Abstract

The Internet of Things (IoT) market has expanded and the variety as well as quantity of IoT devices has been rapidly increasing. Which also bring new security threats as new vulnerabilities are found within IoT devices. One area where these devices are being used is in smart homes and where smart locks especially needs to be designed in a secure way. This study aims to assess the security of a smart lock unit. We first present a background including threat modelling and previously found vulnerabilities. Then a methodology section for different attacks performed as well as the results from doing them. Finally, a discussion where we assess the security implications of the results.

The conclusion from the results are that the smart lock exhibits vulnerabilities in its design. Specifically in its file system encryption, resistance to disruption attacks and the consistency of access granted to guests.

Contents

Chapter 1

Abbreviations

IoT Internet of things

MAL Meta attack language

CAD Computer-aided design

IoT Internet of things

DoS Denial-of-service

MITM Man in the middle

IP Internet protocol

TLS Transport layer security

SSL Secure sockets layer

ARP Address resolution protocol

DDoS Distributed-denial-of-service

ID Identifier

SMS Short message service

BLE Bluetooth low energy

OWASP Open web application security project

MQTT Message queuing telemetry transport

SYN Synchronize

TCP Transmission control protocol

Chapter 2

Introduction

When considering the rapid increase in IoT devices on the market and the related questions of security research has showed that there is a pronounced risk when implementing these devices. A threat report done by Palo Alto Networks Unit 42 research team in 2020 concludes that 98% of all IoT device traffic is not encrypted.[1] The report further found 57% of IoT devices to be vulnerable to medium- or high-severity attacks. Which means many enterprises and owners thus are at risk. Something that makes the continuous work into security testing and threat analysis over these devices all the more important.

Smart homes refers to the fact that more and more of the different devices we have in our homes are being made remotely accessible through smartphones using either the internet or other network protocols such as Bluetooth. Thus allowing the users to control various parts of their home without directly interacting with the different devices itself or even set them up to run automatically.

Smart locks grants users some quality of life features such as; granting or denying access to visitors remotely, lock or unlock the door through a smartphone app, the ability to monitor who is coming and going in real-time and more.

Smart locks can be considered as the most essential asset in a smart home to be secure since it effectively protects everything else in one's home. However there is also additional threats to take into consideration with a smart lock outside of physical threats as there also exists digital ones [1][2].

[1] Unit 42. "2020 Unit 42 IoT Threat Report". https://unit42.paloaltonetworks.com/iot-threat-report-2020/ Last accessed 15May2021. 2020

Although smart locks might also offer additional security. For example, in the form of a video entry system that may help in preventing forced entries. There is still a great need to properly test the security these devices aim to offer. Since smart devices create a grater span of potential risks there exists a need to make sure no unwanted vulnerabilities are found.

2.1 Purpose

In this thesis we will explore how secure a smart lock is in terms of threat by performing penetration tests. In other words we will perform ethical hacking, which is hacking with legal authorization and for a good cause, such as finding vulnerabilities in systems. In order to keep consumer products safe there exists a need for ethical hacking [3]. This need for penetration testing can also be seen in both small and more sizeable organizations [4].

2.1.1 Research Question

The research question for this research would be as follows:

> *How secure is the smart lock unit when looking at previously found vulnerabilities in smart locks and IoT systems?*

2.2 Ethics and laws

2.2.1 Responsible disclosure

If threats are found within the product the person that found the threat is responsible for informing the producer about the potential breach in security. Allowing them the time to develop a patch to fix the problems discovered. If a patch is not produced or a response from the producer is not given within a reasonable time frame the results should be reported to the National Vulnerability Database and made public to make sure the potential users are made aware of the threats. The recommended time frame before releasing the results is 90 days.

2.2.2 Ethics

In general hacking is considered unethical due to how an attacker would use or access a product in an unintentional and most often illegal ways. Were the attacker not only goes against the will of the company affected through breaking the terms of service given with the product or causing unintentional damage at the cost of the company. With the attacker also breaking multiple laws in the process.

