards more secure coding"[1]. OWASP also compiles and publishes a list of the top ten most common vulnerabilities for specifically IoT systems [5]. The latest revision was released in 2018. This list is highly relevant to this report as the system in question is an IoT system. In this thesis, this list was

[1] https://owasp.org/www-project-top-ten/

used to identify and determine the most important threats to investigate. The top ten vulnerabilities for IoT systems according to OWASP's report are the following, in order of importance:

1. **Weak, Guessable, or Hardcoded Passwords**. This includes passwords that can be easily brute-forced, are publicly available, or unchangeable. Famously the MIRAI botnet, which mostly included IoT devices, managed to recruit over half a million devices to its botnet by testing just 60 default credentials [14].

2. **Insecure Network Services**. Often unneeded network services will be left open on IoT devices, even after they are shipped to customers. Commonly, these are used during development but are never removed when the device is shipped. An example is open telnet services or ssh being left open, leaving a backdoor open for the manufacturer. However, as a consequence, these network services open the device up to a whole slew of potential vulnerabilities. Often they are also completely unnecessary for the functionality of the IoT system. A real-world example of this is again the MIRAI botnet, which scanned IP ranges for devices with telnet services hosted on port 21 or 2121 [14].

3. **Insecure Ecosystem Interfaces**. Many IoT systems are much more than just the device itself. There is often an entire infrastructure behind it to control and interact with the system, such as a mobile application or a website, with an accompanying backend API server. Vulnerabilities anywhere in this ecosystem can lead to the security of the IoT device itself being compromised.

4. **Lack of Secure Update Mechanism**. To add new features and to fix security issues, updating the firmware of IoT devices is often unavoidable. However, over-the-air (OTA) firmware updates add an additional attack vector and can introduce many potential security issues. This can include the device not validating the firmware it receives, delivering the firmware unencrypted in transit, a lack of a mechanism to prevent rolling back changes, and a lack of notifications of security changes due to updates.

5. **Use of Insecure or Outdated Components**. This means using old versions of libraries and components that themselves are vulnerable. These can then in turn compromise the security of the system as a whole. This can include everything from insecure versions of operating systems, insecure customization of the OS, third-party software, and

insecure hardware components from higher up in the supply chain.

6. **Insufficient Privacy Protection**. This refers to the user's personal information being stored in an insecure way. Either on the IoT system itself or in the larger ecosystem.
7. **Insecure Data Transfer and Storage**. Lack of encryption or proper access control, both in transit and in storage. This can, once again, be both on the IoT device and anywhere else in the larger ecosystem.
8. **Lack of Device Management**. This includes lack of secure asset management, update management, secure decommissioning of the system, proper monitoring of the system, as well as capabilities to respond to a security issue.
9. **Insecure Default Settings**. This refers to the device being sold with insecure default settings. Often the user is never prompted or encouraged to change these insecure defaults.
10. **Lack of Physical Hardening**. During development, debug hardware interfaces are used. However, a common vulnerability is not removing these before shipping the product. This can include open JTAG or UART interfaces on the circuit board, allowing for the extraction of the firmware or even terminal access to the system over a serial interface. Often these vulnerabilities require physical access to the system. However, they can be used to extract valuable information which can help an attacker in developing remote exploits.

Note that some of these threats were not considered in this report. Some of these fall under the delimited areas described in section 1.4, such as number three for example. Others were not considered simply due to time constraints. However, the OWASP IoT top 10 list was the basis for the identified threats against the system, along with the STRIDE model described in section 2.3.

4.2 ETSI EN 303 645, a standard for IoT security of consumer products

The *European Telecommunications Standards Institute* (ETSI) is a standardization organization within information and communication technology (ICT). It is

responsible for standardizing key technologies like GSM, 3G, 4G, and 5G. One of the standards produced by ETSI is of relevance to this report. In their standard *CYBER; Cyber Security for Consumer Internet of Things: Baseline Requirements*, they list 13 provisions that should be followed by manufacturers selling consumer-facing IoT products [6]. While following the standard is certainly not a guarantee that the system is secure, it provides a set of best practices of common security issues to avoid in IoT systems. Within the provisions are sub-provisions which will not be covered in full detail here. What follows is an overview of the 13 provisions as outlined by ETSI [6].

1. **No universal default passwords**. All passwords have to be either supplied by the user or unique per device. No password should work across all devices. Additionally, they warn against schemes where the password is generated using publicly available information like the system's MAC address or WiFi SSID.

2. **Implement a means to manage reports of vulnerabilities**. The manufacturer should make publicly available a process for disclosing vulnerabilities. It should include contact information for disclosing security issues and information on timelines during a disclosure process.

3. **Keep software updated**. The manufacturer needs to implement a way of sending out security patches in a timely manner and make sure that this is done in reaction to discovered vulnerabilities.

4. **Securely store sensitive security parameters**. This could be encrypted storage or dedicated security components implemented in hardware for example. Additionally, it states that hard-coded critical security parameters in the device software should not be used.

5. **Communicate securely**. Best practices in encrypted communications should be used in all communication, including using reviewed and well-tested cryptography methods.

6. **Minimize exposed attack surfaces**. The standard recommends adhering to the *principle of least privilege*, meaning one should aim to give each actor the minimum privilege needed to perform its function. Additionally, it states that all unused or unnecessary network and logical interfaces should be disabled. Only software services that are absolutely required for the functionality of the device should be enabled.

7. **Ensure software integrity**. The system should verify its firmware and other software by, for example, using secure boot mechanisms.

Additionally, if a discrepancy is noted the system should report it to the user/administrator and should not connect to any wider network than the minimum required to send the notification.

8. **Ensure that personal data is secure**. Personal data in transit between devices should be properly secured using cryptography best practices.
9. **Make systems resilient to outages**. Resilience should be built into the system, in terms of handling outages both in power and in network services for example. In case of a network failure, the system should remain operational and keep the still viable functionality. In the event of a power loss, the system should recover cleanly.
10. **Examine system telemetry data**. If the system collects telemetry data it should be continuously monitored for anomalies with potential security implications.
11. **Make it easy for users to delete user data**. It should be clear to the user how they can delete their data and functionality should be provided giving them access to do so. Additionally, users should be given clear notification of when their data has been deleted.
12. **Make installation and maintenance of devices easy**. The installation and maintenance of the system should involve avoid decisions from the user as much as possible and follow security best practices on usability. Additionally, the manufacturer is responsible for providing adequate guidance on how to set up the system securely as well as how to check that it is set up securely.
13. **Validate input data**. The system has to validate all inputs from any interface it reads from. It should never assume that the input is benign.

4.3 Related work 1: *Examination of LUPUS-Electronics devices*

This section details a security analysis of a very similar system, built on similar hardware from Climax Technology. The study was conducted by members of *Embedded Lab Vienna for IoT & Security* (ELVIS)[1], a project at the University of Applied Sciences Campus Vienna in Austria.

[1] https://www.elvis.science/

Lupus Electronics is a German manufacturer of smart home security systems[1], much like *Alarm.com* in the system examined in this thesis (see section 3.2). Just like Alarm.com, Lupus Electronics mostly provides the software of their system and also purchase hardware from the Taiwanese manufacturer *Climax Technology*. In 2019, security researchers at ELVIS examined the security of the *XT2 Plus Main Panel*. According to their report, similar hardware to the system examined in this thesis is used in that system. They reported several vulnerabilities. The most critical vulnerability they found was a Telnet server hosted on an unconventional high TCP port on the main panel. After examining the firmware of the system, and reverse engineering one of the applications, the researchers found that the password to the `root` user could be derived from a hardcoded salt and the MAC address of the panel. This meant that as long as an attacker had access to the local network, they could log in as root on the device, meaning with full privileges. With those privileges, one could easily bypass the security of the alarm completely.

While the Lupus system is not identical to the one in this thesis, much of the firmware from the hardware manufacturer is presumably the same. This assumption is strengthened by the fact that the endpoints found in the web interface of the Lupus system are the same in the thesis' system. The system under consideration in this report does not host a telnet server, however, it does host an application listening on a *non-standard* high TCP port. According to their report, Climax Technology was notified about the vulnerabilities on `2019-01-09` and a firmware revision fixing the vulnerabilities was released on `2019-03-26`.

4.4 Related work 2: *The Internet of Things: a privacy label for IoT products in a consumer market*

In their master thesis, author Diermen examines the design of an IoT privacy label, to help consumers recognize the security risks of the products they bring into their home [16]. The main objective of their thesis, while interesting, is not strictly relevant to this report. However, the thesis includes three case studies of different consumer products, one of them being highly relevant to this project. They examine the security of a home alarm system from

[1] `https://www.lupus-electronics.de/en/`

Egardia[1], a dutch company producing smart home alarm systems. Much like *Alarm.com* or *Lupus Electronics* (see section 4.3), Egardia are mainly responsible for the software platform and base their product on hardware from Climax Technology. While this system is not the main focus of their thesis, the author presents a short security analysis of the system and reports a few vulnerabilities. One of them is a replay attack vulnerability in the RF communication between the remote keypad and the central panel. The author demonstrates how using a SDR to capture the RF traffic of the user arming and disarming the system, can be simply replayed. The system is then armed/disarmed without issue. There was to be no mechanism in place to prevent this type of attack. While the author labeled this a medium-level security vulnerability, it has the potential to completely subvert the functionality of the system. All it requires is physical proximity to the system and a SDR, which can be bought by anyone for about 350 USD[2]. Additional vulnerabilities in the system were related to the Egardia software platform and are therefore not relevant to this thesis.

4.5 Related work 3: *How Secure is Verisure's Alarm System?*

In their thesis, authors Hamid and Möller examine the cybersecurity of a home alarm system from *Verisure*. The company claims to sell the most widely installed home alarm in Europe[3], installed in over 350 000 homes in Sweden alone. Their thesis mostly focuses on the SCTP communication between the main panel and the external Verisure servers, as well as the web security of the Verisure software platform. The authors found several Cross-site request forgery (CSRF) vulnerabilities, allowing an attacker to disarm the system and create new users.

The examined system from Versiure, from the point of view of a user, is similar to the one in this report. It offers almost the same components and features. From a technical perspective, however, the two systems are quite different. There is no overlap in the hardware components and much of the network technologies used are different. For example, the main panel in the Verisure system communicates over a broadband connection using the SCTP protocol,

[1] https://www.egardia.com
[2] https://greatscottgadgets.com/hackrf/one/
[3] https://www.verisure.se/english

in contrast to the system in this thesis that uses 3G telecommunication. Their work does, however, show that the cybersecurity of similar products in the industry may be lacking.

4.6 Related work 4: *Hacking The IoT (Internet of Things) - PenTesting RF Operated Devices*

At the 2016 AppSecIL conference, an OWASP hosted a conference on cyber security[1], Erez Metula held a talk on RF hacking [18]. In their talk, they outline the basics of RF hacking, what tools you need, as well as how to approach reverse engineering a custom RF protocol. Metula is an application security expert, author of the book *Managed Code Rootkits*, and a member of the OWASP IL Board.

Initially, the presenter talks about the common architecture of IoT systems. The IoT system will normally communicate with an external cloud server and some application that lets the user control the system, such as a mobile app or website. Additionally, for RF systems RF communication is used between local components. Furthermore, the concept of Software Defined Radio's (SDR) is covered. Normally radio components will be implemented in hardware. However, these are physical devices that, through software, can emulate any radio component. This allows for much more flexibility and allows a hacker to tune it to the system under consideration. Metula discusses controlling an SDR using the open-source program *GnuRadio Companion*, which lets the user create visual flow graphs to control an SDR.

The presenter goes on to discuss *replay attacks* in RF communication. These are "zero knowledge" attacks, requiring no knowledge from the attacker about how the devices communicate, what protocols they use, etc. Instead, the attacker simply records a signal and replays the exact same signal later. If successful the system will recognize this as a valid signal and perform the same action again. The disadvantage of this type of attack, according to Metula, is that one cannot create a valid message from scratch, and one cannot tamper with and change the message.

To gain further insight into how the RF protocol works, Metula describes a seven-step process to analyze the RF traffic to figure out the structure of the

[1] https://appsecil.org/

protocol and to be able to construct and transmit new messages:

1. **Information gathering**. Initially, one needs to find out more information about the system. The presenter suggests using OSINT techniques to gain more information. For example, usually, the FCCID is written on the back of the device, or it can be found by looking at the manufactures website. By using publically available FCCID databases[1], one can find a lot of information about the RF aspects of the system. If one has the tools, Metula also suggests doing some hardware research. One can open up the device and look for serial/UART/JTAG interfaces on the PCB, which can potentially let you extract the firmware from the device for example.
2. **Frequency**. Next, one needs to find the operating frequency of the RF communication. Often this can be found in the FCC documentation, or one can use a spectrum analyzer to capture traffic and see where the devices communicate.
3. **Modulation**. To transmit binary data over radio waves, a process called modulation is used. Finding out which modulation technique is used is vital to be able to extract the binary data out of a captured signal. This can either be found from official documentation, e.g the FCC documentation or user manuals, or by visually inspecting the captured signals.
4. **Deviation**. If the system uses Frequency-shift keying (FSK) modulation then one needs to find the distance between the two frequencies.
5. **Preamble/syncword**. According to Metula, RF protocols will often start with a preamble. The preamble is an alternating series of zeroes and ones. The purpose of the preamble is to sync the two devices and to let the receiver know that a message is starting. Following the preamble is often a syncword. This is a static string that indicates the start of the data section. It can also signify what protocol this is, or from what device it is. This can let the receiver drop the message if it is not intended for them, for example, if a system in the same proximity is communicating on the same frequency range.
6. **Symbol rate**. Next one needs to find how many samples correspond to a single symbol, e.g a `1` or a `0`. If say you capture at a sample rate of `2 million samples/s`, each symbol might correspond to a few

[1] https://fccid.io/

hundred samples. This depends on the sample rate and the baud rate of the devices.

7. **Transmission**. From the previous steps, you have all the information you need to transmit a new signal. Demodulating the signal, one can see the data of the packets. Often IoT systems lack any form of encryption in RF communication and instead simply send the data in plain text. If that is the case then reverse-engineering the protocol and generating your own messages is not too difficult. Metula suggests creating a flowgraph in GnuRadio to transmit binary data according to the RF protocol. However, he notes that this is often quite complicated to create and is not very interactive. Another approach is using a command line RF tool like RfCat, however, it does not support all SDRs.

