DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Evaluating the security of a smart door lock system

KTH Thesis Report

Sebastian Veijalainen
Tommy Noreng Karlsson

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Authors

Sebastian Veijalainen <sebvei@kth.se>
Tommy Noreng Karlsson <tommynk@kth.se>

Information and Communication Technology
KTH Royal Institute of Technology

Place for Project

Stockholm, Sweden

Examiner

Robert Lagerström
Division of Network and Systems Engineering
KTH Royal Institute of Technology

Supervisor

Fredrik Heiding

Division of Network and Systems Engineering

KTH Royal Institute of Technology

Abstract

Smart home appliances and IoT devices in general are a fast growing market. Smart door locks are one type of device which is included in these categories. The smart door lock replaces a common deadbolt lock. In this study a smart door lock was penetration tested. The project started with a literature study, followed by a reconnaissance phase for gathering information about the smart door lock system. A threat model was created based on the findings during the reconnaissance phase and with the help of the STRIDE and DREAD frameworks. Penetration tests were chosen and conducted based on the threat model. The results of the penetration tests showed no major vulnerabilities, but a weakness in the communications between the mobile app and the server.

Keywords

Ethical hacking, penetration testing, smart door lock

Sammanfattning

Smarta apparater och generellt enheter på sakernas internet är en snabbt växande marknad. Smarta dörrlås är en slags enhet som inkluderas i dessa kategorier. I den här studien penetrationstestades ett smart dörrlås. Projektet började med en litteraturstudie, följt av en spaningsfas för att samla in information om det smarta låssystemet. En hotmodell skapades baserad på fynden från spaningsfasen och med hjälp av ramverken STRIDE och DREAD. Penetrationstester valdes och genomfördes baserat på hotmodellen. Resultaten från penetrationstesterna visade inga större sårbarheter, men en svaghet i kommunikationen mellan mobilappen och värddatorn.

Nyckelord

Etisk hackande, penetrationstestning, smart dörrlås

Acknowledgements

We would like to thank our examiner Robert Lagerström and our supervisor Fredrik Heiding.

Acronyms

IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
TLS	Transport Layer Security
SSL	Secure Sockets Layer
RFID	Radio Frequency Identification
NFC	Near-field Communication
UID	Unique Identifier
OWASP	Open Web Application Security Project
MITM	Man-in-the-middle
IETF	Internet Engineering Task Force
IP	Internet Protocol
ARP	Address Resolution Protocol
MAC	Media Access Control
CSRF	Cross-site Request Forgery
API	Application Programming Interface
TCP	Transmission Control Protocol
CA	Certificate Authority

DoS Denial of Service

Contents

Chapter 1

Introduction

The concept of smart homes, a part of the broader concept of the Internet of Things (IoT), can be described as homes in which there is a network of electrical appliances and services that the owner has access to and can control, both inside the home and remotely, and usually involves automation[8]. The idea of home automation has been around since the 1970's, but implementations were not very effective. Thanks to technological advances over the years, smart homes have been made possible and more functional and smart home projects are becoming more popular[19].

The market for smart home appliances is growing fast. In 2020 the market size was USD 36.5 billion, and it is forecast to be USD 92.72 billion in 2027 [16].

Door locks are one of the most fundamental of home security mechanisms and almost every home has at least one. Along with the development of smart home appliances and IoT devices, door locks have received their smart counterpart - smart door locks. The smart door lock provides residents with more options for unlocking the door. The idea is to improve convenience by removing the need for physical keys and enabling remote control of the door lock.

1.1 Problem

Moving sensitive data onto the internet and enabling remote control of IoT devices presents new security risks that were not present before [7, 21]. In the case of a smart door lock system, the consequences of a successful attack may be severe since an attacker could gain physical access to the home of the system's owner if control of

the system or the door lock is taken. Security should therefore be of high importance when developing IoT devices generally, and smart door locks especially.

1.2 Purpose and Goal

The goal of our work is to produce a threat model of a smart door lock in conjunction with a gateway and the accompanying mobile application and conduct a number of tests based on our threat model with the purpose of evaluating the system's security.

1.3 Delimitations

Due to time restrictions delimitations were made to the work. The hardware, firmware and radio communication presented in the threat model in section 4 was not tested.

1.4 Outline

In chapter 2, the system under consideration is described, as well as related work. In chapter 3 the methodology of the project, such as threat modeling is described. Chapter 4 contains the preparatory work and threat model for this project. In chapter 5 all of the penetration tests are presented with background, method, results and discussion. Chapter 6 presents the overall results of the project. Chapter 7 includes a conclusion and discussion about the results and the project.

Chapter 2

Background

This section describes theory about penetration testing, the smart lock tested in this study and related work on penetration testing smart locks.

2.1 Threat modeling

Threat modeling is a term which does not have a clear

2.2 System under consideration

definition and may be used in different ways[13, 18]. In this study, the threat model is a method of identifying and assessing potentially exploitable vulnerabilities in a system. Threat modeling can be done in several ways, and there are several frameworks for creating a threat model. Most threat modeling methods include gathering information about the system and using this information to create a model in which potential threats can be identified. The threat model is used as a guiding tool for the penetration testing, creating a better understanding of the system and what tests should be prioritized.The main components of the system are the following:

The door lock The door lock replaces a regular deadbolt door lock completely and includes doorhandles. When the door is closed, the door is locked automatically. The side of the lock on the exterior of the door has a touch pad for entering numbers and is used for pin codes and some configuration. The side on the interior of the door has a button which can be used to put the door in a state of being unlocked even when closed.

The door lock runs on AA batteries. When the batteries run out of power, the lock stays locked, but can be opened with physical keys.

There are several ways of unlocking the smart door lock.
• Pin code – Can be used to unlock the door. Up to 999 pin codes can be added. Adding a pin code requires first entering a master code. Up to three master codes can be added. At least one master code is added at the time of installation.
• RFID key tag – The smart door lock comes with 5 RFID key tags which can be added as valid keys by entering a command along with a master code.
• Mobile App – To be able to use the app a gateway is needed. To be able to unlock the door with the app, a Zigbee communication module is needed for the door lock.
• Physical keys – These are described as "emergency keys" and are only meant to be used in case the smart door lock cannot be opened using any of the other methods.

The gateway) The gateway enables the owner to control the smart door lock and any other devices that have been connected to the gateway via the mobile app. The gateway is connected to the internet over ethernet through a router and uses Zigbee to communicate with the door lock over radio frequency.

The Mobile app The mobile app offers the user a range of settings and controls for devices and the system. Below are the functionalities of the app most relevant to this report:

Add user/connect to gateway: When the app is opened, a guide for connecting the app and the gateway is shown. A button on the back of the gateway is pressed to start connecting the app and the gateway. Once the connection has been established, the user connecting to the gateway gets access to the app with administrator privileges.

Security: Here, the app includes options of changing privileges of users. There are four levels of privileges:
• Administrator: Has full access to make any possible changes in the app.
• Installer: Has the same access as an administrator, but without the possibility to change access of other users and configuring security settings of devices.

• Resident: Has the same access as the Installer, but cannot manage drivers or add or remove devices from the system.
• Guest: Only has access to controlling lights, climate and some other devices. Can start and view camera recordings. Guests cannot unlock the door lock.
The app also allows for using a 4-digit pin code to open the app.

Gateway settings: This section includes information on the current version of the gateway software, connectivity, protocol management and options to restart and factory reset the gateway. The protocol management option lets the user configure the username and password of a MQTT-broker in the gateway.

System Information: Includes information about the app version, the IP address of the gateway, a unique ID number and the uptime of the gateway. An event log is also included in this section of the app.

Other: There are several other functionalities meant to personalize the app for the user, such as themes, languages etc. Among these there is one function in particular which may be of interest, which is the notification option.