Comparably, ethical hacking is instead the act of hacking with good intention. Were the goal is to ensure that the product is secure and cannot be used maliciously. Something that is often done with specific permission from the company of the product being hacked as a means for producing secure products and ensuring that they remain as such.

2.2.3 Laws

When in comes to the laws that relates to the topic of ethical hacking, the following Swedish laws are to be considered:

- Brottsbalken 4 kap. 9c §: "Den som olovligen bereder sig tillgång till en uppgift som är avsedd för automatiserad behandling eller olovligen ändrar, utplånar, blockerar eller i register för in en sådan uppgift döms för dataintrång till böter eller fängelse i högst två år. Detsamma gäller den som olovligen genom någon annan liknande åtgärd allvarligt stör eller hindrar användningen av en sådan uppgift. Är brottet grovt, döms för grovt dataintrång till fängelse i lägst sex månader och högst sex år. Vid bedömande av om brottet är grovt ska det särskilt beaktas om gärningen har orsakat allvarlig skada eller avsett ett stort antal uppgifter eller annars varit av särskilt farlig art. Lag (2014:302)."
- Lagen om företagshemligheter (brand new): "1 § Lagen innehåller bestämmelser om skadestånd, vitesförbud och straff vid obehöriga angrepp på företagshemligheter."

- Lagen om upphovsrätt till litterära och konstnärliga verk: “2 § Upphovsrätt innefattar, med de inskränkningar som föreskrivs i det följande, uteslutande rätt att förfoga över verket genom att framställa exemplar av det och genom att göra det tillgängligt för allmänheten, i ursprungligt eller ändrat skick, i översättning eller bearbetning, i annan litteratur- eller konstart eller i annan teknik.”

- 8 § Den som olovligen bereder sig tillgång till ett meddelande, som ett post- eller telebefordringsföretag förmedlar som postförsändelse eller i ett elektroniskt kommunikationsnät, döms för brytande av post- eller telehemlighet till böter eller fängelse i högst två år. Lag (2012:280).

In general the owner of the device hacking the product and systems that are entirely their own and that does not affect any other users is allowed. However what is not allowed is hacking the company, other users or any cloud services without permission from the company or user in question when concerning ethical hacking. The terms of service for the locking unit and smartphone app is also something to consider. Since they usually specify what is allowed and what is not.

Chapter 3

Background

3.1 Smart locking unit

The smart lock unit can be installed on more or less any door. Replacing the standard mechanical locking unit and the need for physical keys

3.1.1 Features

Included with the main locking unit is a smarthub that acts as a wi-fi bridge to allow the lock to be connected to the home network. Which then allow the lock to be operated at a distance using the accompanied app.

There are three different methods of operating the smartlock. Using the smartphone app, a keyfob or a personal pincode. With the latter two being possible to combine to minimise the risk if the keyfob was to be lost. It is also possible for the owner to allow temporary or time sensitive access to other users.

With the smartphone app itself allowing for two different methods of access. Either through the home network using the connected wi-fi bridge or through a Bluetooth connection directly between the phone and the lock. With the latter being limited to close range access only due to the limitations of the direct connection.

Some additional features included for the lock are the option to set automatic locking as well as the locking unit having a built in doorbell, clock and calendar.

With some security features being the mechanical lock accessible from inside the house as well as the option to temporarily charge a dead lock from outside using a 9v battery. Which allows users to exit and enter the house in emergency cases were the locking unit might otherwise be inaccessible. With all physical access points and the basic structure of the lock being shown in figure

3.2 Threat modeling

In order to create a secure application it is important to understand how the application works and what potential threats and vulnerabilities could be found within the system. Which would allow for security measures to reduce the overall security risks to be done within the creation process of the application. One way to accomplish this would be to utilise threat modeling to get an overall view on the greatest security risks of the system and how potential attacks could take place [5].

With the four main questions to work through during the threat modelling process being as follows [6]:

- What is being built?
- What potential risks are there in the system?
- What preventive measures can be done to prevent the risks?
- Was the analysis done decently?