The presenter goes on to discuss a common attack against RF communication, *jamming*. In this type of attack, the attacker continuously emits noise with a high decibel at the operating frequency of the system. This lets one block the traffic between the devices. One could for example prevent the message of arming the system from ever reaching the main panel. Metula likens this to a type of DoS attack against RF systems. This can be done using an SDR with a very simple GnuRadio flowgraph. One can also use dedicated jammer devices which can jam much more effectively, however, those are generally illegal to own.

4.7 Related work 5: *RF Exploitation: IoT and OT Hacking with Software-Defined Radio*

At the 2019 RSA Conference, an annual conference on computer security[1], a talk was given on RF hacking of IoT devices [19]. The presenters were Agrawal and Mehta, a researcher from MIT Academy of Engineering and a security expert from the company Symantec respectively.

In the talk, they initially cover the IoT landscape, citing how the number of IoT devices is expected to grow 31 billion by 2020 and that the total IoT market is forecast to be valued at 520 billion USD by the year 2021. Additionally, they cover the typical IoT threat model, consisting of multiple components.

[1] https://www.rsaconference.com/

A controlling device, typically a smartphone. This device typically talks to a cloud service, which in turn is connected to a global network (e.g the internet). Lastly, the cloud service, using the global network, talks to and manages the "things". In this model, there is often a need for wireless communication and this is often achieved through radio frequency communication. The authors claim that the radio wave spectrum and RF communication is like the world wide web in the 90s, in terms of security, with obvious security flaws riddled all over the place.

The authors go on to outline a process for how to assess the RF security of a device. Initially, one has to establish how the device operates normally. How do they connect? What is the operating frequency? They present the phases of an RF attack as the following:

1. **Information gathering**. Use OSINT techniques like the FCC submission to find information about the device. If you are lucky this could include documentation detailing at what frequency the device communicates, with what modulation, etc.

2. **Frequency**. Using programs like GQRX[1] to analyze the frequency spectrum from a Software Defined Radio (SDR). By analyzing the spectrum while the devices are communicating one can find the center frequency.

3. **Modulation**. Modulation is the technique of encoding binary data in a radio carrier wave. There are three basic types. In this step, one has to figure out which modulation scheme the system uses by either visually inspecting the recorded signals from your SDR or through official documentation discovered in the first step.

4. **Transmission**. When you know the frequency and modulation type you can now send arbitrary signals to the device. They mention tools the GnuRadio[2] and the RfCat library[3] to do so.

Furthermore, the presenters then cover one of the most common attacks against RF communication, replay attacks. This type of attack is a “zero knowledge” attack, meaning the attacker needs no knowledge of how the protocol works or how the devices communicate. If vulnerable, an attacker can simply record a signal and replay it at a later time to achieve the result again. Another

[1] https://gqrx.dk/
[2] https://www.gnuradio.org/
[3] https://github.com/atlas0fd00m/rfcat

advantage they list is that it works even if the protocol is encrypted. To understand the protocol the presenters talk about demodulating the signal using the open-source audio processing tool Audacity[1]. By demodulating a captured signal you get out the binary data that is sent between the devices. By analyzing the resulting bit pattern one can start to understand how the protocol is structured and what kind of information is being sent.

The presenters list five common attacks against RF protocols. One of them is replay attacks, as covered in the previous paragraph. Another one is a jamming attack, where the attacker sends out noise on the same frequency band which blocks legitimate traffic. To simply listen in on traffic is another attack they cover. Since the radio frequency band is an open medium, one needs encryption to protect against these types of information leaking vulnerabilities. However, the fact that two devices *are communicating* is not possible to hide. *Wardriving* is another attack they discuss. This is common against WiFi networks for example where an attacker simply drives around in their car with an RF receiver and scans for open networks and active RF traffic. Lastly, they discuss an *evil-twins attack*. This is when an attacker sets up a device that mimics an access point in the network, for example, a GSM network tower.

This talk largely echoes the sentiment of the talk covered in section 4.6. Their method for analyzing an RF device is almost identical and they both mention the same type of attacks and threats, giving credence to both.

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

Chapter 5

Threat Model

This chapter contains the threat model established for the system under scrutiny, which was used to identify and document all threats to the system. The methodology and threat model technique is described in section 2.2.

5.1 Identified Assets

As part of the first phase of our threat modeling technique, assets of the system were identified. These can be found in table 5.1.

Asset description
Physical access to the house
Personal four-digit pin
Arm/disarm state of the system
State of triggers, like the sabotage sensors
Door contact sensor state
Authentication to the admin web application
Triggered alarm state
Login credentials to the local webserver

Table 5.1 – The identified assets of the system.

5.2 Architecture Overview

This section contains an architecture overview of the system. Included in this are three components presented below. First is a list of all identified use cases of the system. The second is a diagram visually presenting all components of the system, how they interact, and the data flow of the system. Lastly, a table of all identified technologies used in the system is presented.

5.2.1 Use cases

As part of the architecture overview, all use cases of the system were identified. These are the use cases a regular user would encounter when using the system normally. The system, from the perspective of a user, is quite simple. The use cases are documented in table 5.2.

Use case
The user arms/disarms the system via the remote keypad panel.
The user arms/disarms the system via the web portal.
The user arms/disarms the system via the mobile app.
The user receives a notification about a state change in the system.
The user requests a photo be taken by the camera.

Table 5.2 – Use cases of the system.

5.2.2 Architecture diagram

Figure 5.1 presents a visual overview of the system, from a technical perspective. All identified components of the system are included, as well as how they all communicate, with what protocol, etc.

5.2.3 System Technologies

Table 5.3 contains all identified technologies present in the system. There are undoubtedly additional technologies used but these are the ones identified and relevant.

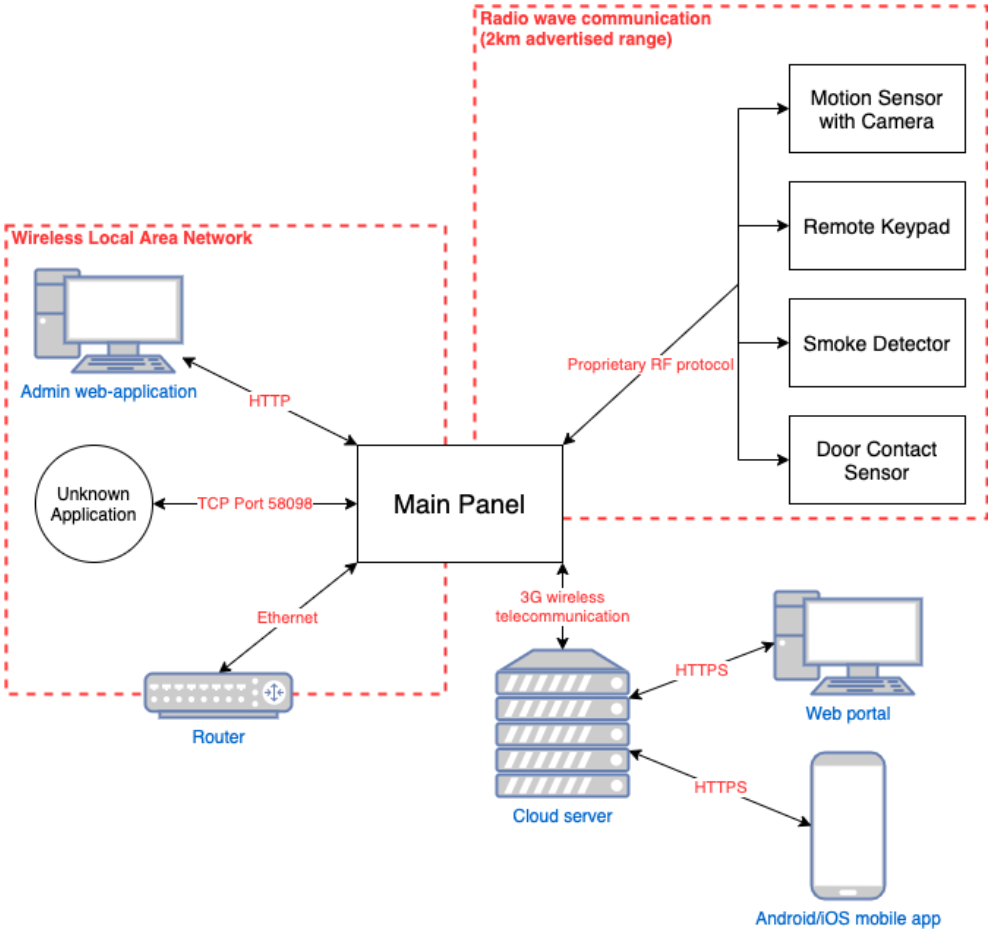


Figure 5.1 – A data flow diagram of the system.

Technology	Description
Main Panel	Linux 2.6-2.30. Hosts a web server over HTTP, using Mongoose (an embedded webserver), version unknown. Unknown application listening on TCP port 58098. Hosts DNS on TCP/UDP port 53. Has a USB and ethernet port.
HTTP	Protocol used by the admin web application. A clear text protocol used to communicate with the web admin panel.
F1 RF protocol	A proprietary RF protocol from the hardware manufacturer, Climax Technology. Uses 868 MHz frequency. This is an undocumented protocol, meaning no technical specifications have been publicized.
Mongoose web server	An open-source web server, in C. Used by the main panel to host the local admin web page, version unknown. Information leaked via the HTTP `Server` header on some endpoints.
ARM architecture	The CPU of the main panel is an ARM (little-endian) SOC system from *Grain Media*.
3G telecommunication	The main panel communicates with the external servers over the 3G mobile communication network.
TCP	The TCP network protocol is used in communication. The main panel listens on three different TCP ports (53, 80, 58098).

Table 5.3 – Technologies used in the system.

5.3 Decomposition of the system

This section presents the results of the fourth step of the threat modeling technique. This includes all identified entry points of the system. Considering the architecture of the system, and the delimitations of this thesis, only the entry points of the main panel are presented.

As part of the decomposition of the system, all entry points of the system were identified. These are essentially all points of contact an attacker could probe. The entry points are documented in table 5.4.

Entry point	**Description**
Local web admin page	See section 3.3.4. Provides very basic functionality but no control over the system. Has an undocumented login page via *HTTP Basic Auth*. Data is transferred over HTTP on the local network.
Unknown application	This is a completely undocumented process, listening on TCP port 58098. Does not send any response.
Main panel	The physical device features an Ethernet port to connect to the local network, as well as a USB port for unknown purposes. It communicates with other devices over an 868 MHz proprietary RF protocol.
3G telecommunication	The device has a SIM card and communicates over the 3G telecommunication network.
RF communication	The device talks to the other peripherals over a proprietary RF protocol, called *F1*. There seems to be little to no information available to the public about this protocol, other than its operating frequency of 868 MHz.
USB port	The device has a USB 2.0 Type-A connector.
Firmware	The firmware of the system.

Table 5.4 – The entry points of the main panel.

5.4 Identified Threats

The following section contains all identified threats. These are categorized after the threat categories of the STRIDE model. For an explanation of the model and a description of each category see section 2.3. Note the additional category *Supply chain issues*, which is a proposed extension to the model when threat modeling IoT devices [4] (see section 2.2). The *OWASP IoT top 10* [5] was referenced to identify common threats that the system might be vulnerable to (see section 4.1). In these threats, the OWASP IoT number is referenced as *OWASP IoT #n*. Similarly, the *ETSI EN 303 645* standard [6] for IoT manufacturers was used to identify threats (see section 4.2). These are referenced below as *ETSI #n*. Many of the threats related to the RF communication were identified through the references in section 4.6 and 4.7.

5.4.1 Spoofing Identity

- Spoof the remote keypad through the RF communication. (OWASP IoT #7)
- Spoof the door contact sensor through the RF communication. (OWASP IoT #7)
- Spoof the smoke detector through the RF communication. (OWASP IoT #7)
- Spoof the motion detection camera through the RF communication. (OWASP IoT #7)
- Spoof the remote keypad through the RF communication. (OWASP IoT #7)

5.4.2 Tampering with Data

- Replay attack on the RF communication through message blocking via jamming.
- Change and tamper with RF packets in transit.

- Send valid RF packets to the main panel, triggering events, without proper authorization.
  (OWASP IoT #7, ETSI #5)
- Make the main panel fall back to its Ethernet connection instead of 3G telecommunication.
  (ETSI #9)

5.4.3 Repudiation

- Replay attack on messages in the RF protocol between devices.
- Disrupt/disable logging of events.

5.4.4 Information Disclosure

- Information leak in the packets of the RF protocol.
  (ETSI #5)
- Weak or non-existent encryption in the RF protocol.
  (ETSI #5)
- Password sniffing on the local webserver.
  (ETSI #5)
- Leaking the existence of the system by continuous RF traffic.
- Extracting firmware from the main panel.
  (OWASP IoT #10)

5.4.5 Denial of Service

- Jamming of the RF protocol.
  (ETSI #5)
- Stop the alarm from triggering until after you've left the premises.
- Spamming traffic to the local webserver.
  (OWASP IoT #2)

- Crash the main panel or parts of the main panel.
  (ETSI #13)
- Crash the external devices through the RF communication.
  (OWASP IoT #3, ETSI #13)

5.4.6 Elevation of Privilege

- Insecure credentials used in the system.
  (OWASP IoT #1, OWASP IoT #9, ETSI #1)
- Password attack on the local webserver login page.
  (OWASP IoT #1, OWASP IoT #9)
- Send valid RF packets to the main panel without proper authorization.
  (OWASP IoT #7)
- Gain an authenticated connection to the main panel through one of the network services.
  (OWASP IoT #2, ETSI #6)

5.4.7 Supply chain issues

- Using official applications from Climax Technology to access the system, like their mobile app Vesta Home 5 [1].
  (OWASP IoT #3)
- Using debug applications from Climax Technology to access the system. These can be found on their website [2].
  (OWASP IoT #3)

[1] https://play.google.com/store/apps/details?id=com.climax.vestasmarthome.eu

[2] https://climax.com.tw/downloads/

Chapter 6

Penetration testing

This chapter details all penetration tests that were performed on the system. These were derived from the threat model created in chapter 5. All penetration tests are described in the following format. For some tests, the method part was omitted due to there being no method to describe.

- *Background*. Details the required background knowledge to perform and evaluate this penetration test.
- *Method*. Describes in detail how the test was performed, e.g in what environment, with what tools, what commands were used, etc.
- *Results*. Describes the findings of the penetration test.
- *Discussion*. This section contains a discussion about the reliability, validity, and generalizability of the results.