2.3 Penetration tests

2.3.1 Man-in-the-middle

A man-in-the-middle attack means that an attacker is able to eavesdrop and possibly alter communication between two parties without either of them noticing. There are several methods of doing this. ARP spoofing, in which an attacker exploits the ARP protocol, which handles translation between IP and MAC(physical) addresses, to associate its own MAC address with another host's IP address is one method[11, 14]. In this study, a pre-configured router was used to be able to run TCP-dump, a packet analyzer, in combination with Wireshark, to capture packets sent through the router. In the penetration tests, mitmproxy and Burp Suite were used. These programs act as a proxy server, on which HTTP, HTTPS and WebSocket (only Burp Suite) can be captured, analyzed and modified. HTTPS can be decrypted using a CA certificate provided by the developers of these programs.

Man-in-the-middle attacks can have severe consequences for the targeted parties. The communication between the targets could contain sensitive information which can be exploited by an attacker, such as authentication information or medical records. Sensitive information is usually encrypted, most commonly using HTTPS, meaning that a simple man-in-the-middle attack would not reveal the sensitive information. However, as mentioned earlier in this section, it is possible to break this encryption. If private certificates are trusted, meaning they are not validated by a CA, an attacker could intercept the traffic and forward it using its own forged certificate, enabling decryption of the HTTPS communication[1] [1].

2.3.2 Replay attack

A replay attack means that an attacker captures packets and sends them again to the receiver in order to execute an action[2]. Burp Suite has functions both for capturing and sending packages, which were both used in this study.

An example of a replay attack related to this study is as follows: A resident with unlock privileges uses the app to unlock the door. An attacker intercepts the unlock message being sent to the server and saves this message. At a later point in time, the attacker sends the captured message to the server. The server receives the message and sends an unlock command to the door lock, and the door lock is open.

The cipher text of messages encrypted with TLS will be different for different sessions, meaning that it prevents using the same encrypted message in a new session, since the decrypted message will be different. If the message can be decrypted and encrypted again with a correct session key, it may be possible to use the message. Timestamps may also be used to prevent replay attacks by only allowing a certain amount of time to have passed before the message is invalid.

[1] "How to defend yourself against MITM or Man-in-the-middle attack," in. [Online]. Available: https://web.archive.org/web/20131124235452/http://hackerspace.lifehacker.com/how-to-defend-yourself-against-mitm-or-man-in-the-middl-1461796382 (visited on 2021-06-22).

[2] "What Is a Replay Attack?" In. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/replay-attack (visited on 2021-06-17).

Chapter 3

Related Work

A literature study of other penetration tests on smart door lock systems was conducted to get a better understanding of the subject area. Google Scholar and the KTH Library were searched with the keywords ”penetration testing”, ”door lock”, ”smart lock”, ”smart door lock” and ”smart door lock penetration test”. A list of previous theses was also used[1].

3.1 ”Security evaluation of smart door locks” by Arvid Viderberg

One previous degree project at KTH was of interest due to several discovered deficiencies, namely ”Security evaluation of smart door locks” done by Arvid Viderberg in 2019[17]. The aim of this study was to investigate whether known vulnerabilities in smart locks existed in smart locks at the time. A descriptive list of known vulnerabilities in smart locks was presented and threat model for smart locks was created. The Net Smart Lock and the Ali Lock were tested for vulnerabilities. The penetration tests done were state consistency attacks, brute force attacks as well as man-in-the-middle attacks. The study found that state consistency attacks and brute forcing the password reset mechanism was possible in the locks tested, as well as vulnerabilities in the password policy. It was concluded that some known vulnerabilities in smart locks were present in smart locks at the time.

[1]“Ethical Hacking,” in. [Online]. Available: `https://www.kth.se/cs/nse/research/software-systems-architecture-and-security/projects/ethical-hacking-1.914053` (visited on 2021-06-15).

3.2 ”Security evaluation of a smart lock system” by Raihana Hassani

In the KTH degree project ”Security evaluation of a smart lock system” from 2021 by Raihana Hassani, the Verisure smart lock system was examined and tested for vulnerabilities [5]. The study followed the guidelines of the ”IoT penetration testing cookbook: identify vulnerabilities and secure your smart devices.” by Aaron Guzman and Aditya Gupta [4] for creating a threat model. The study included man-in-the-middle tests both by capturing traffic from a router and using a *mitmproxy* proxy server, DoS against a user account, information disclosure in the password reset function, skimming and cloning a key tag and reverse engineering the APK-files of the android application. The results of the penetration tests did not show any major vulnerabilities. However, a few weaknesses were discovered. A DoS attack, inconsistent password policy, sensitive user information exposure as well as cloning a key tag were included as weaknesses. Countermeasures to combat these weaknesses were presented, such as implementing two-factor authentication. The study concluded that the system is relatively secure with a few weaknesses that could be damaging, but that such an outcome is unlikely.

3.3 ”Finding Vulnerabilities in IoT Devices: Ethical Hacking of Electronic Locks” by Joachim Toft and Christopher Robberts

Another degree project at KTH was ”Finding Vulnerabilities in IoT Devices: Ethical Hacking of Electronic Locks”. Like Hassani, Toft and Robberts also based their threat modeling methodology on Guzman and Guptas book. They used STRIDE to categorize threats and DREAD to rate them. Toft and Robberts performed a MITM attack to intercept, replay and fuzz communication from their chosen target. They also looked for state consistency attacks and performed a reverse engineering analysis of the accompanying mobile application.

The MITM attack allowed the authors to see the communication which was encrypted but revealed a ”general message structure” that they ultimately felt they spent too much time investigating. Replaying and fuzzing messages proved unsuccessful. A state

consistency attack was found that enables users to avoid access revocation while their internet access is turned off. The reverse engineering of the application did not expose any security vulnerabilities but proved useful in better understanding the system being investigated [12].

3.4 ”Security analysis of Internet-of-Things: A case study of august smart lock” by Mengmei Ye et al.

In the study ”Security analysis of Internet-of-Things: A case study of august smart lock”, the security of the August smart door lock was analyzed[20]. The report presents an overview on the system, illustrates a threat model, as well as attack and defense strategies for the August smart lock system. Four different attacks were done against the system: A Handshake Key Leakage Attack, an Owner Account Leakage Attack, a Personal Information Leakage Attack and a Denial-of-service (DoS) Attack. All four attacks were carried out successfully.

The Handshake Key Leakage Attack and Owner Account Leakage attack were both considered extremely severe since both of these attacks give the hacker full control of the smart lock. The Personal Information Leakage Attack could have serious consequences due to sensitive information being obtained by the attacker, although it did not affect the lock itself. All of the attacks in this study involved investigating the file system of a rooted Android device or a jailbroken iPhone, where information such as handshake keys and user account information was found.

Countermeasures for all of the demonstrated attacks were suggested and security measures for apps, mobile operating platforms and smart home appliance hardware in general were also discussed.

3.5 Smart Locks: Lessons for Securing Commodity Internet of Things Devices by Grant Ho et al.

In this study, the security of smart locks was examined[6]. Five different smart locks were chosen; the August, Danalock, Kevo, Okidokeys, and Lockitron. Threat models

for these smart locks were presented and the locks were tested for state consistency attacks in the form of revocation evasion and access log evasion. Unwanted unlocking was tested in the form of unintentional unlocking (if the phone is within a certain radius of the door lock). Relay attacks were discussed but not tested for in this study.

The results of the study showed that state consistency attacks were possible in all of the locks except the Lockitron. The three locks with auto-unlocking features were all vulnerable to unwanted unlocking. Several defenses for the possible attacks found in this study were presented.

Chapter 4

Method

This chapter presents the methods of the reconnaissance and threat modeling.

4.1 Reconnaissance

Before and during threat modeling, reconnaissance work was done in order to gather useful information about the system. The company's websites and user manuals as well as a number of practical tests of for example the functionality of the app were used to gather information.

The following tests were done:

1. Testing features of the mobile app.
2. Examining hardware.
3. Port scan on the gateway.
4. TCP dump using Wireshark.

Ports were scanned using SYN, FIN, NULL and Xmas scans. All scan types scan for TCP ports. A scan for UDP ports was not performed due to time constraints.