The first step is to work out the details and flow of the system being created. Something that can be done using the various tools available online or simple pen and paper. With the important details to include being the flow of data, the different components the data can flow between and finally the restrictions put in place for the connections made. With an example being what kind of data that could be sent between two specific components and who can access this data.

Once the diagram has been created one could then move on to discovering all potential threats to the system in question. Something that would be the main part of the creation of a threat model [7].

Were the main questions to answer would be as follows:

- Where is the application most vulnerable?
- What threats are most relevant?
- How to prevent these threats?

For this thesis the threat modeling tool SecuriCAD will be used in combination with a meta attack language (MAL).

3.3 MAL

MAL can be used to assess cyber security in systems through attack simulations[8]. Instead of building new attack graphs for each new system one wants to access, MAL offers generic attack logic that can be used in domain specific scenarios. As mentioned before when setting up a threat model one must first identify all possible threats, which can be challenging. MAL in combination with securiCAD simplifies the process for the security assessor by running virtual penetration tests on a given system. Since the security assessor does not need to know what security protocols or how weakness can be exploited the main challenge thus becomes to gather information about the system.

3.4 SecuriCAD

SecuriCAD is a computer-aided design, CAD tool provided by foreseeti[1] that is used to model an IT environment. That then using the research and knowledge acquired from within the field allows for visualisation of potential threats and

[1] Foreseeti. https://foreseeti.com/securicad/?gclid=CjwKCAjwv_iEBhASEiwARoemvMSP-RTAeAGraku0P03cy_rxhxOWyAFaKdGAtk_wV3DVLQItKa0dKxoCKwIQAvD_BwE Last accessed: 19May2021. 2021

- User
- Group
- Data
- Credentials
- Exploit
- Information
- PhysicalZone
- RoutingFirewall
- SoftwareProduct
- Vulnerability
- Exploit

3.5 Penetration testing

3.5.1 Vulnerabilities

Decoding communication messages

Decoding communication messages - Can the messages sent to and from the lock be decoded and understood to a point that the security is compromised

Denial of Service

A DoS attack is an attempt by an attacker to make a service inaccessible to the user of the product. Something that is usually done by overwhelming the system in some way.

Man in the middle

MITM attack is when the attacker attempts to listen in on the communication done between two different systems. Which is done by redirecting traffic to include the hacker before it is sent on to the actual target and something that can be done by having the attacker pretend to be the router in a network connection.

Password attacks

Password attacks are any attempts made by an attacker to gain access to the account of a user through guessing or cracking the password. With password cracking entailing the use of a program to go through and test possible passwords until the correct one is found. Something that is mainly done either by using a dictionary of known common passwords or simply going through any possible viable password combination.

Data tampering

Data tampering includes any attack that alters or otherwise changes the data sent between different devices or stores within one. Something that that is done as an attempt to compromise the device and can be done on the firmware to try and gain unauthorised access.

Chapter 4

Previous work

In order to better understand the security of IoT devices arguably the best way is to look at previous work. By gaining an understanding of previously found vulnerabilities finding and anticipating new ones becomes a simpler task.