6.1 Lab Environment

This section describes the lab environment and physical setup used when performing the pentests described below.

For the pentests involving the RF communication between the devices of the system-specific hardware is required. Specifically, a Software Defined Radio (SDR) is required to be able to receive and transmit RF signals. The SDR

used in this report is the HackRF One[1] from Great Scott Gadgets[2]. This is a popular and relatively cheap SDR, costing around 350 USD. Additionally, an ANT 500 antenna was used since the HackRF One does not come with an antenna. As for the physical setup, the RF transmitting devices of the system were placed relatively close to the SDR, within `10-20 cm`. This was done to increase the quality of the captures. By placing them close clean signals could be captured without increasing the gain parameters of the SDR. Figure 6.1 shows a picture of the physical lab environment.

Figure 6.1 – The lab setup used to capture and replay RF signals.

Additionally, the pentests involving probing the alarm system over the local network were done using a laptop connected to the same WiFi network. During this, the main panel was connected to the WiFi router using an Ethernet cable.

[1] `https://greatscottgadgets.com/hackrf/one`
[2] `https://greatscottgadgets.com`

6.2 Task 1: Replay attack on the RF communication

This section details a penetration test against the RF communication between the hardware devices of the system. The specific attack vector explored in this section is a replay attack.

Background

A replay attack is an attack in network communication. The attacker listens in on the network traffic and simply retransmits the whole packets or information discovered in the packets to perform authenticated requests against a system they would otherwise not be able to. This type of vulnerability can have devastating consequences since even if several security measures have been taken, such as encrypting the data, the system can still be vulnerable to replay attacks. A replay attack is what is known as a "zero knowledge attack", meaning the attacker needs no knowledge of how the message is structured or what it contains, and can even work on encrypted protocols [18, 19]. This makes it often a very easy attack to perform.

Remote Keyless Entry (RKE) systems are notorious for being vulnerable to replay attacks. A famous example is RKE systems in car keys, used to unlock a car with from a distance a press of a button. When these first arrived on the market the security was extremely lacking and researchers in 2016 found that many cars on the market have used these insecure systems for over 20 years [20]. Often one-way communication from the key to the car over RF signals is used with no protection at all against replay attacks. Simply capturing the RF signal and replaying it would unlock the car, allowing an attacker free access whenever they please. A simple and well-tested protection against replay attacks is sending time-stamps along with your message. If the timestamp is older than some threshold when the receiver would reject the message [21]. This can, however, sometimes be difficult to implement in embedded systems since it requires both parties to agree on the time. For devices with an internet connection, this is easy thanks to the Network Time Protocol (NTP)[1]. However, most low-powered simple IoT devices, such as a car key, do not have internet connectivity. A solution many modern systems used is a rolling code scheme [18, 22]. Both systems keep an initially synchronized internal code C. When the transmitter sends a signal it includes this number and increments it

[1] https://tools.ietf.org/html/rfc5905

internally. The receiver stores the last such number it received and checks that the number in the new signal is within some interval $[C + 1, C + K]$, for some constant K. One usually accepts an interval of codes from the last accepted one in case a signal is lost or the button is pressed when you are outside the range of the car. If it is not within that range then the message is rejected. In practice, one might use this number to encrypt the message or use a sequence from a pre-defined pseudo-random number generator instead of an integer C.

This technique protects against replay attacks, however, it is still vulnerable against what is called a *rolljam attack*. By jamming the signal and at the same time recording it the attacker now has a signal they can replay to have one-time access to unlock the system. Most modern cars are still susceptible to this type of attack [18, 22].

Method

The physical setup used during this test is described in section 6.1. As stated, a HackRF One SDR was used to receive and transmit RF signals. While low-level protocols exist to control the HackRF One, one usually uses higher-level tools to interact with it. The method of this test builds on the open-source tool *Universal Radio Hacker* (URH), which was created by researchers at Hochschule Stralsund – University of Applied Sciences in Germany [23].

Initially, the center frequency at which the system communicates had to be found. This was done using the URH's *Spectrum Analyzer* tool, see figure 6.2. In the menu, *HackRF* was selected under devices and the frequency band to listen to was selected. We know from the official documentation that the system communicates at `868 MHz` 2020. Initially, that frequency was set in the menu options. The rest of the parameters were not important and left at their default values. To get the system to transmit data over RF communication, the tamper sensor located on the back of the door sensor was repeatedly pressed and released. While doing this, one could see a clear spike in the frequency spectrum and deduce that the center frequency used by the system is approximately `868.64 MHz`. This is clearly visible in figure 6.2. The result makes sense since the frequency band `868.6-868.7 MHz` is allocated specifically for alarm systems in Europe [25], as regulated by ETSI

[1].

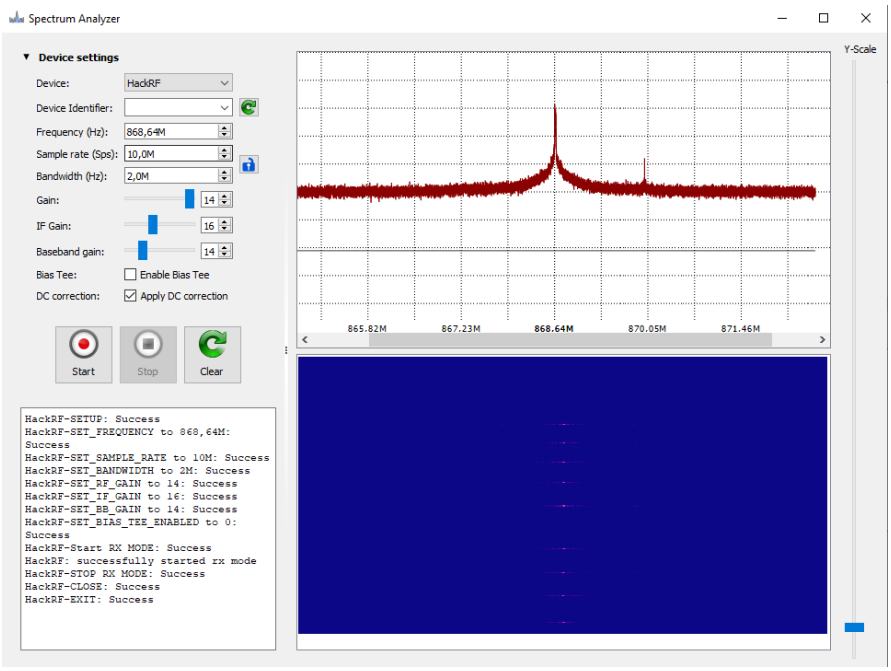


Figure 6.2 – Finding the center frequency of the RF communication.

When the center frequency had been found, signals could be captured using URH's *Record Signal* tool, see figure 6.3. This tool records raw audio signals captures from an SDR. In the menu options, one has to select the correct frequency. After some trial and error, `868.638 MHz` was found to yield well-captured signals. By placing the HackRF close to the devices, the SDR was able to capture clean signals without increasing the gain parameters. See figure 6.1 for the physical setup used during this test. By zooming into the captured audio signals and seeing clear sinusoidal waves with a non-trivial pattern, one could verify that the capture was most likely done correctly. Figure 6.4 shows a magnified view of a good signal captured from the door sensor's temper alarm. This process was repeated for all identified communication endpoints of the system, capturing their signals.

Lastly, once signals had been captured they could be replayed using URH's *Send Signal* tool, see figure 6.5. The tool lets you easily select and edit which parts of the captured signal to replay. Through a trial and error process, several combinations of packets were tested as a replay attack.

[1] `https://www.etsi.org/`

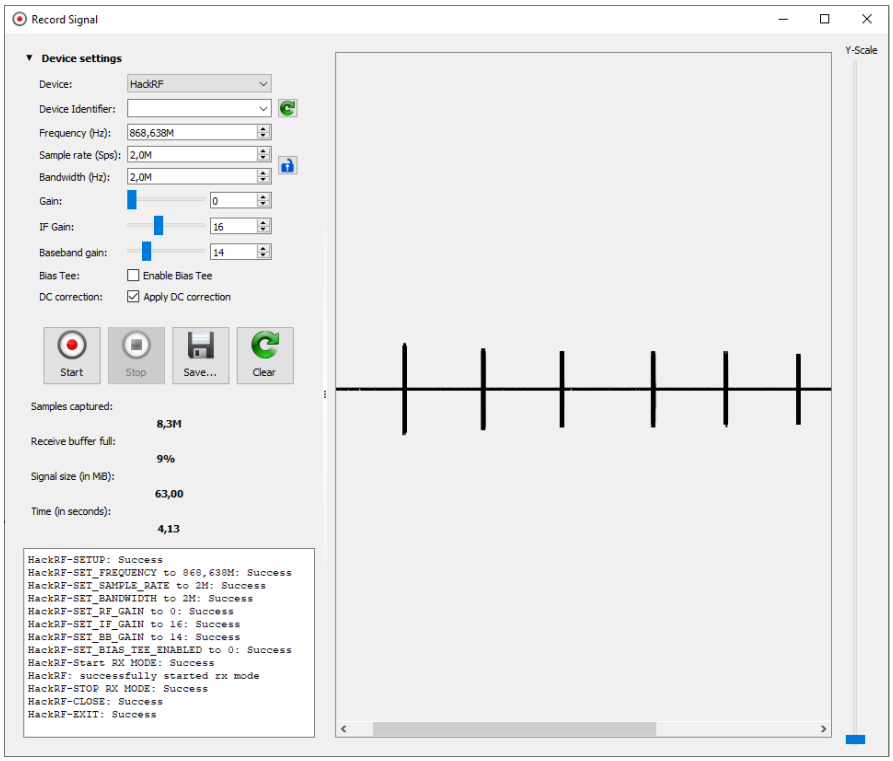


Figure 6.3 – Capturing an RF signal using Universal Radio Hacker.

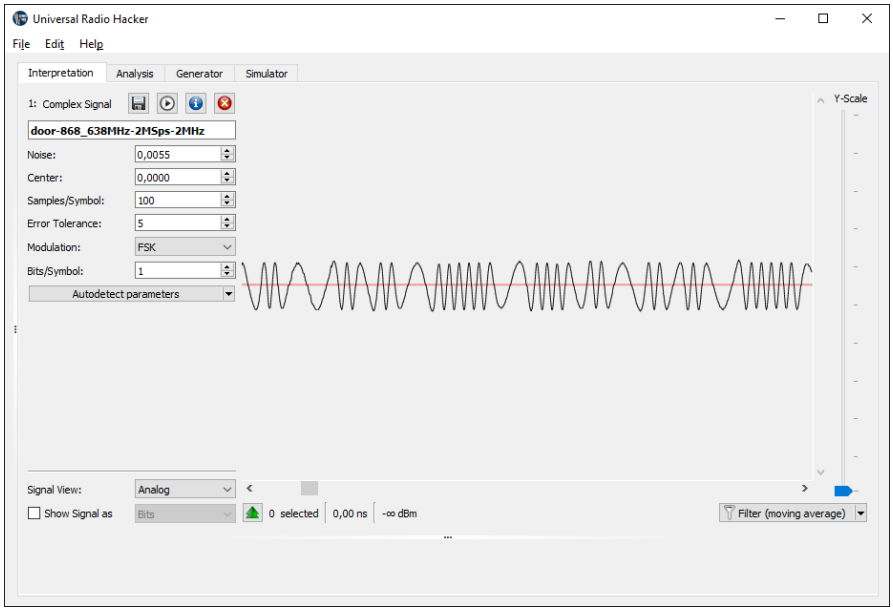


Figure 6.4 – A cleanly captured RF signal, showing a clear sinusoidal pattern.

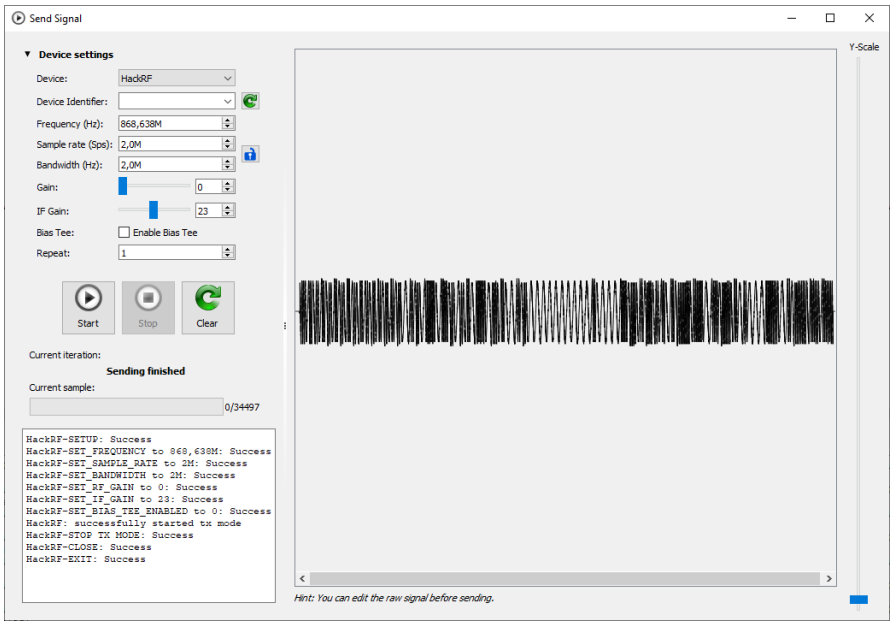


Figure 6.5 – Performing a replay attack using Universal Radio Hacker.

Results

This test was successful and revealed *critical* security flaws in the system. Every single tested endpoint of the RF communication was deemed vulnerable to replay attacks. The communication endpoints tested include the following:

- **Arming and disarming** the alarm from the remote keypad.
- Triggering/resolving the tamper sensor of the door sensor.
- Triggering/resolving the tamper sensor of the camera.
- Triggering/resolving the tamper sensor of the main panel.

This has critical consequences for the security of the system. Capturing the RF signal of the user arming and disarming the alarm gives an attacker complete control of the arming state of the system. By simply replaying the signal, the main panel perceives this as a valid signal coming from the remote keypad.

Discussion

The system has no protection against replay attacks whatsoever. This conclusion is strengthened by the fact that replaying the same signals several *months* later was still successful. However, replaying the signals against a different system of the same model was tested and it was unsuccessful. Presumably, some kind of ID of the device is sent making the packets invalid for another copy of the system. It clearly shows, however, that the manufacturer Climax Technology either lacks the knowledge and competence to implement secure RF communication or has decided not to prioritize security at the RF layer. Either way, this is a big mistake on their part which compromises the security of the whole system. SDRs have become much more available and much cheaper in the last few years. Attackers now have access to RF attacks which would have just a couple of years ago required very expensive and difficult to acquire hardware.