4.2 Threat modeling

In this study it was decided to follow the guidelines of Guzman and Gupta for threat modeling [4]. The modeling process was comprised of the following steps:

1. Identifying the assets of the system.

2. Creating an overview of the architecture of the system.

3. Decomposing and analyzing dataflows and protocols.

4. Identifying, documenting and rating threats.

These steps were followed and accompanied with reconnaissance work as described in 4.1. Primarily step 3 of the list required some additional work to gather the information needed.

4.2.1 Identifying assets of the system

To identify the assets of the system, the product description and manuals were read. The hardware was inspected from the outside. A TCP dump on the router connected to the gateway was viewed in Wireshark to identify any connections made to remote servers.

4.2.2 Creating an overview of the architecture of the system

The functionalities and features of the app were tested and documented. Microsoft's Threat Modeling Tool[1] was used to create a model of the system architecture.

4.2.3 Decomposing and analyzing dataflows and protocols

Protocols were identified using TCP dump and Wireshark and the dataflows were added to the model.

[1]"Threat Modeling," in. [Online]. Available: `https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling` (visited on 2021-06-14).

4.2.4 Identifying, documenting and rating threats

Identifying threats was done using the STRIDE framework [15].

- Spoofing - Illegitimate user identifying themselves as a legitimate user.
- Tampering - Changing data in any way without permission.
- Repudiation - Evading detection while performing illegal activity.
- Information Disclosure - Data that could be considered sensitive being exposed.
- Repudiation - Evading detection while performing illegal activity.
- Elevation of Privilege - User gaining higher privileges than they are allowed.

Assessing the threats was done with the DREAD framework (taken from[2]).

- Damage Potential - If the vulnerability is exploited, how much damage will be caused?
- Reproducibility - How reliably can the vulnerability be exploited?
- Exploitability - How difficult is the vulnerability to exploit?
- Affected Users - How many users will be affected?
- Discoverability - How easy is it to discover the threat?

[2]"Security/OSSA-Metrics," in. [Online]. Available: `https://wiki.openstack.org/wiki/Security/OSSA-Metrics#DREAD` (visited on 2021-06-14).

Chapter 5

Threat Model

This chapter first presents the results of the reconnaissance tests, and then the threat model of the system.

5.1 Results of reconnaissance

This section presents the results of the reconnaissance tests.

5.1.1 Testing the features of the mobile app

During testing of the mobile app's features, it was noticed that every new user is given admin privileges as a default. Push notifications when a user is added are turned off by default. A 4-digit pin code may be chosen for app access.

It was also found that the MQTT-broker of the gateway had a preset default username and a preset random 7-digit password. Given that the attacker knows this, a brute force attack taking little time may be possible if username and password are not changed and no countermeasures are implemented[1]. It is also noteworthy that passwords as short as one character are allowed. However, it was discovered that the MQTT-broker is implemented for users to integrate their own IoT devices into the system. It is not related to the door lock and was therefore not included in the threat model.

The event log is often updated with events about the running state of the gateway as well as a battery level and voltage. It is unclear which battery the update is about, since

[1] "Brute Force Calculator," in. [Online]. Available: `https://tmedweb.tulane.edu/content_open/bfcalc.php` (visited on 2021-06-15).

the gateway does not run on batteries, but the updates are made regardless of devices being connected to the system or not. The event log does include an event when the door lock is unlocked, where the user's name is included. There does not appear to be a way of erasing any events in the event log from a user perspective. If a user unlocks the door and the user is then deleted, the name of the user is replaced with ”Deleted user”.

The mobile app only uses internet to communicate and is not usable without an internet connection. This means that a state consistency attack like the ones found by Arvid Viderberg[17] and Ho et al.[6] in previous studies is not possible.

To be able to use the mobile app, it must be paired with the gateway by pushing a button on the gateway while connected to the same network. Authentication is done using keys instead of username and password, meaning that password or password reset mechanism attacks such as the one in Arvid Viderberg's or Ho et al.'s studies [6, 17] mentioned in section 3 are not possible.

5.1.2 Inspecting hardware

The only open hardware port found was an ethernet port. Decomposing the devices was avoided due to risk of breaking the devices.

5.1.3 Port scan

All port scans yielded the same results, shown below:

```
Not shown: 65531 closed ports
PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 7.6 (protocol 2.0)
5355/tcp open  llmnr
8080/tcp open  ssl/http Tornado httpd 4.5.1
8883/tcp open  ssl/mqtt
```

5.1.4 TCP dump/Wireshark

A TCP dump[2] was run on the router to which the gateway is connected.

[2]“TCP dump and libpcap,” in. [Online]. Available: https://www.tcpdump.org/ (visited on 2021-06-21).

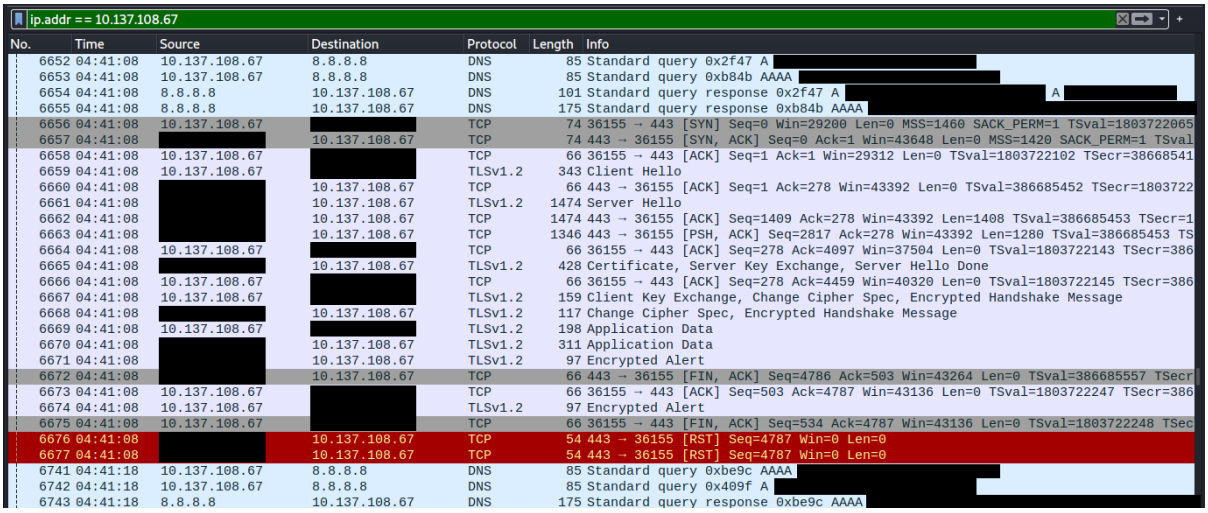

Figure 5.1.1: IP packets viewed in Wireshark

It was discovered that there are two different types of connections between the mobile app and the gateway. If the two are on the same network, there is a local connection directly to the gateway. If they are on different networks, a cloud server is used as an intermediary.

It was also noted that the gateway makes frequent DNS requests for the cloud servers' IP addresses.

Figure 5.1.1 shows an excerpt of the communication captured with TCPDump. The excerpt shows some of the DNS queries frequently being made by that gateway (packets № 6652–6655 and 6741–6743), a TLS handshake that initiates encrypted communication with a server (packets № 6659–6668) and some of the encrypted data being exchanged between gateway and server (packets № 6669–6670).

5.2 Threat model

5.2.1 Identifying assets and technologies

Assets of the system and the technologies used were identified and documented together in table 5.2.1.

Table 5.2.1: Identified assets of the system.