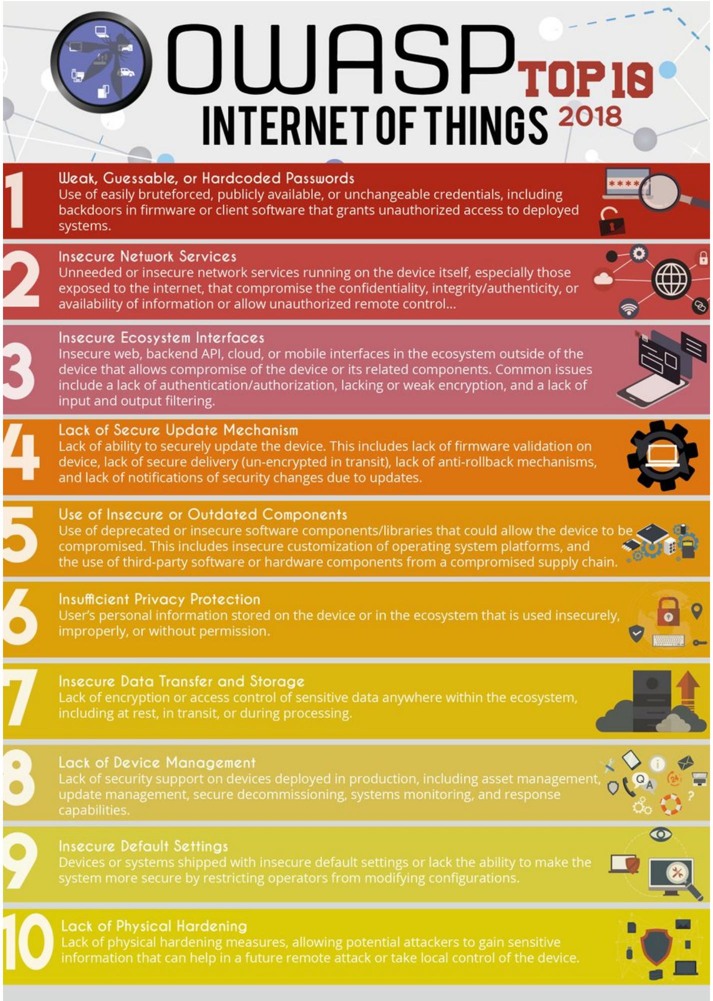
One study points to IoT systems to generally not be designed in a secure way [10]. The article also highlights the importance of designing IoT devices in a secure way since our society relies on them more and how the consequences could be potentially fatal. The countermeasures suggested is a standardisation of mitigation techniques as well as constant testing and patching. One study considered four categories of attack; physical, network, software and encryption [11]. The main finding was that lesser known attack vendors needed a stronger focus on security measures. They also showed that all four attack categories were viable for several IoT devices.

According to one article composed of a literature review of homebased IoT devices there exists many different components in said devices which amplifies the vulnerabilities [12]. Mobile application were found to have a significant control over the functionality of IoT devices and limiting the control could help in reducing potential attack impact. Vulnerabilities associated with IoT cloud endpoints are somewhat lacking in the literature. Some IoT cloud platforms rely on third-party infrastructure which shifts the security responsibilities. What communication protocols are used are not well studied and thus needs further research. However some research has found IoT devices using insecure protocols like UpnP (Universal plug and play). Also highlights how customers will not know themselves how vulnerable the devices or mobile application are to weak encryption or MITM attacks unless they test themselves. Even TLS/SSL implementations shows vulnerabilities. The suggested countermeasures to these vulnerabilities being system patching although it is an area which could be further improved upon.

A Defcon speaker and hacker who goes by the handle Jmaxxz found several vulnerabilities in the August smart lock, which is considered to be one of the most secure smart locks [13]. Guests being able to turn themselves into admins, firmware not being signed, enabling debug mode through app functionality to bypass Secure Socket Layer ping are some of the security issues found. Anthony Rose and Ben Ramsey also found vulnerabilities but on several different smart locks. Password being transmitted in clear text, reverse engineering mobile applications to identify sensitive information, fuzzing, spoofing and susceptibility to replay-based attacks being the found vulnerabilities.

4.1 OWASP

The Open Web Application Security Project (OWASP) IoT project covers the top 10 vulnerabilities for IoT devices. Which includes; weak guessable, hardcoded password, insecure network services, insecure ecosystem interfaces, lack of secure update mechanism, use of insecure or outdated components, insufficient privacy protection, insecure data transfer and storage, lack of device management, insecure default settings, lack of physical hardening as can be seen in figure 4.1.



OWASP TOP 10 INTERNET OF THINGS 2018

1 Weak, Guessable, or Hardcoded Passwords
Use of easily bruteforced, publicly available, or unchangeable credentials, including backdoors in firmware or client software that grants unauthorized access to deployed systems.

2 Insecure Network Services
Unneeded or insecure network services running on the device itself, especially those exposed to the internet, that compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control...