Furthermore, there are some factors that could potentially make this attack easier for an attacker in practice. When trying this attack signals were first incorrectly recorded at `868.0 MHz` instead of the correct frequency at `868.64 MHz`, giving rise to a very spiky signal. Still, the replay attack worked using this signal without issue. This indicates that perhaps the signal does not have to be captured that cleanly and that a badly captured signal, say from a distance or outside of the house, could be used to perform a replay attack. Presumably, the system implements some kind of error correction and noise filtering to increase the range and reliability of the RF communication. Additionally, regarding the tamper sensor signals, six signals are sent after one and the other, see figure 6.3. However, replaying only one of them still yields a successful replay attack. All of the six packets work individually. Presumably, the system repeats the same or similar signal several times for redundancy, in case one gets lost or corrupted in transit. This means, however, that an attacker only has to capture one of them to be able to perform a replay attack. The reliability of capturing and transmitting the signals from a distance is, however, a bit unclear. This was not tested. On the other hand, an attacker could easily purchase an RF amplifier to increase their range and sensitivity.

The consequences of this vulnerability are huge. An attack can completely bypass the alarm, defeating the whole purpose of the system in the first place. By replaying a captured disarm signal the attacker can disarm the system, granting them access to the property without triggering an alarm. Additionally, replaying any of the other signals, namely triggering the tamper sensor, an

attacker could trigger an alarm of an armed system without actually entering the premises. The owner would then get a message of an active breach and security personnel would be sent to the site. One could imagine an attacker repeatedly doing this until the owner perhaps thinks the system is broken and uninstalls it, at least temporarily, from the premises. At which point the premises would no longer be protected and an attacker could strike.

6.3 Task 2: Reverse engineering the RF protocol

This section covers the process of reverse-engineering the RF protocol, via captured RF signals. This included mainly trying to demodulate the signals into binary data and then analyzing the contents.

Background

The system in question uses RF signals to wirelessly communicate across the devices. These radio waves are used to transfer binary data, ones and zeroes, between each device. Transferring binary data over radio waves is done by a process called modulation [26]. Digital modulation, e.g modulating binary data, is a whole field of study in itself, and that this is only a brief overview covering the basics of modulation.

There are three primary simple ways of modulating a binary signal: Amplitude- (ASK), Frequency- (FSK), and Phase- (PSK) shift keying. They each produce a distinct waveform, which can easily be visually identified. This is shown in figure 6.6.

Each technique uses a different property of the sinusoidal wave to encode a zero and one respectively. Amplitude-shift keying (ASK) uses the amplitude of the wave, where a higher amplitude usually encodes a `1` and a lower amplitude a `0`. The frequency and phase of the wave is kept constant. FSK, on the other hand, keeps both amplitude and phase constant. Instead two different frequencies are shifted between to differentiate between a `0` or `1`. Lastly, Phase-shift keying (PSK) uses a phase-shift to differentiate between the two symbols. There are many much more complicated techniques to more efficiently encode a binary signal in radio waves, combining these techniques [26]. This can allow for much more information-dense modulation schemes, however, these are considered outside the scope of this thesis.

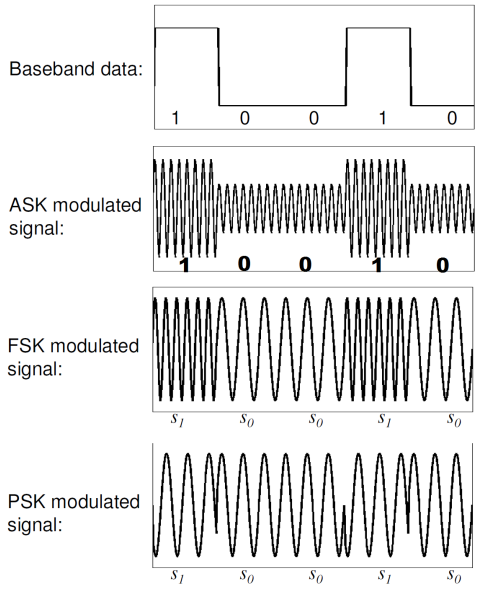


Figure 6.6 – The three primary techniques for digital modulation.

Conversely, *demodulation* is the process of converting a modulated signal back into the original binary data. In most real-world applications of RF communication this is done directly in hardware, using specialized radio receivers and circuitry to automatically convert the signal back into binary data [26]. Often this hardware also implements error correction, noise filtering, and other techniques to increase reliability of the communication, as interference and other disturbances are unavoidable in the real world. Figure 6.7 shows a very simplified circuit that implements binary FSK (BFSK) demodulation, e.g FSK modulation using two parameter frequencies.

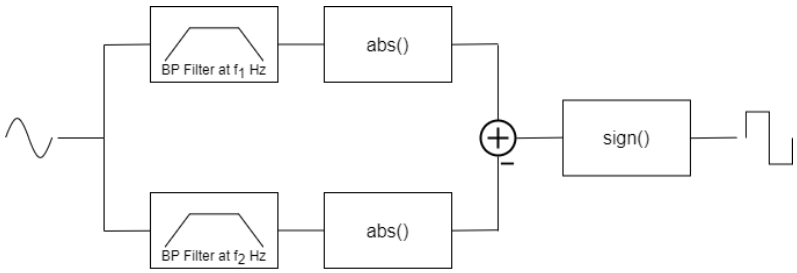


Figure 6.7 – A simplified BFSK demodulating circuit.

Given a BFSK signal, modulated using the frequencies f_1 and f_2, the circuit does the following. First, the signal is split and passed into two band-pass

filters, tuned to the two respective parameter frequencies. Then the absolute value is applied to the signals, one of them is inverted, and they are added together. Lastly, to convert the result into the final binary wave, the signal is passed to the $sign()$ function. This circuit was inspired by material from the course *EE123 Digital Signal Processing* at Berkeley EECS[1], specifically Lab 4 which covers FSK demodulation. Demodulation can of course also be done in software, but doing so in real-time is quite CPU-intensive and consequently a lot less energy efficient. Moreover, for each modulation type, there are several tuneable parameters that are hardcoded in the hardware demodulation chip, such as the frequencies of the FSK modulation, which during reverse engineering have to be figured out [18, 19], as covered in section 4.6 and 4.7.

Method

Initially, signals were captured according to the method described in the method part of section 6.2, using Universal Radio Hacker (URH). Carefully inspecting the signals in URH, as shown in figure 6.4, one can see that clearly FSK modulation is used to modulate the signal. This conclusion is corroborated by the official user manual submitted to the FCC [24], as well as official test documentation submitted to the *Ministry of Internal Affairs and Communications* (MIC) in Japan [27], in which the modulation type is specified to be FSK.

Furthermore, one of URH's key features is automatic demodulation [23]. This was tested on all captured signals. However, it was unfortunately continuously unsuccessful to automatically find the correct parameters of the modulation. Instead, the signals were demodulated *by hand* using the open-source audio processing tool *Audacity*[2]. The method to do this was inspired by the circuit shown in figure 6.7, as well as an excellent tutorial by hobby radio enthusiast Nick Oakman, showing how to demodulate an FSK signal using Audacity [28]. This was done in the following steps:

1. The raw signal data, as captured by URH, was imported into Audacity via the menu options *File - Import - Raw Data*. URH saves the signal as raw IQ data of signed 8-bit bytes. By selecting the encoding *Signed 8-bit PCM* with *2 Channels (Stereo)*, the I and Q components of the data

[1] https://sites.google.com/berkeley.edu/ee123-sp20/labs
[2] https://www.audacityteam.org

get separated and imported into two separate tracks. Using only one of them is enough to extract information in this case. As such the stereo track was split into two mono tracks in Audacity and the second one was then deleted. The rest of the import options do not matter.

2. Secondly, the center frequency, as perceived by Audacity, was noted. This can be found by the menu options *Analyze* - *Plot Spectrum*. Note that due to several features like the sample rate not being part of the raw signal data, this frequency might not equal the actual center frequency of `868.64 MHz`. This frequency was used in the steps below.

3. A *High-Pass Filter* was applied to the first track, and a *Low-Pass Filter* to the second, with the *Roll-off* value set to `48 dB`. The former essentially map a higher frequency to a higher amplitude in the output wave, and the latter does the opposite.

4. Using Audacity's scripting language Nyquist[1] the absolute value was applied to each track. The short Nyquist program `(s-abs *track*)` was used to do this. Then a low-pass filter was applied to both tracks, computing the envelope of the signal. Lastly, the track that originally had the low-pass filter applied to it was inverted. This means that the two tracks now correspond to where the original signal was of high frequency and low frequency, respectively.

5. Next, the two tracks were mixed together into a new track, e.g the signals were added together.

6. The mixed track was amplified as high as possible with clipping enabled, creating a binary signal. This created the final signal, the original binary signal, demodulated from the original signal.

7. Lastly, the final binary wave was exported as raw data to a file. This was done by selecting the final track and using the menu options *File* - *Export* - *Export Selected Audio*. In the subsequent file dialog "Other uncompressed files" was selected as the type, with no heading (e.g RAW), and signed 8-bit PCM encoding. This gives you the signal as raw binary amplitude data, represented as signed bytes.

Figure 6.8 shows the process of demodulating a signal in Audacity. Each track in the figure is the result of one of the steps in the above explanation. This process was applied to all captured signals. However, the result of this

[1] `https://www.audacityteam.org/about/nyquist`

process is a binary wave, not the actual binary data. To extract the binary data a simple python program was created, see figure 6.9. The program expects a list of x-axis offsets, where the first bit of the signal is. These values were found by hand, by manually looking at the plot that the program creates. The signal length and symbol length were also measured by hand, however, these are constant for all captured signals. Only the signal start offsets had to be manually derived for each signal, which was quite labor-intensive. Due to this, this process was only applied to a handful of signals, until enough information was gathered to make conclusions about the protocol.

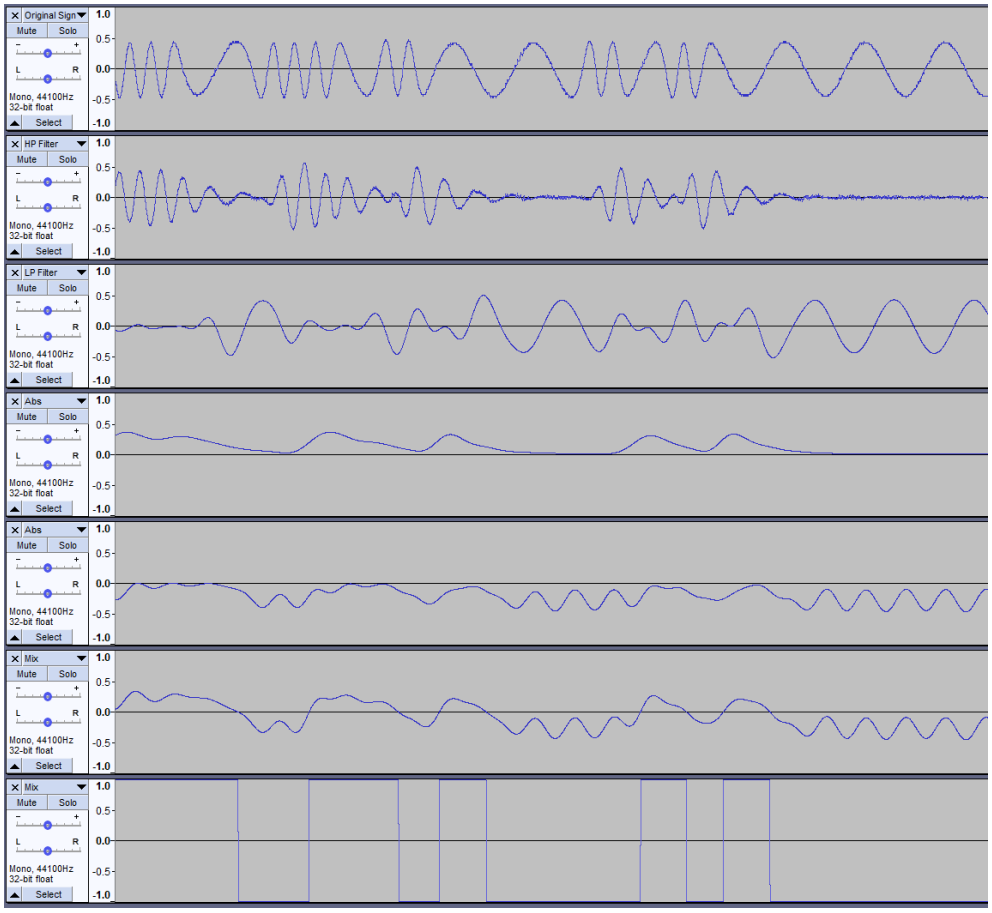


Figure 6.8 – The process of demodulating a BFSK signal in Audacity.

```
import matplotlib.pyplot as plt

SIGNAL_LEN, SYMBOL_LEN = 34000, 200
FILE, SIGNAL_OFFSETS = "door-signal.raw", [1800, 856160, ...]

with open(FILE, "rb") as f:
  signal = [b if b < 128 else b - 256 for b in f.read()]

for i, offset in enumerate(SIGNAL_OFFSETS):
  xs = range(offset, offset + SIGNAL_LEN, SYMBOL_LEN)
  plt.scatter(xs, [0 for _ in xs], c="red")
  bits = "".join(['1' if signal[x] > 0 else '0' for x in xs])
  print(f"Packet {str(i).ljust(2)} =", hex(int(bits, 2)))

plt.plot(signal)
plt.show()
```

Figure 6.9 – A program to extract bits from a binary wave and plot it.

Results

By using the Audacity and the method described, binary data was extracted from each recorded signal. This is presented in table 6.1, in hexadecimal form. Analyzing the data one can see a very clear structure emerging. All recorded signals decoded into exactly 170 bits and seemingly always have the following structure. See also figure 6.10.

1. A preamble of alternating ones and zeroes. This is a very common technique in RF protocol to notify of an incoming signal and to sync clock frequencies [18].
2. A sequence of bytes that is constant for each device. E.g the door sensor will always send the same byte sequence, the camera another one, etc. This corresponds to a syncword, which is often found in RF protocols, used to determine the protocol type or from which device the message originates [18]. Presumably, this is some kind of ID to tell the main panel from which peripheral this message is.
3. A sequence of zero bits. Presumably, this is part of the syncword. This is also a common technique in RF protocols, to let the receiver know what protocol this is and exactly where the payload data starts [18].
4. A sequence of seemingly random bits, presumably, the payload.