Asset	Description
The door lock	The smart lock replacing the common door lock. Can be used seperately with pin code, key tags (RFID) and pysical keys, or in combination with a gateway such as the Boreas X10, which enables use with the mobile app.
RF module	Enables communication between the smart lock and the gateway through radio coummunication using the Zigbee protocol.
Gateway	Enables communication between the smart lock and the app. The gateway communicates with the smart lock using the Zigbee protocol and with the app over internet through an ethernet connection via a router.
Mobile app	The mobile app is available for iOS and Android and communicates with the server or the gateway via HTTPS through any router using WI-FI or GSM.
RFID key tags	Key tags that can be used to unlock the smart lock. Uses RFID.
Physical keys	Common physical keys that can be used to unlock the smart lock.
Communication	Mobile app uses WI-FI and gateway uses Zigbee. Other communication is over ethernet. The RF communication over Zigbee is out of scope for this report.
Firmware	The firmware is out of scope for this report.

5.2.2 Architecture overview and data flow graph

An overview of the system's architecture and a data flow graph of the system was drawn using Microsoft's Threat Modeling Tool[3], shown in figure 5.2.1. Each box or circle represents an asset or part of the system. The arrows represent data exchanges.

5.2.3 Documenting Threats

Threats were identified and categorized using the STRIDE table 5.2.2, a list generated by the Microsoft Threat Modeling Tool as well as OWASP's Internet of Things (IoT) Top 10 2018[4]. Due to the hardware, firmware and radio communication were out of scope for this project, threats pertaining to these categories were not identified.

[3]"Threat Modeling," in. [Online]. Available: `https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling` (visited on 2021-06-14).

[4]"Internet of Things (IoT) Top 10 2018," in. [Online]. Available: `https://sectigostore.com/blog/owasp-iot-top-10-iot-vulnerabilities/` (visited on 2021-06-16).

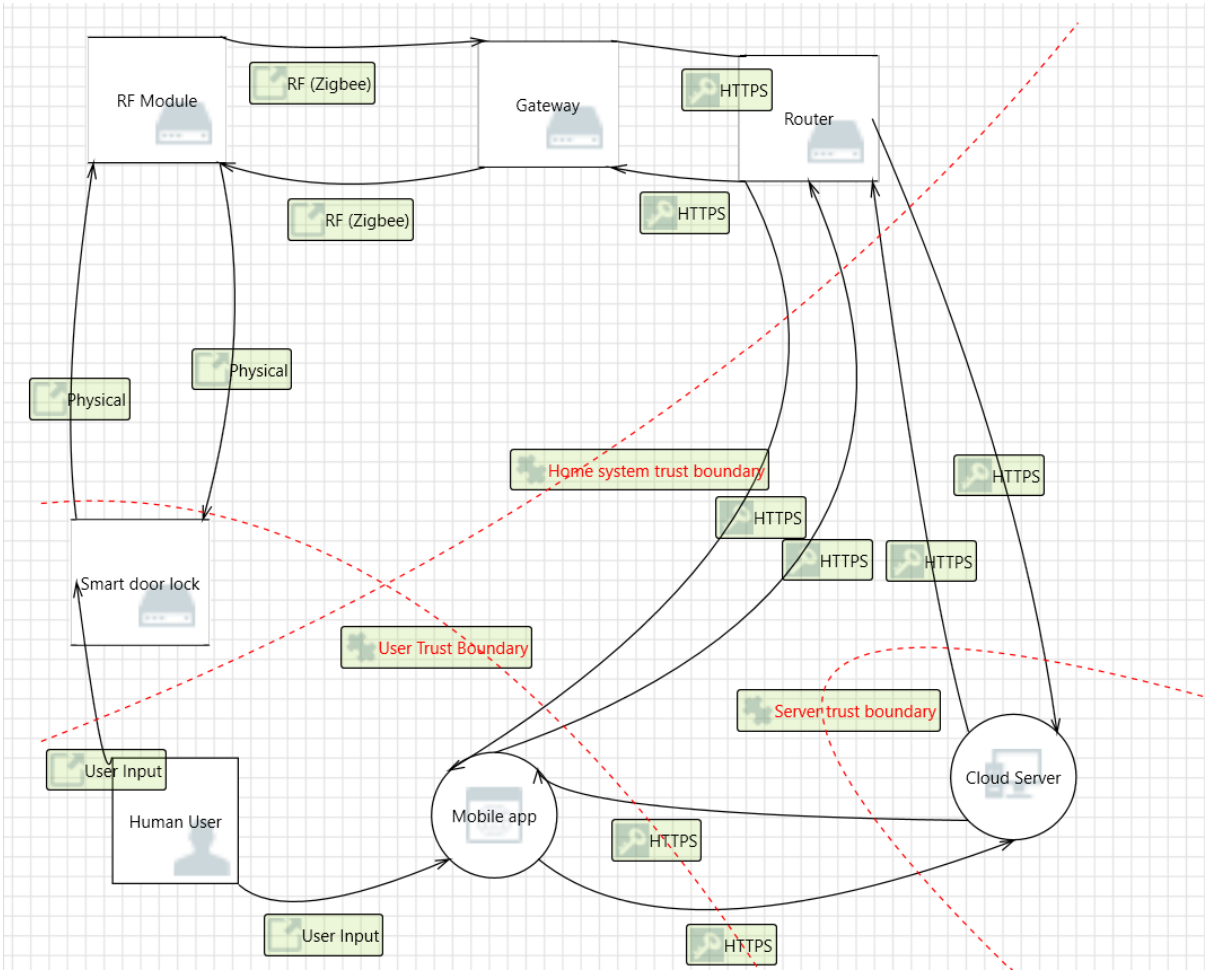

Figure 5.2.1: Architecture overview and data flow graph made with Microsoft's Threat Modeling Tool.

Table 5.2.2: Identified threats using STRIDE.

Threat Category	Description
Spoofing	Imitating the identity of the user or any of the assets in the system during communication. Imitating a legitimate key tag by cloning or emulating it.
Tampering	Capturing and/or modifying any communication. Making changes to the mobile app or related files. Making changes to a key tag.
Repudiation	Erasing or evading the mobile app's event log. Concealing an illegal action in a legitimate user's identity.
Information Disclosure	Extracting sensitive information from communications. Extracting sensitive information from the mobile app, its code or related files.
Denial of Service	Flooding the server, router, gateway, app or lock. Jamming the wireless communication between the lock and the gateway or the app and the gateway.
Elevation of Privelege	Gain user-level access to the system and through some method and elevate priviliges through some method.

Table 5.2.3: Threat 1

Threat description	Attacker gains access to the authorization information of a legitimate user or component by eavesdropping communications. The information can be used to spoof their identity.
Threat target	Communications.
Attack techniques	Intercepting communications between the server and the app, the server and the gateway or the app and the gateway.
Countermeasures	Use encryption and authentication for communications.

Table 5.2.4: Threat 2

Threat description	Cloning or emulating a legitimate key tag which can then be used to open the lock.
Threat target	Key tag.
Attack techniques	Use a device capable of reading, writing and/or emulating RFID tags.
Countermeasures	Use read-only UID key tags. Implement read-only UID check in RFID reader of the lock.

Table 5.2.5: Threat 3

Threat description	Attacker intercepts communications and modifies or replays communications to perform actions.
Threat target	Communications.
Attack techniques	MITM attack on any of the communications.
Countermeasures	Use encryption, authentication and integrity checks for communications.

Table 5.2.6: Threat 4

Threat description	Modifying the mobile app to change functionality.
Threat target	The mobile app.
Attack techniques	Reverse engineering the application by e.g. extracting the APK files of an Android app and modifying them.
Countermeasures	Code obfuscation can make the code less comprehensible.

Table 5.2.7: Threat 5

Threat description	Modifying a key tag, casuing it to not be accepted as a legitimate tag.
Threat target	Key tag.
Attack techniques	Use a device capable of reading and writing RFID tags to overwrite sections of the tag.
Countermeasures	Use read-only tag.

Table 5.2.8: Threat 6

Threat description	Attacker gains access to sensitive information sent over communications.
Threat target	Communications.
Attack techniques	Eavesdropping communications with a MITM attack.
Countermeasures	Use encryption for communications.