3 Insecure Ecosystem Interfaces
Insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.

4 Lack of Secure Update Mechanism
Lack of ability to securely update the device. This includes lack of firmware validation on device, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.

5 Use of Insecure or Outdated Components
Use of deprecated or insecure software components/libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.

6 Insufficient Privacy Protection
User's personal information stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.

7 Insecure Data Transfer and Storage
Lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.

8 Lack of Device Management
Lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.

9 Insecure Default Settings
Devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.

10 Lack of Physical Hardening
Lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device.

Figure 4.1: A graphic over the OWASP top 10 vulnerabilities of IoT devices.[1]

4.2 Denial of service

DoS attacks on smart locks needs some extra thought put into its countermeasures since in case of an emergency like a fire, it is crucial to able to leave one's home [14]. The device needs to stay “fail open" which means the door would be left in a unlocked state in case the system fails. A DoS attack may

[1] "OWASP Internet of Things Project".
https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Projecttab=IoT_Top_10
Last accessed: 14May2021. 2019

therefore prove to be serious risk to one's home since the attack effectively have the same outcome as unlocking the smart lock.

Methods for DoS attacks on IoT systems have been proven to work by previous studies [15] [16]. Both studies explore how effective certain DoS attacks are by verifying the communication disruption by said attacks. This is accomplished through monitoring the time for the first package to be lost and the package lose rate, i.e, at what rate a packet fails to reach their destination.

DoS attack has been shown to work and be a security vulnerability for smart locks in the past [17]. The authors of the mentioned study wrote script in Kali Linux in combination with a arduino microprocessor connected to the smart lock and the blynk app for authentication purposes. They successfully stopped the device from working properly by using Slowris which is a type of Dos attack.

4.3 Man in the middle

Unencrypted or weak encrypted communication has been shown as a weakness for IoT systems [12][18][19][20].

A study on vulnerabilities of a web camera found no encryption were being used in the device [18]. That is to say, sensitive information such as login account details and IP camera registration were sent in plain text. They further found that Wi-Fi password were being sent to the IP camera servers as clear text. In other words the device and potentially more devices on the local network are thus vulnerable to an MITM attack.

MQTT which a messaging protocol commonly used in IoT devices has been found as exhibiting vulnerabilities [20]. They also found that there exists other encryption algorithms that can protect sensitive information. They proposed a wireless sensor network authentication encryption method based on the SM9 algorithm.

Users not being able to know themselves how vulnerable their device or mobile application are to weak encryption or MITM attacks unless they test themselves has been brought up as a concern [12].

User credentials such as, user codes, passwords and usernames have been previously successfully sniffed in a smart lock unit [21]. The study used mitmproxy and ARPspoof. However this attack can only be accomplished by installing a certificate on the targeted device. Meaning it is only possible if the hacker has access to the device.

A Bachelor's thesis has shown that even when there exists encryption in communication from the smart lock there exists certain patterns [22]. By listening on the communication they were able to categorize six different interactions.

4.4 Reverse engineering

Reverse engineering an application can be used to find vulnerabilities that can be exploited in other attack vectors [23][24]. A study on the security of August smart lock performed four types of attacks by utilizing reverse engineering [23]. The attacks are the following; Handshake Key Leakage Attack, Owner Account Leakage Attack, Personal Information Leakage Attack and DoS Attack.

One study used reverse engineering to find where accounts resets occurred in order to be perform a brute force attack [24]. Another study used static source code analysis to find a couple of security weaknesses in the Samsung-owned SmartThings [19]. Firstly, they found that many of the apps were overprivileged, that is to say a SmartApp is given full access even if only limited is needed. Secondly, they found the communication were executed in a unsafe way and sensitive information not being properly protected. Finally, they found the SmartThing event subsystem design to be insecure.

4.5 Handshake key leakage attack and Personal information leakage attack

In order to establish a secure connection over TLS between a client and server sometimes a certificate and a key is needed from the client side. A handshake key is needed because of authentication purposes. One study found the handshake key to be stored locally and with no encryption in a smart lock unit [23].