Door tamper sensor on
0xaaaaaaaa29cd29cd0a000015d477e072b922530064
0xaaaaaaaa29cd29cd0a0000028648b07e291d2ceecc
0xaaaaaaaa29cd29cd0a0000280b9d2e1d2d2ca7f31c
0xaaaaaaaa29cd29cd0a00002548c662f2feeea7fe22
0xaaaaaaaa29cd29cd0a000019201db301398d538674
0xaaaaaaaa29cd29cd0a00000806d6a5ee37481e2f76
Door tamper sensor off
0xaaaaaaaa29cd29cd0a0000102366a5cb78d61c0d0c
0xaaaaaaaa29cd29cd0a00000a3b2cb0867bf62aa616
0xaaaaaaaa29cd29cd0a000028fe2271f089a9e8c984
0xaaaaaaaa29cd29cd0a00001e23195bcbe8c65107ec
0xaaaaaaaa29cd29cd0a00001913b1ee7e3448da1cf0
0xaaaaaaaa29cd29cd0a000006f69dbb732deb2a120c
Camera tamper sensor on
0x155555554539a539a14000034164758f44cfae66f1
0x155555554539a539a14000034164758f44cfae66f1
0x155555554539a539a14000034164758f44cfae66f1
0x155555554539a539a14000034164758f44cfae66f1
0x155555554539a539a14000034164758f44cfae66f1
0x155555554539a539a14000034164758f44cfae66f1
Camera tamper sensor off
0x155555554539a539a140000342724d66fce053d3d7
0x155555554539a539a140000342724d66fce053d3d7
0x155555554539a539a140000342724d66fce053d3d7
0x155555554539a539a140000342724d66fce053d3d7
0x155555554539a539a140000342724d66fce053d3d7
0x155555554539a539a140000342724d66fce053d3d7

Table 6.1 – Binary data extracted from demodulating RF signals.

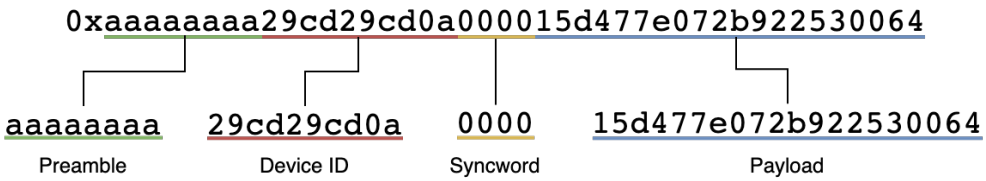


Figure 6.10 – The structure of a message in the proprietary RF protocol.

Given the final part of the message, which is seemingly random, and the fact that the replay attack presented in section 6.2 worked for any of the messages, we can conclude that some kind of encryption or obfuscation is used. For the tamper sensor on the door, the payload is seemingly random for every single message. For the camera sensor, however, the packets are always identical.

Discussion

The signals were successfully demodulated using the described method. Data extracted from this process shows a clear structure to the messages, giving further indication that the capturing and demodulation were done correctly. By visually inspecting the signals one could see that FSK modulation was used. Additionally, this is actually documented in the official user manual [24], meaning we can be sure that this conclusion was correct. Furthermore, the signals follow patterns and structures commonly used in RF protocols [18], such as starting with a preamble of alternating symbols and a syncword, which in this case seems to be some kind of ID to identify themselves in the protocol. This is discussed in section 4.6.

However, the payload is clearly encrypted or at least obfuscated in some way. This is in line with the documentation provided by Climax Technology, in which they reference a “Private Encryption Method” used in the RF communication [24]. What the actual method is and whether or not it is a cryptographically secure method is left unanswered. It could be the case that the encryption is quite weak. Without access to the firmware or additional documentation about the RF protocol, further reverse engineering is next to impossible given the current “black box” state of the RF protocol. This means that one, unfortunately, cannot create new messages from scratch.

In conclusion, this pentest did not yield any additional actionable information. While the demodulated data definitely gives some insight into the system, it

does not indicate any further vulnerabilities. Trying to reverse engineer the encryption method would yield a lot of additional information, and allow an attacker to create new messages from scratch. However, without access to the firmware, this is quite difficult and left as potential future work.

6.4 Task 3: RF Jamming Attack

This section covers a pentest of a jamming attack against the RF communication in the system. This is a DoS attack which if successful blocks or interferes with the RF communication between two devices, making them unable to send messages to each other [18].

Background

The radio frequency spectrum is a medium used for wireless communication. It is inherently vulnerable to jamming attacks. This can be likened to the RF equivalent of a DoS attack [18]. A jamming attack against RF communication involves directing electromagnetic energy in one or more radio frequencies against a system to disrupt or prevent signals from being transmitted between two systems [29]. In practice, this means sending out signals on a specific frequency, carrying enough energy to overpower anyone transmitting in the same frequency band. By continuously sending out signals, such that the wireless band is filled, legitimate traffic can be blocked.

Since RF communication uses a shared medium, this is an attack vector that can be incredibly hard to protect against. Often a system will communicate on a single, fixed frequency which can make the system particularly vulnerable to jamming attacks [30]. While many sophisticated techniques to detect jamming have been developed, detection and reporting are about the extent to which a system can react to a jamming attack [31]. Little else can usually be done, except to perhaps switch frequency band or fall back to an alternate mode of communication.

Method

To transmit signals the HackRF SDR was used. It was placed close to the system, within `10-20 cm`. See section 6.1 for a detailed description of the

lab setup. To generate a jamming signal the open-source program *GnuRadio Companion*[1] was used. This is a graphical tool used to control an SDR. It is based on creating flowgraphs of connected components to receive, process, modify, and transmit real-time radio signals from and to an SDR.

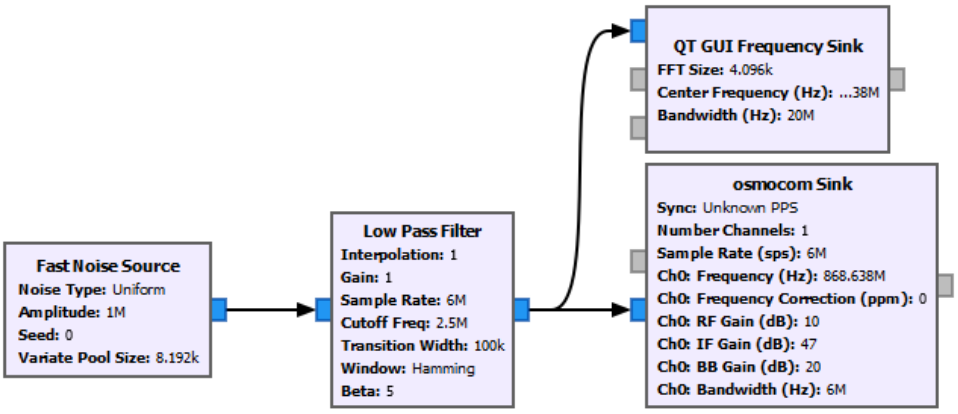


Figure 6.11 – A flowgraph in GnuRadio which performs a jamming attack.

To generate a noise signal, a flowgraph was created in GnuRadio Companion, see figure 6.11. Initially, a `Fast Noise Generator` was used as the source signal. The noise output was then linked to a low-pass filter, to concentrate the signals to the specific frequency band of interest. Lastly, the output was sent to the HackRF via the `osmocom Sink` block. Additionally, the output from the low-pass filter was also sent to a `QT GUI Frequency Sink` to visually present the sent signal data while performing the attack. This is shown in figure 6.12.

Results

This pentest was successful. Signals between the devices were jammed, leaving the system unable to communicate. However, after approximately 60 seconds of running the attack, the main panel triggered an *interference fault event*, meaning the jamming attack is successfully detected by the system. Crucially, however, the event is not logged in the mobile or web application and no notification is sent to the owner of the system whatsoever.

[1] `https://www.gnuradio.org/`

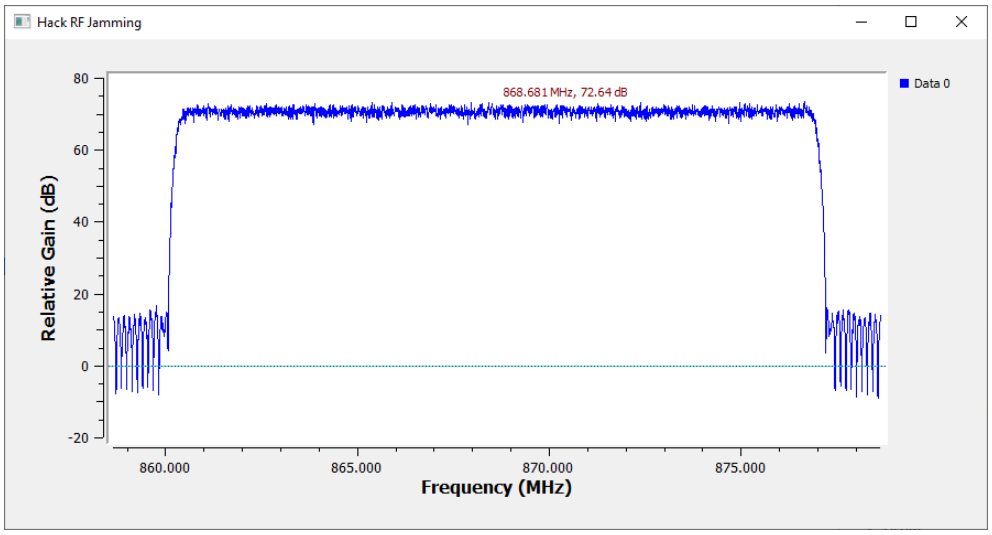


Figure 6.12 – A frequency graph from GnuRadio during the jamming attack.

Discussion

As expected, the system fails to communicate during the jamming attack. However, the system successfully detects this attack. Arguably, reasonable and appropriate measures have been taken by the manufacturer in this case. Given the inherent vulnerability of jamming attacks in a shared and open medium like radio frequencies, detecting and reporting is the only reasonable course of action [31]. This behavior in the system is documented in the official user manual submitted to the FCC [24] by Climax Technology. It describes in the *interference* status in section 6.1 of the manual that if a jamming period of 30 seconds or more during a one-minute window is detected then the event is triggered. This lines up with the behavior observed during the test. However, the system fails to log and report the attack to the user. This could potentially let an attacker jam the signal and do a malicious act without the user being notified. Additionally, the parameters of the jamming detection could be questioned. Letting an attacker potentially block `49%` of the communication channel might still disturb the system significantly and potentially block important communication.

In this test the most basic jamming technique was used, e.g to send out as much noise as possible at the correct frequency band of as high energy as possible. This makes the attack quite trivial to detect. There exist much more sophisticated jamming attacks, taking less of a brute force approach, which could potentially evade detection and yet still be effective [32]. These,

however, were considered outside the scope of this test and this report. Additionally, in this test, the jamming device (the SDR) was placed very close to the system. In a real-world application of this attack, the attacker would be at least a few meters away, blocked by a door. It is doubtful an SDR like the HackRF could generate enough noise to jam the system in those conditions. However, more powerful jammers, using dedicated hardware, could most likely successfully block the system's communication even in those conditions.

6.5 Task 4: Insecure Network Services

This section covers the network services found in the system and their implications on the system's security. As rated by OWASP, this is the second most common source of security issues within IoT systems [5], see section 4.1.

Background

During development, many manufacturers deploy network services on IoT devices for remote debugging. However, a very common source of vulnerabilities is leaving these network services active on live devices shipped to customers [5]. This could be done in negligence or on purpose to leave a backdoor for the manufacturer to debug live devices. These services could for example be a telnet or SSH server. This combined with weak passwords is an all too common occurrence in IoT systems [14]. This is such a common issue that OWASP ranks it the second most common source of security issues in IoT devices [5]. Additionally, the ETSI EN 303 645 standard explicitly mentions minimizing the system's attack surface, in provision 6, by removing unnecessary network services [6] (see section 4.2).

Often such services are completely unnecessary for the functionality of the system and only serve to increase its attack surface. Even when they are protected by a strong password it can sometimes be found by analyzing the firmware. Additionally, the system can be vulnerable due to vulnerabilities present in the network service itself, the protocol used, or the OS network stack. Additionally, these systems are often using outdated versions of said services, potentially leaving the system exposed to publicly disclosed

vulnerabilities. This is so common that OWASP ranks it as the fifth most common source of vulnerabilities for IoT systems [5].

Method

Initially, which network services the system hosts had to be identified. This was done using the open-source network scanning tool *Nmap*[1]. First, the basic port scanning functionality of Nmap was launched against the system, using the following command:

```
nmap -p- $IP
```

This scans the device for processes listening on all TCP ports in the entire 0-65535 range. Similarly, a scan was made on UDP ports as well. However, due to the known difficultly of UDP port scans, only the most common UDP ports were scanned, as a scan of the full port range was estimated to take several weeks. This was done using the following command:

```
sudo nmap -sU -F -v $IP
```

The UDP scan showed no active services except a DNS server on port 53. However, the TCP showed three network services listening on different ports on the device. The following network services were identified:

- 53/udp: A DNS server.
- 53/tcp: A DNS server.
- 80/tcp: The local admin server, as described in section 3.3.4.
- 58098/tcp: An unknown process listening in an unconventional port.

The web server listening on port 80 provides very little functionality, as described in section 3.3.4. It features a login form. An unsuccessful password attack was conducted against this form, see section 6.6. From the *Server* HTTP header, however, it leaks information about which web server is used.

To gain more information about these services a version detection scan was launched against the three ports, using the following command:

```
nmap -sV --version-all -p53,80,58098 $IP
```

[1] https://nmap.org/

The Nmap version detection scan took around 10 minutes and yielded no result for the unknown application running on port 58098 or any further information about the other two services. To further investigate information about the process the open-source application scanner *Amap*[1] was used. This is a lesser-known program created by the prolific hacker group *The Hacker's Choice*[2]. According to the authors, the Amap application scanner has largely been superseded by Nmap. Thus unsurprisingly, it provided no additional information about the services. However, the application suite includes a very simple TCP fuzzer called *Amapcrap*. This is a program that sends random bytes to an open TCP port to elicit a response from an otherwise silent application, which can be very useful in black box testing an opaque network service. This was tried against the system using the following command:

```
amapcrap $IP 58098
```

Results

The result of the full port scan shows three active services on the device:

- A DNS server, running on the standard DNS ports, *53/tcp* and *53/udp*.
- A web server, *80/tcp*, the functionality of which is covered in section 3.3.4.
- An unknown application, running on *58098/tcp*.