Table 5.2.9: Threat 7

Threat description	Attacker gains access to sensitive information such as passwords or keys in the code of the mobile app.
Threat target	The mobile app.
Attack techniques	Reverse engineering the application by e.g. extracting the APK files of an Android app and looking for sensitive information.
Countermeasures	Code obfuscation can make the code less comprehensible. Avoid including any sensitive information in clear text.

Table 5.2.10: Threat 8

Threat description	Attacker floods the gateway with traffic to cause denial of service.
Threat target	Gateway.
Attack techniques	DoS attacks such as SYN-flood and ACK-storm.
Countermeasures	Implement protection against DoS attacks

Chapter 6

Penetration tests

In this chapter the penetration tests are presented. Each penetration test will be presented with background/theory, method, results and discussion. The tests were chosen based on the threat model's DREAD ratings, primarily looking at damage potential, and whether the tests can be performed legally. Penetration tests involving a computer were either performed on a VM or a desktop, both running Kali Linux, as suggested in [3].

6.1 Penetration test 1 - Man-in-the-middle between mobile app and server using mitmproxy

As mentioned in 5.1.4, communications are encrypted with TLS. In this test, attempts were made to decrypt the communication between the mobile app and the cloud server to retrieve sensitive information. The test relates to threat number 1 and 6.

6.1.1 Background

A man-in-the-middle (or MITM) attack is an attack in which a malicious person has access to the communication between two parties and is able to eavesdrop and/or alter the communication is some manner.

The WebSocket protocol allows two-way communication without relying on multiple HTTP connections and uses the origin-based security model[1].

[1]A. Barth. (Dec. 2011). "The Web Origin Concept," [Online]. Available: `https://rfc-editor.`

WebSocket communication may be vulnerable to manipulating WebSocket messages, manipulating the WebSocket handshake and cross-site WebSocket hijacking[2].

The communication channels of the system are encrypted using TLS. The IETF description of TLS is as follows: ”TLS allows client/server applications to communicate over the Internet in a way that is designed to prevent eavesdropping, tampering, and message forgery”[3]. Decrypting this communication requires authentication using certificates. The program *mitmproxy* acts as a proxy server and has a certificate which can be used to decrypt the communication and then forward it to its intended receiver.

6.1.2 Method

The guides on the website of mitmproxy were followed to install the program and certificates, making configurations and running the program correctly[4]. A certificate needs to be installed on the target device, and the device's proxy settings need to be set to the proxy server's IP address.

The app was run on an Android 6 device. This was due to the fact that Android 7 and later requires decompiling, modifying and recompiling the APK-file of the mobile app, which was to be avoided for legal reasons (see 7.4.1).

The communication was monitored for the following functions:

1. Opening the app.
2. Changing user roles.
3. Unlocking the door lock.
4. Accessing the event log.

org/rfc/rfc6454.txt (visited on 2021-06-23), A. Melnikov and I. Fette. (Dec. 2011). “The WebSocket Protocol,” [Online]. Available: https://rfc-editor.org/rfc/rfc6455.txt (visited on 2021-06-23).

[2]“Testing for WebSockets security vulnerabilities,” in. [Online]. Available: https://portswigger.net/web-security/websockets (visited on 2021-06-23).

[3]E. Rescorla. (Aug. 2018). “The Transport Layer Security (TLS) Protocol Version 1.3,” [Online]. Available: https://rfc-editor.org/rfc/rfc8446.txt (visited on 2021-06-17).

[4]“mitmproxydocs,” in. [Online]. Available: https://docs.mitmproxy.org/stable/ (visited on 2021-06-17).

6.1.3 Results

When opening the app with mitmproxy HTTP messages between the app and the server can immediately be seen. We can also see messages when the event log is accessed and when a user role is changed. Figure 6.1.1 shows what sort of communication can be intercepted when the app is used.

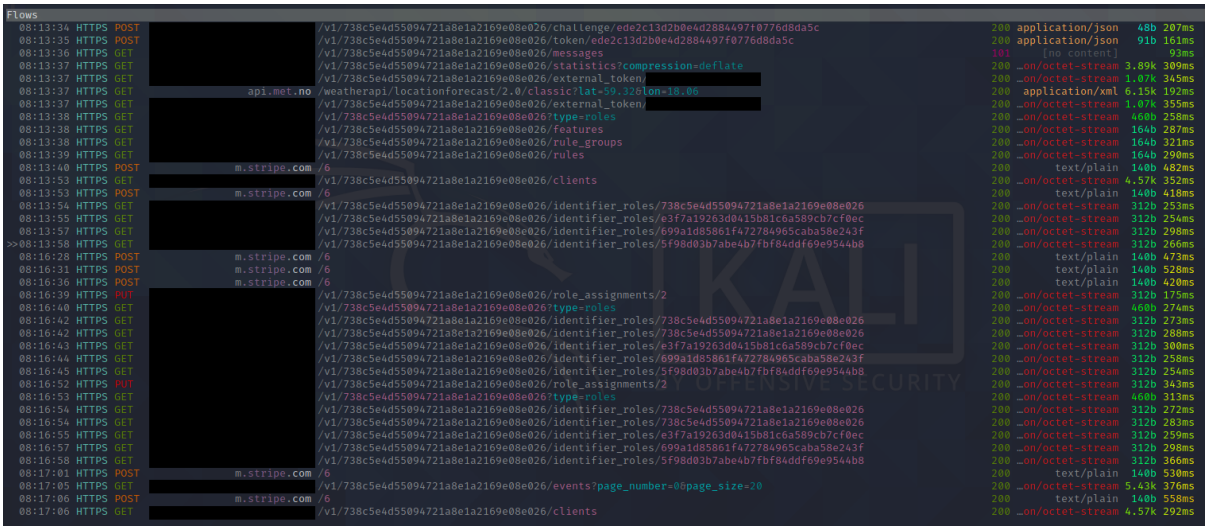

Figure 6.1.1: Captured communication using mitmproxy.

6.1.4 Discussion

ARP spoofing could be used instead of changing the proxy settings on the device[5]. It is an attack where the attacker sends spoofed ARP messages to associate its MAC-address with the IP address of another host in order to redirect traffic meant for that host to the attacker [11]. However, the certificate must still be installed on the target device.

By observing the communication between the app and the server one can get a rough idea of how the app uses HTTP to authenticate the user, change user roles and access the event log. Unfortunately, some uncertainties remain since the bodies of the HTTP messages do not contain human readable text. The text that appears in the bodies contain the same characters that base64url strings can contain but trying to decode the text as base64url does not yield a more readable result. Many messages sent from the app to the server do not contain anything in the message body, they only contain the URL and headers making it easier to figure out the workings of those messages.

[5]Valbrux, "MITM using arpspoof + Burp or mitmproxy on Kali Linux.," in. [Online]. Available: https://www.valbrux.it/blog/2017/11/26/mitm-using-arpspoof-burp-or-mitmproxy-on-kali-linux/ (visited on 2021-06-21).

A quick glance at figure 6.1.1 shows that the app mainly communicates directly with the host app.homcontrol-cloud.com. All communication with app.homecontrol-cloud.com uses a path that begins with /v1/. That is possibly the API version, but it is hard to verify. What follows is an alphanumerical string that identifies the user. That the string is an identifier can be verified by listening to communication made by another user. The HTTP requests will be identical, bar the identifier.

What follows the identifier depends on what action is being performed in the app. For example, when the event log is being accessed the identifier is followed by the word events and a query string and when the settings for access control is being accessed the word clients is used to get information about every user.

It is evident that some communication happens without HTTP. The lock can be unlocked via the app and if another user unlocks the door, using any method other than keys, the lock status is shown without an HTTP request being sent. That is because the app also communicates with the server using a WebSocket. mitmproxy does not display WebSocket communication but the HTTP headers in one of the early HTTP messages (see figure 6.1.2 and 6.1.3) shows the app switching communication protocol.