This poses a vulnerability for the locking unit since with the key unofficial communication, that is to say, communication without the official mobile application is possible. The study also found that there was an overall lack of local encryption. Which meant the database folder was unprotected and personal information such as usernames and profile pictures could be retrieved.

4.6 Brute forcing

One study found vulnerabilities in how IoT devices handled account password resets by utilizing reverse engineering [24]. According to the article current security countermeasures against brute force attack to SMS authentication code are insufficient. They developed a automated tool for a brute force attack and succeeded in showing that brute forcing can be used to break into IoT devices.

A master thesis brings up the potential for a brute force attack on two different smart locks [25]. Both locks lacked the recommended password policies as well as seemingly allowing multiple failed attempts without blocking further attempts. Thus the passwords can easily be brute forced as long as enough time is provided.

4.7 State consistency

A master thesis found vulnerabilities with state consistency as the application did not properly update who has access to unlock the door [25]. Temporary guest could still access the lock even though their privilege had been revoked. However this only worked when the hub was disconnected and the phone lacked a internet connection. A bachelor thesis found similar results with guest not being denied access if they have their internet turned off [26].

4.8 Bluetooth

Study done on saftey of Bluetooth Low Energy (BLE) [27]. They found that BLE devices are vulnerable to information leakage and even malicious control.

They used tools for connection degrading, MITM attacks, Bluetooth signal jammer as well as Wireshark to sniff the traffic.

Chapter 5

Methods

5.1 Threat modelling with SecuriCAD

SecuriCAD was used to graphically represent the structure of the connected system. Which includes the locking unit as well as the simulated possible attacks that could be done on the system.

Threat modeling tools have been proven to improve the results compared to a manual approach when a lack of expert knowledge without the field is held by the modellers [28]. Although this may result in specific threats being missed due to certain threats not being found by the tool in use. It still provides a good initial view into the security of a system without requiring the experience and greater knowledge within the field that would be required for the manual approach. With SecuriCAD being one such tool available on the market that was deemed suitable for this thesis with the probabilistic approach the tool provides.

To do the actual modelling the MAL language corelang was used. With the model created to represent the connected system and data transfer between the units connected to the locking unit can be seen in figure 5.1.

The simulations was done using multiple different access points for the potential hacker to attempt to compromise the system through. With the different points being as seen below:

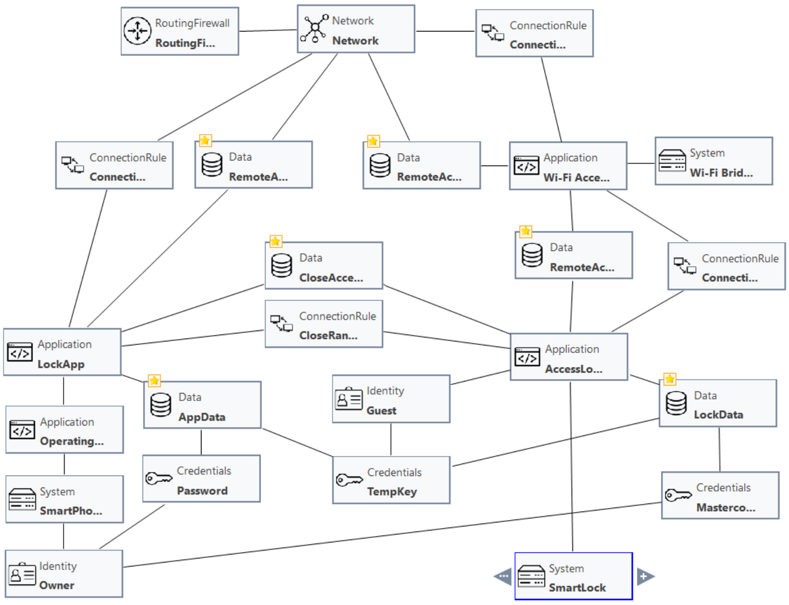


Figure 5.1: The securiCAD model using the corelang MAL language over the locking unit and its connections