Additionally, the application and version scanners that were run against the services gave no information at all about the three services. The `Server` HTTP header returned from the webserver, however, leaked the fact that it uses an open-source web server for embedded systems called *Mongoose*[3].

Lastly, fuzzing the unknown service, *58098/tcp*, using the *AmapCrap* tool, made the service unresponsive. This is shown in figure 6.13. Presumably, the application crashed on this specific input.

Examining the input further reveals that the first two bytes represent the characters `[]` in ASCII, which is intriguing. Through manual testing using telnet on the correct port and trial and error, it was discovered that the application crashed on more inputs. These include `{}` and `[1,2,3]`, for

[1] https://github.com/BlackArch/amap
[2] https://www.thc.org/
[3] https://github.com/cesanta/mongoose

```
~/Desktop/amap on git:master x [9:31:58]
λ ./amapcrap 192.168.1.90 58098
# Starting AmapCrap on 192.168.1.90 port 58098
# Writing a "+" for every 10 connect attempts
# +

Error: Operation timed out

# Service seems to have crashed from the following trigger:
PROTOCOL_CRASH::tcp:1:0x5b5ddc776d4d28c70089eaa3af64cce2b82e5e087effa43b2d07f9f72
```

Figure 6.13 – The 58098/tcp application crashing during a fuzzing attack.

example. While the application presumably crashes and becomes temporarily unresponsive, the system seems to automatically revive the process. Within about a minute it starts accepting TCP connections again. None of the other functionality of the system is seemingly affected during this downtime, however.

Discussion

Performing a network scan on the system revealed three network services listening on three different TCP ports. However, no vulnerabilities were found as a result of these three services. The purpose of the first one, the DNS server, is quite unclear. There is seemingly no self-hosted domain documented anywhere so why the system hosts a DNS server is as of now unknown. The second server, the HTTP web server, has been covered at length in section 3.3.4, as well as a pentest that was launched against it which is covered in section 6.6. The third server is the most interesting find from this section. It is hosted on an unconventional TCP port, presumably to hide it slightly from attackers. The application is completely opaque and does not seem to react at all to most inputs. Through all fuzzing and manual testing, the application never sent a single response over the TCP connection. However, though fuzzing several inputs were found that seemingly crash the server. The pattern of said inputs suggests that the server might be trying to parse JSON and crashing when the input is of an unexpected format. It could perhaps be a successful DoS attack against the system. However, the purpose of this service is unclear and seemingly no functionality of the system is negatively affected while this application is unresponsive.

A guess as to the functionality of this server is an obfuscated backdoor to the system. Instead of a standard telnet server, it might be hidden behind

an application that expects a specific string. Only when it receives it does the remote shell launch. However, as it stands currently the application is a complete black box. Without access to the firmware additional investigation is very difficult. It was therefore considered unfeasible and it was not investigated further.

As stated, no additional vulnerabilities were discovered as a result of this investigation into the systems network services. The more important question, however, is *why*? When OWASP ranks it as the second most common source of security issues for this type of system [5], why are these network services active on the device in the first place? It is in clear violation of provision 6 of the ETSI EN 303 645 standard, urging manufacturers to minimize the attack surface. There is seemingly no reason for them to exist. If one were to discover the password to the admin web panel, for example, every aspect of the security of the system is compromised. It opens up the system to potentially critical vulnerabilities. Climax Technology have already made this mistake at least once before. Researchers investigating a similar system from the same manufacturer discovered a telnet server listening on a similar unconventional high port. By analyzing the firmware they found that the root password could be derived simply from the panels MAC address [15] (see section 4.3), giving them complete control over the system. This is a typical example of both the number one spot in the OWASP IoT top 10, and a violation of the first provision of the ETSI EN 303 645 standard.

Unfortunately, the firmware of the system in this thesis was not found publicly available. It was decided to be outside the scope of this thesis but if one were to, for example, solder off the flash memory from the main panel's PCB and read its contents one could get access to the firmware. It is not unlikely that through reverse-engineering the firmware one could find the password to the local admin panel or understand more about how the *58098/tcp* application works and what its purpose is. This could lead to additional *critical* vulnerabilities. The fact is that these services are seemingly not needed for the functionality of the system, and only serve to increase the attack surface of the system, is worrying and in clear contradiction of the ETSI EN 303 645 standard [6].

6.6 Task 5: Online Password Attack

This section covers an online password attack launched against the local webserver. The local web admin page (see section 3) has a login page that could potentially be brute-forced. Insecure or easy to guess passwords is a very common source of vulnerability in IoT systems. It is even ranked as the topmost common by OWASP [5] and the focus of the first provision of the ETSI EN 303 645 standard [6].

Background

A password attack, or password cracking, refers to cyber-attacks where the attacker tries to figure out valid credentials, to gain authorized access to a system. These can be categorized into two groups: offline and online password attacks. The former refers to attacks requiring no communication with the system in order to test a valid password [33]. An example could be listening in on network traffic and seeing a password hash. One could then perform an offline password attack by trying to figure out which password produced that hash and thus login to the system. In an online attack, the system under attack is in continuous communication with the attacker. This could be, for example, writing a script to try many different passwords on a login page of a web page. Online password attacks are generally harder to successfully perform. They are often much slower, as the communication with the system incurs a major overhead, and also poses the potential risk of getting caught in the middle of the attack if the administrator of the system notices the malicious traffic. Servers also often implement rate-limiting against IP addresses to combat these types of attacks and DOS attacks.

For both online and offline password attacks, there are several techniques one can use to try and guess the correct password. The simplest one is a *brute-force attack*, where the attacker simply tries all possible passwords up to some length [33]. Given c possible characters in the password and a password length of l, there are c^l possible passwords to try. This has exponential complexity in the length of the password, and will thus scale very poorly with longer passwords. Another technique is called a *dictionary attack*, where the attacker uses a large list of known common passwords [33]. Often these lists are created from actual passwords from leaked databases. For offline attacks like hash-cracking, there are additional techniques like *rainbow tables*.

In the case of the system in this thesis, we have no opportunity for an offline password attack, as no information about the password such as a hash is leaked as far as the author is aware. The local admin web page does, however, feature a login page. This page uses *HTTP Basic authentication* to log in to the main panel (see figure 3.8). If this login system has not implemented any form of rate-limiting then guessing the right password might be possible, given enough time and resources.

Method

We know from sources like the one covered in section 4.3 that *admin* is most likely a valid user name. This is further indicated by the official user manual from Climax Technology[1], which includes default login credentials with the admin user name (the credentials do not work on this system). As stated, the login form (see section 3.3.4) uses *HTTP Basic authentication* to authenticate the user. A dictionary attack was performed against this login page. The well-known password list *rockyou.txt*[2] was used as the dictionary. A useful program to perform online password attacks is called *Hydra*[3], which is a command-line tool. Using the following command, Hydra was used to perform the attack:

```
hydra -l admin -P rockyou.txt $IP \
    http-get "/action/login:F=Access Denied"
```

Results

The test was mostly unsuccessful. A password attack against the system was successfully executed. However, Hydra only manages to perform around 23 requests per minute against the main panel, see figure 6.14. This is too slow to be able to guess enough passwords to have a meaningful probability at a correct guess.

[1] https://fccid.io/GX9HSGWF1919/Users-Manual/Users-Manual-4873123

[2] https://github.com/danielmiessler/SecLists/blob/master/Passwords/Leaked-Databases/rockyou-20.txt

[3] https://github.com/vanhauser-thc/thc-hydra

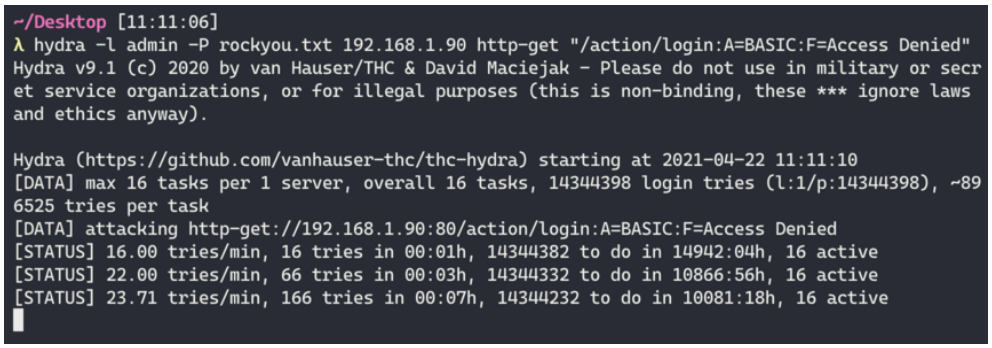


Figure 6.14 – The results of running a password attack.

Discussion

The Hydra tool was only able to do around 23 requests per minute. Hydra is known to be very fast, so that is not the bottleneck. One explanation for why this was so slow could be that the server rate limits users. This is a common technique to protect against password attacks, if enough failed login attempts occur from a single IP the server would temporarily block or throttle it. However, there are signs indicating that this is not the case. For example, accessing the main web page slows down tremendously while performing the attack. Initially, it had around a *16ms* response time, which increased to over *17 seconds* during the attack. This was even confirmed on another computer, indicating that the main panel has not implemented any form of rate-limiting per IP address. Presumably, the system is instead simply resource-bound and cannot serve requests at a faster rate. The CPU is most likely not that powerful, as is often the case in IoT systems. While the webserver slowed down significantly, the system seemingly had no issue detecting triggers from sensors during this time. This indicates that this is not a viable DOS attack, as the system as a whole was still functioning.

Due to the main panel not being able to serve more than 23 requests per minute, this type of attack is not feasible. For example, running a brute force attack, testing just all possible eight-character passwords, would take well over a hundred thousand years. A well-crafted dictionary attack could perhaps be effective. However, there is some indication that the password might be just a random string of characters, given that the default password given in the user manual is `cX+HsA*7F1` [24]. If that is the case then a brute force technique is the only possibility. The attack was therefore deemed unfeasible and not pursued further.

6.7 Task 6: Climax Technology's Vesta platform

Climax Technology, the manufacturer of the hardware used in this system, does not seem to be a consumer-facing business. Nonetheless, they have their own software platform to control their system, called *Vesta*. In the SecuritasHome system, this platform and its components are essentially replaced by *Alarm.com*. The Vesta platform features a mobile application[1] to control the system, as well as a web portal[2]. A potential security vulnerability is if this Vesta platform is still active and able to control this system.

Background

In the app *Vesta Home 5 EU*, see figure 6.15, one can perform essentially all actions that the Alarm.com mobile application provides (see section 3.3.3). On the landing page, you are greeted with simple a login page (where your Alarm.com credentials don't work). However, it also includes a button labeled *First Time Registration* (see figure 6.15a), where one can register a new account connected to a new system. To register a new system one only needs to enter its MAC address, see figure 6.15b, which is available without authorization from the local admin page (see 3.3.4). Potentially, one could then register the system in the Vesta platform to gain authorization to control the system, thus bypassing the security completely. In the Vesta web application, there is a very similar form, allowing you to register a new device using the MAC address, see figure 6.16.

Method

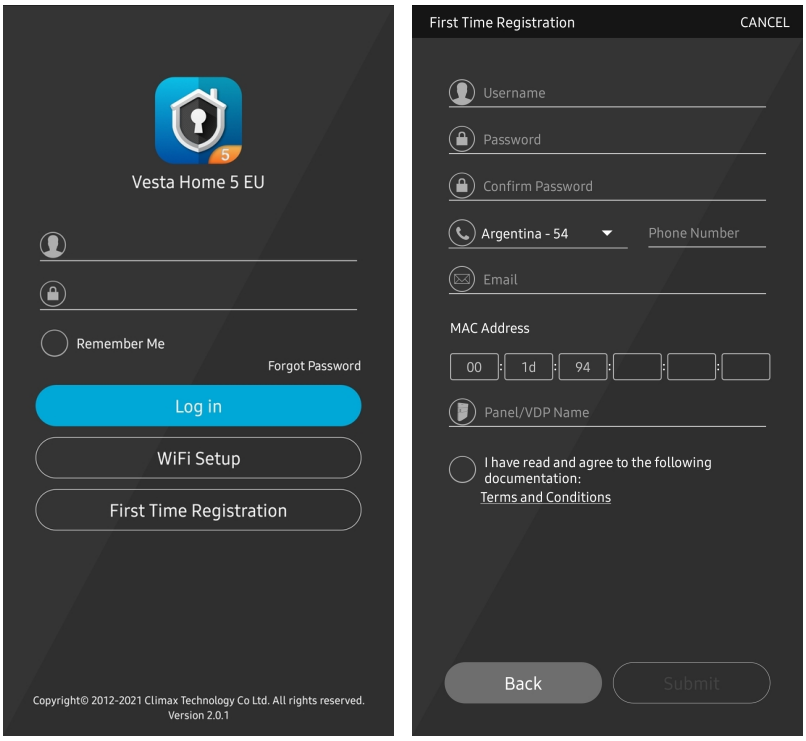
The mobile application is free to download. To be able to confidently monitor the network traffic, the mobile application was installed on an android emulator for PC, and Wireshark was run on the host machine.

Both applications use HTTPS, meaning the communication is encrypted. An attempt was made to perform a Man-in-the-middle (MITM) attack on the mobile application to view the HTTPS traffic, using *mitmproxy*[3] and the built-

[1] https://play.google.com/store/apps/details?id=com.climax.vestasmarthome.eu

[2] https://eu.vestasmarthome.com/Vesta/

[3] https://mitmproxy.org/



(a) The landing page

(b) The "First Time Registration" page

Figure 6.15 – The Vesta Home 5 EU mobile application.

in proxy settings of the android emulator. However, this made the application yield an error message saying it could not reach the server. Presumably, the application uses certification pinning to protect against this type of attack. This was not explored further since the traffic can easily be viewed in the web application, using the Chrome network tab (see figure 6.17), and presumably, both applications access the same backend API.

Results

The penetration test was unsuccessful. Both the mobile application and the web application yielded identical results. A simple error message is shown, saying the *MAC/IMEI* is incorrect, see figure 6.17. In the Chrome network tab, we can see when trying to register the system through the web application that the API responds with the message "no data found!".