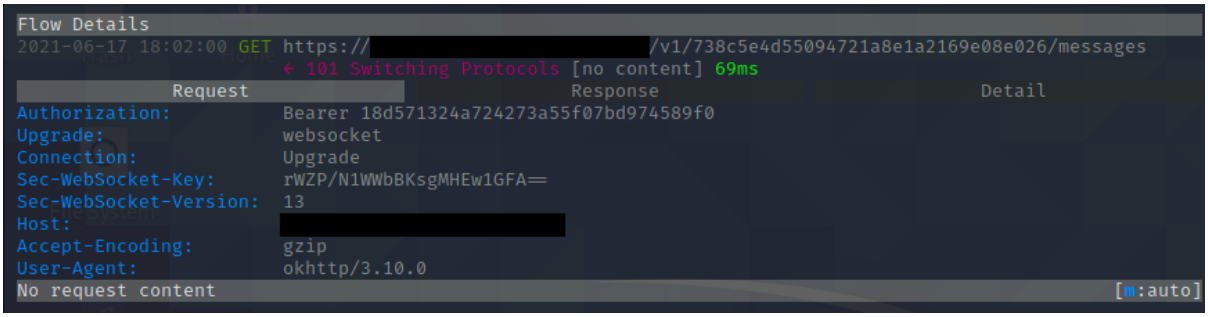

Figure 6.1.2: Client request to switch protocol.

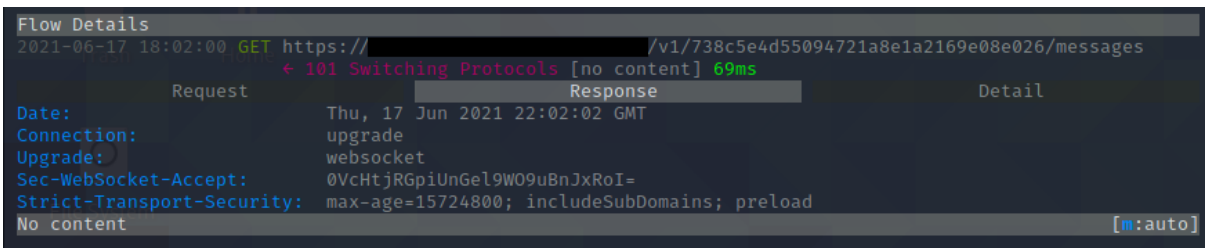

Figure 6.1.3: Server accepting protocol switch.

6.2 Penetration test 2 - Man-in-the-middle and replay attack using Burp Suite

In this test, attempts were made to intercept communications and use this to perform actions on the door lock without permission. The test relates to threat number 1, 3, and 6.

6.2.1 Background

The theory behind this attack is similar to that of 6.1. Burp Suite acts as a proxy server and has a certificate that can be installed on the target device. Burp Suite includes the additional capability to capture WebSocket traffic, which is why it was used in this test.

A replay attack means that a legitimate message is captured by the attacker and sent again at a later point in time to perform the action again[6].

6.2.2 Method

Instructions on how to install and configure Burp Suite, the certificate and mobile device were taken from Port Swigger's website[7][8].

Burp Suite acts as a proxy. A certificate needs to be installed on the target device and the proxy settings need to be set to the IP address of the device running Burp Suite.

Several different scenarios for the replay attack were tested. The sending of a message is done by selecting a message and sending it to the 'Repeater'-function of Burp Suite, followed by pressing send.

[6]"What Is a Replay Attack?" In. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/replay-attack (visited on 2021-06-17).

[7]"Downloading and installing Burp Suite," in. [Online]. Available: https://portswigger.net/burp/documentation/desktop/getting-started/installing-burp (visited on 2021-06-17).

[8]"Installing Burp's CA certificate," in. [Online]. Available: https://portswigger.net/burp/documentation/desktop/getting-started/proxy-setup/certificate (visited on 2021-06-17).

6.2.3 Scenarios and Results

As mentioned in 5.1.1, all new users that are added are given admin privileges and in the tests below it is assumed that every user starts of with the admin role.
A represents a user with administrator privileges.
H represents the hacker.
U represents a third user.
A letter not followed by an apostrophe represents a user that has access to the gateway from the beginning of the test while a letter followed by an apostrophe represents a user that is added to the gateway during the test. With the exception of the first two tests a role that is downgraded is always downgraded to the least privileged role, Guest.

Elevate user role after downgrade to role Guest

A: Downgrade U
H: Elevate U and capture elevation message
A: Downgrade H to Guest
A: Downgrade U
H: Replay elevation message

Result: An HTTP message with the response code 403 Forbidden is returned

Elevate user role after downgrade to Installer

A: Downgrade U
H: Elevate U and capture elevation message
A: Downgrade H to Installer
A: Downgrade U
H: Replay elevation message

Result: An HTTP message with the response code 403 Forbidden is returned

Unlock door after role downgrade

H: Unlock door and capture unlock message
A: Downgrade H
H: Replay unlock message

Result: Door unlocks

Unlock door after user removal

H: Unlock door and capture unlock message

A: Delete H

H: Replay unlock message

Result: Cannot reconnect to WebSocket

Elevate user role after user removal

A: Downgrade U

H: Elevate U and capture elevation message

A: Delete H

A: Downgrade U

H: Replay elevation message

Result: An HTTP message with the response code 403 Forbidden is returned

Downgrade user role after role downgrade

H: Downgrade U and capture downgrade message

A: Elevate U

A: Downgrade H

H: Replay downgrade message

Result: An HTTP message with the response code 403 Forbidden is returned

Delete user by editing and replaying deletion message

H: Delete U and capture deletion message

U': Add user U'

H: Fetch the ID of U'

H: Replay deletion message but with the ID of U replaced with the ID of U'

Result: U' is deleted

Delete user after role downgrade

H: Delete U and capture deletion message

U': Add user U'

H: Fetch the ID of U'

A: Downgrade H

H: Replay deletion message but with the ID of U replaced with the ID of U'

Result: An HTTP message with the response code 403 Forbidden is returned

Replay unlock message to different gateway/lock from underprivileged user

H: Unlock door and capture unlock message
A: Reset gateway and lock
A': Add user A'
H': Add user H'
A': Downgrade H'
H': Replay unlock message

Result: Door stays locked

Replay unlock message with unprivileged WebSocket

H: Unlock door and capture unlock message
A: Downgrade H
H: Restart app
H: Using a new WebSocket, send message captured in first step

Result: Door opens

Replay unlock message with new unprivileged user

H: Unlock door and capture unlock message
A: Downgrade H
H': Create new user H'
A: Downgrade H'
H': Start app and send message captured by H using new underprivileged user and WebSocket

Result: Door opens

Replay unlock message from deleted user with unprivileged user

H: Unlock door and capture unlock message
A: Delete H
H': Create new user H'
A: Downgrade H'
U: Start app and send message captured by H using new underprivileged user and WebSocket

Result: Door opens

6.2.4 Discussion

As long as a unlock-message can be captured, it can be used to unlock the door even though the role has been downgraded. Given that new users are given admin privileges by default, a malicious person who is added as a user could press the unlock button with a Burp Suite or similar proxy server set up to save an unlock-message before being downgraded. This message could be used to unlock the door at a later stage, as long as the malicious person has access to the app as a guest user on that gateway. Being added as a user requires access to the gateway. It is worth noting that this would show the name of the user that has unlocked the door in the event log. If the user is deleted before the event log is viewed, it will only appear as ”Deleted user”.

An alternative form of attack would require that an attacker gets privileged accesses the owner’s mobile device and app for the period of time needed to install the certificate and in cases of Android 7 and later, a modified version of the app. In this case, the attacker would be able to send and capture an unlock message to use at a later point in time, but would still require the attacker is a user on the target gateway in order to open a WebSocket.

Opening a WebSocket without having a user requires authentication as described in 6.1 and would require cracking the authentication to do without having a user and the app.

It seems like a check for the user’s role and privileges is not done during WebSocket communication.

Changing of user roles is done through HTTP-requests. Making changes to these is only possible if the user currently has the privileges to do so. A check for the user’s role is seemingly done for each request.