- Applications
  - LockApp - The smartphone application that allows the user to control and program the locking unit without directly engaging with the lock
  - AccessLock - The locking unit application that allows for the lock to be used
  - Wi-FiAccess - The wi-fi bridge application that allows the locking unit to connect to the local network
- System
  - Smartphone - The smartphone that the lock app is installed on and that is thus used to operate the lock
  - SmartLock - The main system for the locking unit
  - Wi-Fi Bridge - The main system for the wi-fi bridge
- Connection rules
  - Network connection - The wi-fi connection between any device connected to the network through the router

  - device connection - The Bluetooth connection between different devices

- Network - The local network that the locking unit is connected to using the wi-fi bridge

- Identities

  - Owner - The owner of the locking unit with administrative privileges for operating the lock
  - Guest - A guest that has been allowed temporary access to the home and minimal privileges for operating the lock

- Data

  - AppData - The data stored within both the locking unit as well as the smartphone app
  - CloseAccess - The data sent between the locking unit and the phone when accessing the lock within close range
  - RemoteAccess - The data sent over the network between the phone and the smart hub and thereafter the locking unit when accessing the lock remotely

5.2 Denial of service

In order to test how susceptible the smart lock is to DoS attacks a methodology based on two previous studies on DoS attacks on IoT systems were followed [15] [16]. Kali linux was used to launch the attack on the device, specifically targeting the smart hub. The tools and commands used were: hping3, Nping and ping. The different commands used can as seen in table 5.1. The first command in table 5.1 were tested 3 times while sending different packet sizes. This was done by changing “-d <byte size>” to 600, 1200 and 1800 for respective byte size.

The following paragraph explains the meaning of the different flags for hping3 used. “–flood" means sending packets as fast as possible, disregarding responses from the target. “-c 10000" states that 10000 packets should be sent. “-S" means only to send SYN packets. “–rand-source" is to send from random

Command nr	Command
1	hping3 –c 10000 –d <byte size> –S –w 64 -–flood —rand-source 192.168.53.120
2	hping3 –S –P –U -–flood –V -–rand-source 192.168.53.120
3	nping –tcp-connect –rate=90000 –c 9000000 –q 192.168.53.120
4	hping3 -SARFU -V -c 10000 -d 100 -w 64 –flood –rand-source 192.168.53.120

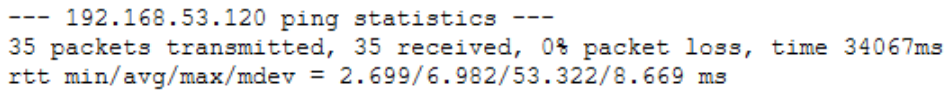
Table 5.1: Table showing the different commands tested.

IP address. “-w 64" specifies the TCP window size. “-P" and “-U" marks the packets with PUSH and URG respectively. Finally, “-SARFU" send packets with different protocols.

Explanation for nping is given in the following paragraph. “-rate=90000" specifies the rate of attacking. “-tcp-connect" means unprivileged TCP connection probe mode. “-c 9000000" specifies that the attack stops after 9000000 rounds. “-q" means to decrease verbose level by one.

In order to check how the connection of the smart hub were affected by the DoS attack the average response time as well as the time for success of attack, i.e the time for attack when first packet is lost were tested with the ping command. For all the tests exactly 100 packets were pinged by issuing “ping -n 100". The app was used to see if the lock was operable during the attack.

As a reference point the ping command were issued before the DoS attack to test the connection between the Kali linux and the smart hub as can be seen in figure 5.2. The average response time is relatively low and the packet lose rate is 0%.



```
--- 192.168.53.120 ping statistics ---
35 packets transmitted, 35 received, 0% packet loss, time 34067ms
rtt min/avg/max/mdev = 2.699/6.982/53.322/8.669 ms
```

Figure 5.2: Initial ping command showing the connection between the Kali linux machine and the smart hub.