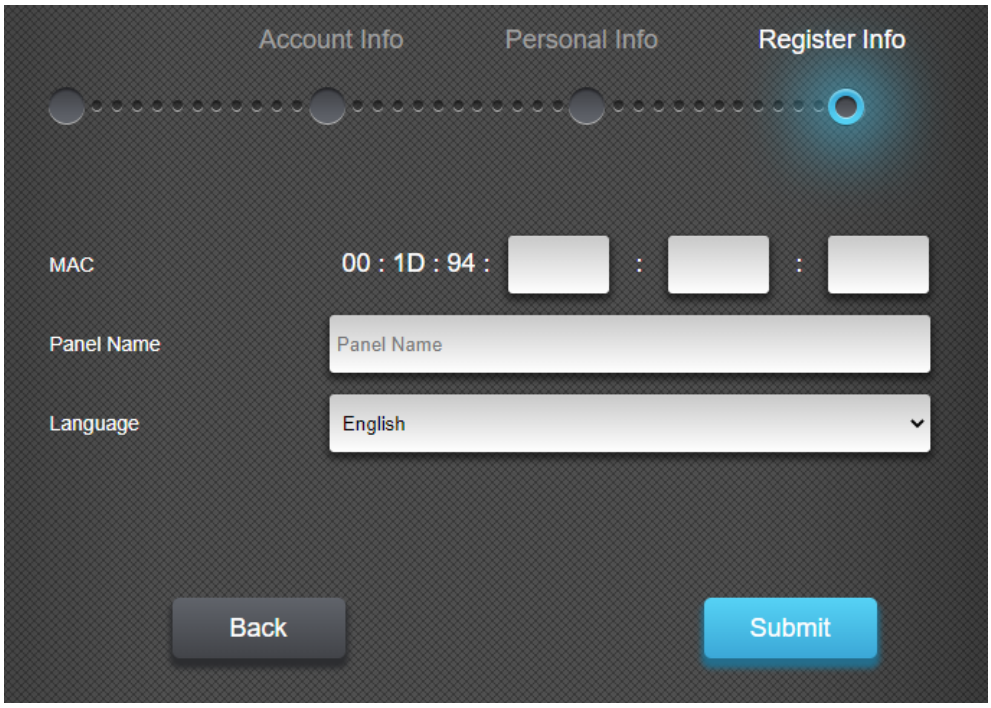


Figure 6.16 – The Vesta web application registration.

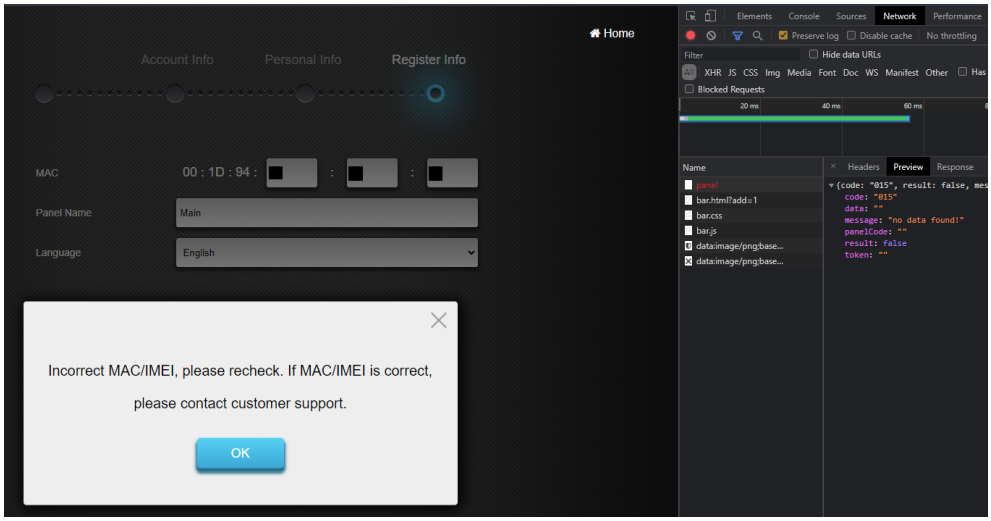


Figure 6.17 – The results of trying to register in the Vesta web app.

Discussion

Trying to register the hardware to the Vesta platform was unsuccessful. The given MAC address was not accepted. Presumably, Climax Technology has a

database of the MAC addresses of all sold systems under the Vesta platform. While the MAC address of the system in this thesis is registered under Climax Technology, it does not seem to be registered by the Vesta platform. Another possibility is that the MAC address and IMEI number pair are not registered because *Alarm.com* has used their own SIM cards and thus a different IMEI number. This is indicated by the error messages shown. Which of these scenarios is the correct one, we cannot know. Either way, we can see that the API responds with a negative result, saying that no data could be found. Therefore, this type of supply chain error seems to have been identified and correctly protected against.

6.8 Task 7: Insecure default credentials

This section covers the topic of insecure default credentials. It does not include a *pentest*, per se, but instead a list of all such credentials discovered in the system. These were discovered during the exploratory phase, and threat modeling phase of the project. Therefore, this section does not include a method part.

Background

Often default credentials are unavoidable. When first accessing the system, you need some way to do that without having previously authenticated yourself. These credentials can, however, be more or less secure. A huge problem within cybersecurity, and specifically IoT systems, is using insecure default credentials that can be easily guessed by a hacker [5]. Additionally, often the user is never forced or even encouraged to change these credentials, leading to potentially severe vulnerabilities. A common example is routers, which almost always feature a local admin page to configure settings. Usually, all routers of the same model have the same default login credentials and often these are as simple as `admin:admin`. This is such a common issue that there are public databases of these passwords for each model[1]. Often the owner of the router is not even aware that this page exists or that they should change the password. OWASP ranks this type of vulnerability as the number one most common and severe vulnerability in IoT systems [5].

[1] `www.routerpasswords.com`

Famously, the MIRAI botnet relied on insecure default credentials to hack into hundreds of thousands of devices. It used just `62` different credentials [14]. MIRAI is a worm malware, which targeted mostly IoT devices. It looked for devices with TCP port 23 or 2323 open, which are both often used by Telnet, and tried these credentials there. After gaining successful authentication against the device, these credentials and IP would be sent to a central server. Using this technique, over half a million devices were hacked and incorporated into the MIRAI botnet, which in turn was able to perform large scale DDOS attacks against websites. The combined bandwidth of the MIRAI bots were able to reach almost `1 TB/s` against a single target.

Results

The system can be armed and disarmed via the remote keypad (see section 3.2) by entering a personal four-digit pin. By default, this is set to `1234`. You are never forced or even encouraged to enter a new code, overriding the easily-guessable default.

Additionally, when an alarm is triggered Securitas will send security personnel to inspect the site. If you accidentally triggered the alarm yourself Securitas has a phone number you can call to cancel the alarm. However, as a security mechanism, when doing this you have to correctly say a four-digit code (different from the previously discussed code). If you do not remember your code or say it incorrectly personnel will be sent to the site regardless. This code is by default the last four digits of *your phone number*. Once again, you are never forced or even encouraged by the system to change this code.

No other bad default credentials were found. Importantly, the local admin panel (see section 3.3.4) does not seem to use bad default credentials. Several hundred common default passwords were tried and none of them were successful (see section 6.6). According to the user manual, the default password of the system, which has been changed by Alarm.com. is `cX+HsA*7F1` [24].

Discussion

Two insecure default credentials were discovered in the system. The first one, the arming pin defaulting to `1234`, is somewhat severe. After triggering an alarm, you have a set time interval to disable it using your personal four-digit pin before the alarm is sent to the alarm center. An attacker could bet on the

fact that this code has not been changed, and use it to turn off an alarm after breaking into the house. One could argue, however, that the probability of someone not changing this code is relatively low. The code arguably has an *obvious* insecurity, leading to people perhaps feeling a greater need to change it. However, using a random code as the default would be substantially more secure.

The second code, used to call off an active alarm, is also not ideal in terms of security. Your address and phone number can often easily be tied to one another and looked up using publicly available resources. An attacker obviously has the address if they are trying to get access to the house. Finding the phone number of the resident could be done in seconds using public websites for example. However, people can have multiple phone numbers and the house can of course have multiple residents. In fact, the ETSI standard recommends against using credentials tied to publicly available information [6].

Both identified insecure default credentials require the attacker to take quite a big risk to be able to exploit. An attacker would have to break into a house and only afterward bet on the fact that these had not been changed. One could then either turn off the alarm or try and call the alarm central before the owner does. All while the owner has been notified of an alarm breach via both email and the mobile app (see section 3.3.3). While perhaps not critical vulnerabilities, they are nevertheless completely unnecessary. Giving it an insecure default or tying it to publicly available information is a common source of vulnerabilities in IoT systems [5, 6]. The system could instead easily give you a random code instead or force you to enter one upon first registering the system.

Chapter 7

Reported Vulnerabilities

This chapter explains which vulnerabilities were reported to the manufacturer as well as a detailed timeline of events that took place during the responsible disclosure process and the two CVEs connected to this thesis.

Two vulnerabilities identified in chapter 6 were deemed sever enough to report to the manufacturer:

1. The RF replay-attack vulnerability identified in section 6.2, which allows an attacker to for example disarm an armed system.
2. The repudiation vulnerability arising from the absence of notifications to the owner of the system during an ongoing RF jamming-attack (see section 6.4).

7.1 Timeline of events

The following outlines the timeline of events during the responsible disclosure process.

- `[2021-06-15]` Securitas is contacted through their customer support and responsible disclosure form, and are notified that vulnerabilities might exist in the system.
- `[2021-07-07]` Terms are agreed upon by both parties and the two vulnerabilities are officially disclosed and described in detail to the manufacturer.

- [2021-08-23] Securitas acknowledges that both vulnerabilities are present in the system. Additionally, Securitas confirms that their hardware vendor has started developing a security patch. A bug bounty is offered and the two vulnerabilities are deemed by the manufacturer to have a CVSS score of 5.5 and 4.5 respectively.
- [2021-08-27] The two vulnerabilities are reported to MITRE and two CVEs are requested.
- [2021-08-29] MITRE responds and accepts both CVE requests. Two CVE ids are reserved:
  - **CVE-2021-40170** for the RF replay-attack vulnerability.
  - **CVE-2021-40171** for the RF jamming-attack vulnerability.
- [2021-09-21] A security patch is rolled out to all systems and Securitas' customers are officially notified via email of the vulnerabilities as well as the security patch.
- [2021-09-25] The responsible disclosure window ends and the thesis as well as the two CVEs are officially published.

Chapter 8

Discussion

This chapter contains a discussion about the methodology used in this report and a discussion of the results found in chapter 6. Lastly, a mandated discussion about the sustainability and ethics of this project is presented.

8.1 Methodology

The methodology used in this report is described extensively in chapter 2. Largely, the methodology applied in this thesis was effective. It gave a clear structure to follow and the threat model gave insights into what type of vulnerabilities to consider. If one were to blindly guess and pentest the system without any structure, one could miss some obvious threats. By following established attack libraries like the OWASP IoT Top 10 [5] (see section 4.1) and the ETSI EN 303 645 standard, it was easier to identify reasonable vulnerabilities to explore. Otherwise, one could quite easily get caught in dead ends with low chances of success.

A seven-step penetration testing methodology described by Weidman [3] was applied in this thesis. It was intuitive and easy to follow. However, one downside was that it perhaps was a bit excessive for this type of project. Being time-constrained was a constant theme during this thesis. Things had to be continuously delimited as the project went on and the deadlines approached. Ranking each threat using the DREAD model, for example, was suggested in the methodology but was decided to not be worth it for such a time-limited project. Since there were more promising threats to examine than there was

time, a more intuitive approach was used when deciding on what threats to focus on. Another downside was the inexperience of the author when it comes to threat modeling and pentesting methodology. It is not something that had been covered previously in the education at KTH. Therefore, a lot of the initial time of the thesis was spent on researching this area.

Overall, however, the methodology was very effective and gave good results. It gave structure to the penetration testing phase, which is otherwise often driven by intuition and probing the system looking for common sources of vulnerabilities. This is often the case during ethical hacking.

8.2 Results

The results from the penetration testing show that the system, in its current firmware version, unfortunately, cannot be considered secure. It is clear that some considerable thought has gone into the security of the system by the manufacturer. The system features tamper sensors to protect against physical attacks, it has a backup battery to guard against power outages, and it uses 3G telecommunication so as to not rely on the connectivity of the local network. Even on the RF level, some attention has been given to security. The system is able to detect RF jamming and implements some kind of encryption or obfuscation. However, in cybersecurity *one* mistake can be all it takes. When something as trivial as a replay attack is not protected against and has such critical consequences, it makes all of the other security measures irrelevant.

These results are made even more severe considering that the critical vulnerability is what is known as a “zero knowledge” attack. It requires no knowledge about the system and RF protocol from the attacker. Anyone with 350 USD left over to buy an SDR, and some very basic technical competence could perform the attack. Tools like *Universal Radio Hacker* [23] presents this process visually through an easy-to-use GUI, making it approachable to almost anyone. The reassurance, however, is that it requires capturing a live signal, e.g. someone actually coming home and arming the system. Intruders would then have to wait until they leave or come back later to exploit the vulnerability. This requires some dedication from the attackers, waiting potentially a long time until the right moment. It also requires physical proximity, however, that is already a given if one wants to enter a property.

Additionally, the results revealed several other promising avenues to explore in the system. For example, it hosts a suspicious network service, on *58098/tcp*. This is quite worrying, as insecure network services are one of the most common vulnerabilities against IoT systems [5, 6]. While this application could not be reverse-engineered in this project, one might be able to if the firmware of the system was acquired. There is potential that this could cause additional severe vulnerabilities. Additionally, one could potentially reverse engineer the RF protocol by analyzing the firmware and thus be able to construct and transmit arbitrary messages. More on this in section 9.2.

The results are reliable. Each penetration test was grounded in an extensive background section and they were all successfully repeated multiple times.

8.3 Sustainability and Ethics

In the field of computer security, ethical concerns are always present. The findings of penetration testing, or hacking, could have huge consequences and in some cases be incredibly damaging to the target. This makes ethical concerns relevant when performing penetration testing and security analysis. One has to take great care when hacking so as to not harm or disrupt unnecessarily. When a vulnerability is discovered another ethical concern emerges, namely how to ethically disclose it. There are three primary ways of handling a vulnerability from the point of view of a hacker [9].