6.3 Penetration test 3 - Reading and emulating a key tag

In this test, attempts were made to read the contents of a key tag and emulate the key tag using a proxmark3. The test relates to threat number 2.

6.3.1 Background

RFID is a technology used to transmit data between tags and readers. The key tag is a passive tag, meaning it is activated when within a certain range of a reader, which is located in the door lock[9].

The tags and reader used for the smart door lock are NFC tags of the type Mifare Classic 1k. NFC (Near-field communication) cards are meant to be used in close proximity, usually a few centimeters apart, and are used in for example credit cards[10]. The Mifare Classic 1k card specifically is widely used, although not in credit cards, but rather in applications such as public transport fare cards.[11].

The Mifare Classic 1k is manufactured by NXP Semiconductors. It uses the standard NFC frequency of 13.56MHz and the operating distance is specified to be up to 100mm, but depends on the placement of antennae and reader configuration. It has a memory of 1kB, organized in 16 sectors of 4 blocks with each block consisting of 16 bytes. The first block (block 0) of the first sector (sector 0) contains the UID (unique identifier) of the card, which is write-protected, meaning it can only be read. There are other cards, compatible with Mifare Classic 1k readers, where the UID is not write-protected.[12]. Each sector has a trailer, containing a 'key A' followed by 4 bytes defining the access conditions of the sector, followed by a 'key B', which is optional and may be used for data instead. The default value of the keys is 'FFFF FFFF FFFF. The encryption used is NXP's own Crypto-1, which has been rendered insecure [2].

The proxmark3 RDV4 is a tool for reading, writing and emulating RFID developed by RFID Research Group.

[9] "Automatic Identification and Data Collection (AIDC)," in. [Online]. Available: https://www.mhi.org/fundamentals/automatic-identification (visited on 2021-06-16).

[10] "What is NFC? Everything you need to know," in. [Online]. Available: https://www.techradar.com/news/what-is-nfc (visited on 2021-06-16).

[11] "MF1S50YYX V1," in. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/MF1S50YYX_V1.pdf (visited on 2021-06-16).

[12] "MIFARE CLASSIC® COMPATIBLE 1K TAG - UID CHANGEABLE," in. [Online]. Available: https://lab401.com/products/mifare-compatible-1k-uid-modifiable (visited on 2021-06-16).

6.3.2 Method

For this attack, the instructions of HackerWarehouse.TV were followed[13]. Although the guide uses a proxmark3 RDV4 configured to be compatible with an android application and a Bluetooth module, the same functionalities were achieved using a proxmark3 RDV4 connected to a computer and the computer's terminal. The proxmark3 RDV4 was used to read and emulate the key tag and a Kali Linux VM was used to flash the required firmware onto the device and to run commands on the proxmark3 RDV4.

The first step is to run the command *hf search* in order to confirm the tag type and acquire the UID of the tag.

Next, the command *hf mf darkside* is run to perform a darkside attack on the tag to acquire a valid key.

The valid key is then used to perform a nested attack using the command *hf mf nested 1 0 A key t*. The command generates the tag data, which can then be saved into files using the command *hf mf dump*.

The data is then loaded to the proxmark3 using *hf mf eload -path to .eml file*. The .eml file is one of the files generated by the *hf mf dump* command from the data from the nested attack.

The command *hf mf sim* is then run to simulate the tag. The proxmark3 should now be acting like the targeted tag.

6.3.3 Results

Reading the contents of the key tag was done successfully. All the sectors, including keys and access conditions, contained the standard values from manufacturing.

The proxmark3 was used like a key tag during emulation, but the smart door lock did not unlock. Some of the tries resulted in a quick flash of green on the touchscreen of the lock.

[13]"Decrypting and Emulating Mifare 1K Cards using the RFID Tools Android App," in. [Online]. Available: `https://hackerwarehouse.tv/product-knowledgebase/proxmark/decrypting-and-emulating-mifare-1k-cards-using-the-rfid-tools-android-app/` (visited on 2021-06-16).

6.3.4 Discussion

The keys having standard values makes the tag easier to read and clone, and indicates that the data in each sector is not used by the reader. This means less time is needed with the tag, and possibly that only the UID is needed to emulate the tag.

The green flash usually indicates that unlocking the door lock has been successful, but the smart door lock did not unlock. This indicates that countermeasures for detecting emulated or cloned tags have been implemented. One such countermeasure could be checking if the tag's UID is writable[10]. A key tag with a one-time-only writable UID was ordered but not received in time to test. It could be interesting to try to use a tag like that.

Chapter 7

Discussion

7.1 Method

STRIDE and DREAD as modeling frameworks were chosen to due to their prevalence in previous work as well as being the frameworks of choice in the ”IoT penetration testing cookbook: identify vulnerabilities and secure your smart devices”[4]. These frameworks facilitated the threat modeling and provided a good overview of what attack surfaces could be tested. The book includes a step of listing use cases which was not done in this study due to the reconnaissance done before the threat modeling, which included testing the functionalities of the system as a user.

7.2 Results

The results of the eavesdropping penetration test using mitmproxy did not result in any serious sensitive information exposure. The API of the cloud server was not human-readable and could not be decoded or decrypted. The most useful information gathered was that an upgrade to a WebSocket protocol is done, as well as identifiers for users. The test required installing a certificate and configuring proxy settings on the target device, meaning that access to the device would be necessary for an attacker. On devices using Android 7 and later a modified version of the app would have to be installed in addition to the previous requirements.

The penetration test using Burp Suite to capture and replay messages had requirements similar to those of the test using mitmproxy. A certificate needed to be

installed and the proxy settings needed to be set correctly on the target device. On Android 7 and later, the app would need to be modified for it to accept the certificate. Some weaknesses were found in the WebSocket communication, where the unlock-message from the app could be reused by users lacking privileges to unlock the lock. To start a WebSocket through the proxy, an attacker needs to be a user associated to the target gateway. This is because opening a WebSocket requires authentication, which would need to be cracked to open one without being a user.

Emulating the key tag was done successfully. However, the lock did not accept the emulated tag to open the door lock. The lock did start to indicate that authentication of the tag had been successful by flashing green, but it remained locked nonetheless. This attack would require access to a legitimate key tag for the period of time required to perform the necessary attacks, which in this case took less than 2 minutes.

7.3 Project

Many of the vulnerabilities found in previous studies were found to be non-factors in the reconnaissance stage of this project. The architecture of the system is formed in such a way that any username and password-related exploits are not possible due to user authentication being handled by keys rather than username and password. State consistency vulnerabilities relating to internet connection is not possible simply because the app requires an internet connection to function at all. The system does not use any Bluetooth connections, which was a common attack surface in related work. The system also does not include features such as unlocking based on the owner's location.

There are several attack surfaces and penetration tests which were not explored in this project due to time limitations and unclear legality of the tests. For example, searching for vulnerabilities in the code of the android application and its related files or trying to access the server without permission could have yielded results, but were not tried. There were plans to test the RF communication using Zigbee, which may be vulnerable to several types of attacks [9]. However, equipment needed to capture and read Zigbee data was not available during the project. The firmware and hardware were not examined due to time constraints, meaning there may be undiscovered vulnerabilities related to those attack surfaces.

Investigating the files system would have been particularly interesting considering the findings of the study mentioned in 3.4.

7.4 Ethics and Sustainability

7.4.1 Ethics and the Law

When conducting penetration tests, it is important to stay within the bounds of the law. For this reason, a decision was made to avoid the following:

1. Any attempts infringing on or disrupting a server not belonging to us or KTH.
2. Reverse engineering code, due to lack of clarity as to what is allowed according to copyright laws.

The decision was based primarily on the laws Brottsbalken 4 kap. 9c § Lag(2014:302)[1] and Lagen om upphovsrätt till litterära och konstnärliga verk 2 § Lag(2005:359)[2].