- **Full disclosure**. The attacker makes the vulnerability fully public, without consulting the other party.
- **Non-disclosure**. The attacker does not disclose the vulnerability and potentially sells it or uses it for their own gain.
- **Coordinated Vulnerability Disclosure (CVD)**. A coordinated disclosure of the vulnerability, done together with the affected party.

The third one, CVD, is “expressly preferred” [9] and the only ethical choice. CVD aims to increase the security of IT systems in general by properly disclosing vulnerabilities. This increases the body of knowledge of the industry and lets the industry learn from other people’s mistakes. At the same time, it requires a dialog with the affected party and that both parties agree on how to deal with the vulnerability and how it will be reported. It is standard practice in cybersecurity to give the affected party at least 90 days to

develop and ship a security patch. If the other party is unresponsive, dismisses the vulnerability, or decides to not produce a patch within 90 days then you should publicize the vulnerability anyway. The cornerstone of CVD is that the knowledge about the vulnerability be made publicly available eventually [9].

Another ethical aspect of this project was making sure the law was followed. Penetration testing a system without the owner's permission is illegal in Sweden [12]. Careful thought had to be put into making sure the law was followed during this thesis. For example, during the initial system selection process products from the companies Verisure and SectorAlarm were considered. However, through contacting their support it was made clear that they would continue to own the physical system and require a technician to install the system on the premise. This would make it illegal to security test the system without being given explicit permission from those companies. Additionally, this law means that security testing the cloud servers that the system communicates with, or functionality of the website and the mobile app could be illegal. Therefore, this was a cause for delimitation of some aspects of the system.

Furthermore, sustainability is a concept that contains many different aspects, such as environmental and societal sustainability. The environmental connection to this project is negligible. A new system was procured, perhaps putting stress on the environment by manufacturing a system that was not intended to be used for a real practical purpose. However, that was a single purchase, and both the alarm system and the HackRF One SDR will be reused for future student thesis projects at KTH. Societal aspects, however, are more present. Our society is getting more and more digitized every year. Even large-scale infrastructure like the electric grid is now controlled by digital systems and is susceptible to cyber-attacks which can have *devastating* consequences. This is already a present threat to our society. In December of 2015, a cyber attack was launched on the Ukraine power grid, leaving over 200 thousand people without electricity for several hours [34]. This project, and all research within ethical hacking, arguably bring attention, knowledge, and resources to the field and work to increase our digital security, which is only getting more and more important by the year.

Chapter 9

Conclusions & Future Work

This final chapter contains the conclusions drawn from the result of this thesis and a conclusion about the system's overall security. Additionally, a discussion about future work that could be done on the security of the examined system is presented.

9.1 Conclusion

Is the SecuritasHome Home Alarm System secure against cyberattacks? The answer to the research question has to be **no**. It is clear that the manufacturer has put some considerable thought into security. They use tamper sensors on all devices to protect against physical attacks, it has a battery to protect against a power outage, they use 3G telecommunication so as to not rely on the local network connection, they are able to detect jamming attacks, and they use some kind of encryption in the RF protocol (the cryptographic security of which is still in question). However, due to a glaring security flaw in the RF protocol, not protecting against replay attacks, these measures are made largely irrelevant. It goes to show how one mistake is all it takes to completely negate the security of an IT system.

Additionally, there are some bad practices found in the system, in clear violation of the ETSI EN 303 645 standard for IoT manufactures [6]. One of them is leaving several network services on the system. They seemingly have no bearing on the functionality of the system and only serve to increase its attack surface, which is cause to worry.

9.2 Future work

There is a lot left to examine regarding the security of the SecuritasHome smart alarm system. This project focused on the RF protocol as well as the systems network services. The results, however, showed many other promising avenues to explore which were delimited mostly due to time constraints.

Firstly, somehow acquiring the firmware of the system would open the door for a lot of interesting research. It could allow you to analyze the behavior of the *58098/tcp* network service, for example, which is otherwise very difficult. One could probably the service by reverse-engineering the firmware. A possible technique to acquire the firmware would be to solder off the flash memory from the PCB and read its contents. However, that is a quite risky process that would certainly permanently break the system and possibly the flash memory in the process. The system supports over-the-air (OTA) firmware updates, however, no firmware update was sent during this entire project. Catching an OTA firmware update as it is happening is therefore quite difficult.

Additionally, one could possibly reverse engineer the encryption method used in the RF protocol by reverse-engineering the firmware. An interesting avenue is analyzing publicly available firmware from a similar system. *Lupus Electronics*, as discussed in section 4.3, publishes the firmware openly on their website[1]. Their system is also from Climax Technology and supports F1-compatible products (the proprietary RF protocol). It is however using a different panel model. Analyzing that firmware, it is quite probable that one could reverse engineer the encryption scheme used, as well as figure out the message structure used in the protocol. However, there was not enough time to include that analysis in this thesis.

The surrounding ecosystem, including the website and mobile application, is another interesting area to explore. This was delimited early on due to both the difficulty regarding the legal aspects, and due to the interests of the author. Analyzing the android app, for example, could reveal interesting exploits. This is often quite approachable since one can decompile android APKs to something quite close to the original java source code. Analyzing the API used by both the website and mobile app is another interesting area.

[1] https://www.lupus-electronics.de/en/service/downloads/

References

[1] N. M. Karie, N. M. Sahri, and P. Haskell-Dowland. "IoT Threat Detection Advances, Challenges and Future Directions". In: *2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT)*. 2020, pp. 22–29. DOI: 10.1109/ETSecIoT50046.2020.00009.

[2] The Business Research Company. *Smart Home Security Global Market Report 2021: COVID-19 Growth And Change To 2030*. 2021. URL: https://www.thebusinessresearchcompany.com/report/smart-home-security-global-market-report.

[3] Georgia Weidman. *Penetration testing: a hands-on introduction to hacking*. No Starch Press, 2014. URL: https://books.google.com/books?id=T_LlAwAAQBAJ&printsec=frontcover.

[4] Aaron Guzman and Aditya Gupta. *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices*. Packt Publishing Ltd, 2017. URL: https://books.google.com/books?id=rEFPDwAAQBAJ&printsec=frontcover.

[5] OWASP Foundation. *OWASP IoT Top 10 - 2018*. Tech. rep. OWASP Foundation, 2018. URL: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf.

[6] *CYBER; Cyber Security for Consumer Internet of Things: Baseline Requirements*. Standard. European Telecommunications Standards Institute (ETSI), 2020. URL: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf.

[7] Geir M Koien and Thomas Haslestad. "Security aspects of 3G-WLAN interworking". In: *IEEE Communications Magazine* 41.11 (2003), pp. 82–88. DOI: 10.1109/MCOM.2003.1244927.

[8] Robert David Steele. “Open source intelligence”. In: *Handbook of intelligence studies* 42.5 (2007), pp. 129–147. URL: https://books.google.com/books?id=U2yUAgAAQBAJ&printsec=frontcover.

[9] *Coordinated Vulnerability Disclosure: the Guideline*. Guideline. National Cyber Security Center, 2018. URL: https://english.ncsc.nl/publications/publications/2019/juni/01/coordinated-vulnerability-disclosure-the-guideline.

[10] Wenjun Xiong and Robert Lagerström. “Threat modeling – A systematic literature review”. In: *Computers & Security* 84 (2019), pp. 53–69. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.03.010. URL: www.sciencedirect.com/science/article/pii/S0167404818307478.

[11] Peter Torr. “Demystifying the threat modeling process”. In: *IEEE Security & Privacy* 3.5 (2005), pp. 66–70. DOI: 10.1109/MSP.2005.119.

[12] Justitiedepartementet. *Brottsbalk (1962:700) 4 kap. 9c §*. 2021. URL: https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/brottsbalk-1962700_sfs-1962-700.

[13] OWASP Foundation. *OWASP Top 10 - 2017*. Tech. rep. OWASP Foundation, 2017. URL: https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_(en).pdf.pdf.

[14] Manos Antonakakis et al. “Understanding the mirai botnet”. In: *26th {USENIX} security symposium ({USENIX} Security 17)*. 2017, pp. 1093–1110. URL: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf.

[15] Dan Fabian. *Examination of LUPUS-Electronics devices*. 2019. URL: https://wiki.elvis.science/index.php?title=Examination_of_LUPUS-Electronics_devices.

[16] Rob van Diermen. “The Internet of Things: a privacy label for IoT products in a consumer market”. MA thesis. Leiden University, 2018. URL: https://hdl.handle.net/1887/64571.

[17] Lars-Eric Hamid and Simon Möller. “How Secure is Verisure’s Alarm System?” MA thesis. KTH Royal Institute of Technology, 2020. URL: http://www.diva-portal.org/smash/record.jsf?pid=diva2%5C%3A1556774.

[18] Erez Metula. “Hacking The IoT (Internet of Things) - PenTesting RF Operated Devices”. AppSecIL. 2016. URL: https://owasp.org/www-pdf-archive/AppSecIL2016_HackingTheIoT-PenTestingRFDevices_ErezMetula.pdf.

[19] Harshit Agrawal and Himanshu Mehta. “RF Exploitation: IoT and OT Hacking with Software-Defined Radio”. RSA Conference. 2019. URL: https://youtu.be/88RfClJvPRQ.

[20] Flavio D. Garcia et al. “Lock It and Still Lose It —on the (In)Security of Automotive Remote Keyless Entry Systems”. In: *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016. URL: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/garcia.

[21] Kyle Greene et al. “Timestamp-based defense mechanism against replay attack in remote keyless entry systems”. In: *2020 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE. 2020, pp. 1–4. DOI: 10.1109/ICCE46568.2020.9043039.

[22] Samy Kamkar. “Drive it like you hacked it: New attacks and tools to wirelessly steal cars”. In: *Presentation at DEFCON* 23 (2015). URL: https://youtu.be/UNgvShN4USU.

[23] Johannes Pohl and Andreas Noack. “Universal Radio Hacker: A Suite for Analyzing and Attacking Stateful Wireless Protocols”. In: *12th USENIX Workshop on Offensive Technologies (WOOT 18)*. Baltimore, MD: USENIX Association, 2018. URL: https://www.usenix.org/conference/woot18/presentation/pohl.

[24] Climax Technology. *HSGW Series IP Alarm System User Manual*. Climax Technology. 2020. 91 pp. URL: https://fccid.io/GX9HSGWF1919/User-Manual/Users-Manual-4873148.pdf.

[25] *Short Range Devices (SRD) operating in the frequency range 25 MHz to 1 000 MHz; Part 2: Harmonised Standard for access to radio spectrum for non specific radio equipment*. Standard. European Telecommunications Standards Institute (ETSI), 2018. URL: https://www.etsi.org/deliver/etsi_en/300200_300299/30022002/03.02.01_60/en_30022002v030201p.pdf.

[26] Saleh Faruque. *Radio frequency modulation made easy*. Springer, 2017. URL: https://www.nvhrbiblio.nl/biblio/boek/Faruque%20-%20Radio%20Frequency%20Modulation%20made%20easy.pdf.

[27] Worldwide Testing Services (Taiwan) Co., Ltd. "Radio equipment according to Certification Ordinance Article 2 paragraph 1 item (8) for Smart Home Alarm Systems Model No.: HSGW-G8". In: (2018). URL: https://www.tele.soumu.go.jp/giteki/SearchServlet2?PageID=jt01&ATF=12866003.

[28] Nick Oakman. *Demodulating SAME FSK with audacity Part 1*. Youtube. 2018. URL: https://youtu.be/fOodFRviCys.

[29] David Adamy. *EW 102: a second course in electronic warfare*. Artech House, 2004. URL: https://books.google.com/books?id=-AkfVZskc64C&printsec=frontcover.

[30] Wenyuan Xu et al. "The feasibility of launching and detecting jamming attacks in wireless networks". In: *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*. 2005, pp. 46–57. DOI: 10.1145/1062689.1062697.

[31] Mingyan Li, Iordanis Koutsopoulos, and Radha Poovendran. "Optimal jamming attack strategies and network defense policies in wireless sensor networks". In: *IEEE Transactions on Mobile Computing* 9.8 (2010), pp. 1119–1133. DOI: 10.1109/TMC.2010.75.

[32] Aristides Mpitziopoulos et al. "A survey on jamming attacks and countermeasures in WSNs". In: *IEEE Communications Surveys & Tutorials* 11.4 (2009), pp. 42–56. DOI: 10.1109/SURV.2009.090404.

[33] Mudassar Raza et al. "A survey of password attacks and comparative analysis on methods for secure authentication". In: *World Applied Sciences Journal* 19.4 (2012), pp. 439–444. DOI: 10.5829/idosi.wasj.2012.19.04.1837.

[34] Defense Use Case. “Analysis of the cyber attack on the Ukrainian power grid”. In: *Electricity Information Sharing and Analysis Center (E-ISAC)* 388 (2016). URL: https://media.kasperskycontenthub.com/wp-content/uploads/sites/58/2016/12/21181126/E-ISAC_SANS_Ukraine_DUC_5.pdf.

For DIVA


```
{
"Author1": {
            "Last name": "Lindeberg",
            "First name": "Axel",
            "Local User Id": "alindeb",
            "E-mail": "alindeb@kth.se",
            "organisation": {"L1": "School of Electrical Engineering and Computer Science ",
                             }
          },
"Degree": {"Educational program": "Degree Programme in Computer Science and Engineering"},
"Title": {
            "Main title": "Hacking Into Someone's Home using Radio Waves",
            "Subtitle": "Ethical Hacking of Securitas' Alarm System",
            "Language": "eng" },
"Alternative title": {
            "Main title": "Hacka in i Någons Hem med hjälp av Radiovågor",
            "Subtitle": "Etiskt Hackning av Securitas Hemlarmsystem",
            "Language": "swe"
},
"Supervisor1": {
                "Last name": "Johnson",
                "First name": "Pontus",
                "Local User Id": "pontusj",
                "E-mail": "pontusj@kth.se",
                "organisation": {"L1": "School of Electrical Engineering and Computer Science ",
                                 "L2": "Division of Network and Systems Engineering" }
              },
"Examiner1": {
                "Last name": "Lagerström",
                "First name": "Robert",
                "Local User Id": "robertl",
                "E-mail": "robertl@kth.se",
                "organisation": {"L1": "School of Electrical Engineering and Computer Science ",
                                 "L2": "Division of Network and Systems Engineering" }
              },
"Other information": {
"Year": "2021", "Number of pages": "x,92"}
}
```

TRITA-EECS-EX-2021:351