7.4.2 Sustainability

The sustainability aspect of this project is mainly related to the life cycle of the product we are testing. It is likely that a product that is known to be secure will be used for a longer period of time than one that has flaws in its security.

[1] "Brottsbalk (1962:700)," in. [Online]. Available: https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/brottsbalk-1962700_sfs-1962-700 (visited on 2021-06-16).

[2] "Lag (1960:729) om upphovsrätt till litterära och konstnärliga verk," in. [Online]. Available: https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/lag-1960729-om-upphovsratt-till-litterara-och_sfs-1960-729 (visited on 2021-06-16).

Chapter 8

Conclusions

The attack surfaces of the smart door lock system that were tested showed no significant vulnerabilities. Some weaknesses were found in the WebSocket communication. Exploiting these weaknesses by the method presented in this report require circumstances that are unlikely to arise during normal use. Having some vigilance as the owner of the lock, such as only adding trusted users, protecting the device with the app and turning on notifications for new users, will prevent these exploits.

The results of the tests in this report could indicate vulnerabilities in the system if some changes to the method or more extensive testing is done.

The system was secured against many previously found vulnerabilities by not including many of the vulnerable features found in those systems.

8.1 Future Work

The hardware, firmware and RF communication were not tested in this study, and are all suggestions for future work regarding penetration testing of the smart door lock system. It would also be interesting to try using a one-time-only writable UID RFID tag when cloning the key tag, as mentioned in 6.3.4.

References

[1] F. Callegati, W. Cerroni, and M. Ramilli, "Man-in-the-middle attack to the https protocol," *IEEE Security Privacy*, vol. 7, no. 1, pp. 78–81, 2009. DOI: 10.1109/MSP.2009.12. (visited on 2021-06-23).

[2] R. V. Carlo Meijer, "Ciphertext-only cryptanalysis on hardened mifare classic cards," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15, Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 18–30, ISBN: 9781450338325. DOI: 10.1145/2810103.2813641. [Online]. Available: https://doi.org/10.1145/2810103.2813641 (visited on 2021-06-21).

[3] M. Denis, C. Zena, and T. Hayajneh, "Penetration testing: Concepts, attack methods, and defense strategies," in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2016, pp. 1–6. DOI: 10.1109/LISAT.2016.7494156. [Online]. Available: https://ieeexplore.ieee.org/document/7494156 (visited on 2021-06-24).

[4] A. Guzman and A. Gupta, *IoT Penetration Testing Cookbook*. Packt Publishing, 2017, ISBN: 9781787280571.

[5] R. Hassani, "Security evaluation of a smart lock system," B.S. thesis, KTH, School of Engineering Sciences in Chemistry, Biotechnology and Health (CBH), 2020, p. 46. [Online]. Available: https://kth.diva-portal.org/smash/record.jsf?pid=diva2:1533957 (visited on 2021-06-21).

[6] G. Ho, D. Leung, P. Mishra, A. Hosseini, D. Song, and D. Wagner, "Smart locks: Lessons for securing commodity internet of things devices," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16, Xi'an, China: Association for Computing Machinery, 2016, pp. 461–472, ISBN: 9781450342339. DOI: 10.1145/2897845.2897886.

[Online]. Available: https://doi-org.focus.lib.kth.se/10.1145/2897845.2897886 (visited on 2021-06-23).

[7] Y. H. Hwang, "Iot security & privacy: Threats and challenges," in *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security*, ser. IoTPTS '15, Singapore, Republic of Singapore: Association for Computing Machinery, 2015, p. 1, ISBN: 9781450334495. DOI: 10.1145/2732209.2732216. [Online]. Available: https://doi.org/10.1145/2732209.2732216 (visited on 2021-06-23).

[8] L. Jiang, D.-Y. Liu, and B. Yang, *Smart home research*. 2004, vol. 2, 659–663 vol.2. DOI: 10.1109/ICMLC.2004.1382266.

[9] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8182–8201, 2019. DOI: 10.1109/JIOT.2019.2935189. (visited on 2021-06-23).

[10] P. Moravec and M. Krumnikl, "Developing countermeasures against cloning of identity tokens in legacy systems," in *Computer Information Systems and Industrial Management*, K. Saeed, W. Homenda, and R. Chaki, Eds., Cham: Springer International Publishing, 2017, pp. 672–684, ISBN: 9783319591056. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-59105-6_58 (visited on 2021-06-23).

[11] V. Ramachandran and S. Nandi, "Detecting arp spoofing: An active technique," in *Information Systems Security*, S. Jajodia and C. Mazumdar, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 239–250, ISBN: 978-3-540-32422-5.

[12] C. Robberts and J. Toft, "Finding vulnerabilities in iot devices : Ethical hacking of electronic locks," B.S. thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2019, p. 23. [Online]. Available: http://kth.diva-portal.org/smash/record.jsf?pid=diva2:1334605 (visited on 2021-06-21).

[13] A. Shostack, *Threat Modeling: Designing for Security*, 1st. Indianapolis: Wiley, 2014, p. 22. [Online]. Available: https://books.google.se/books?hl=sv&lr=&id=YiHcAgAAQBAJ&oi=fnd&pg=PR21&dq=threat+modeling&ots=eUOoNv8RQv&sig=VTFhJo11fqjakAQB5-lG0olv-Nw&redir_esc=y#v=onepage&q&f=false (visited on 2021-06-23).

[14] ——, *Threat Modeling: Designing for Security*, 1st. Indianapolis: Wiley, 2014, p. 66. [Online]. Available: https://books.google.se/books?hl=sv&lr=&id=YiHcAgAAQBAJ&oi=fnd&pg=PR21&dq=threat+modeling&ots=eUOoNv8RQv&sig=VTFhJo11fqjakAQB5-lG0olv-Nw&redir_esc=y#v=onepage&q&f=false (visited on 2021-06-23).

[15] ——, *Threat Modeling: Designing for Security*, 1st. Indianapolis: Wiley, 2014, pp. 61–85. [Online]. Available: https://books.google.se/books?hl=sv&lr=&id=YiHcAgAAQBAJ&oi=fnd&pg=PR21&dq=threat+modeling&ots=eUOoNv8RQv&sig=VTFhJo11fqjakAQB5-lG0olv-Nw&redir_esc=y#v=onepage&q&f=false (visited on 2021-06-23).

[16] *Smart Home Appliances Market Size, Share & Trends Analysis Report By Product (Washing Machines, Refrigerators, TVs, Air Purifiers), By Distribution Channel (Online, Offline), By Region, And Segment Forecasts, 2020 - 2027*. Grand View Research, 2020. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/smart-home-appliances-market (visited on 2021-06-07).

[17] A. Viderberg, "Security evaluation of smart door locks," M.S. thesis, KTH, School of Electrical Engineering and Computer Science (EECS), 2019, p. 69. [Online]. Available: http://www.diva-portal.org/smash/record.jsf?pid=diva2:1336796 (visited on 2021-06-16).

[18] W. Xiong and R. Lagerström, "Threat modeling – a systematic literature review," *Computers & Security*, vol. 84, pp. 53–69, 2019, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2019.03.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818307478 (visited on 2021-06-26).

[19] T. Yamazaki, "Beyond the smart home," in *2006 International Conference on Hybrid Information Technology*, vol. 2, 2006, pp. 350–355. DOI: 10.1109/ICHIT.2006.253633. [Online]. Available: https://ieeexplore.ieee.org/document/4021238 (visited on 2021-06-19).

[20] M. Ye, N. Jiang, H. Yang, and Q. Yan, "Security analysis of internet-of-things: A case study of august smart lock," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2017, pp. 499–504. DOI:

10.1109/INFCOMW.2017.8116427. [Online]. Available: https://ieeexplore.ieee.org/document/8116427 (visited on 2021-06-21).

[21] Z.-K. Zhang, M. C. Y. Cho, C. W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: Ongoing challenges and research opportunities," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, 2014, pp. 230–234. DOI: 10.1109/SOCA.2014.58. (visited on 2021-06-23).

TRITA-EECS-EX-2021:551