5.3 Man in the middle attack

The tools needed for this attack is a rooted/jailbreaked phone, mitmproxy, Burp suite and Wireshark. The methodology followed is mainly based on the mitmproxy website[1], the burp suite website[2], the NSE website[3] and a blog post for configuration of burp suite with a android phone[4].

All traffic that goes through a router was sniffed with the command supplied by the NSE lab website “ssh root@routers.ip tcpdump -U -w - -i br0 not port 22 | wireshark -k -i -". Where “routers.ip" is replaced with the IP address of the router.

Proxy settings needs to be configured for both mitmproxy and burp suite on the phone where the app is installed. This includes the Proxy host name and the Proxy port. The proxy port can be set as any currently available port and the name would be the IP address of the proxy machine. In this case the proxy machine would be the Kali Linux and the IP can be retrieved by simply running “ip a".

For mitmproxy and burp suite to work a system certificate needs to be installed on the phone and the device needs to be rooted/jailbreaked into. In Burp suite this was accomplished by first exporting a certificate in DER format. Secondly running the commands shown in figure 5.3. Finally the certificate was moved to the android device and a application called Root certificate manager was used to install it.

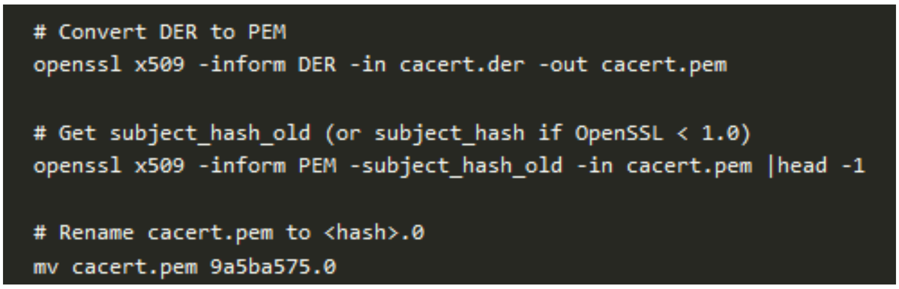
The process for mitmproxy was done in a similar way. The certificate was downloaded from mitm.it. “openssl x509 -inform PEM -subject_hash_old -in mitmproxy-ca-cert.cer | head -1" generates a hash of the certificate. The certificate is then copied to the generated hashcode with “cp mitmproxy-ca-cert.cer <hashcode>". The system certificate was installed on the android device in

[1] Mitmproxy. "Introduction". https://docs.mitmproxy.org/stable/ Last accessed: 15May2021. 2018

[2] PortSwigger. "Configuring an Android Device to Work With Burp". https://portswigger.net/support/configuring-an-android-device-to-work-with-burp Last accessed: 04May2021. 2015

[3] NSE Lab. "Wi-Fi Router MITM". https://nse.digital/pages/guides/Wireless/wifi-mitm.html Last accessed: 22Mar2021. 2021

[4] "Configuring Burp Suite With Android Nougat". https://blog.ropnop.com/configuring-burp-suite-with-android-nougat/ Last accessed: 05May2021. 2018



```
# Convert DER to PEM
openssl x509 -inform DER -in cacert.der -out cacert.pem

# Get subject_hash_old (or subject_hash if OpenSSL < 1.0)
openssl x509 -inform PEM -subject_hash_old -in cacert.pem |head -1

# Rename cacert.pem to <hash>.0
mv cacert.pem 9a5ba575.0
```

Figure 5.3: Converting the certificate from Burp Suite into a system certificate.

the same way as shown previously.

The traffic sent between the app and the smart lock was then sniffed with Burp suite or by running mitmproxy in a terminal.

5.4 Reverse engineering with MobSF

Performing static code analysis of an mobile application can be quite time consuming and difficult since they are usually obfuscated. Therefore MobSF which is an automated malware and security assessment tool was used[5]. MobSF was chosen over other analysis tools since there exists indication of it being more effective [29].

5.5 Handshake key leakage attack

Since number six on OWASP top ten vulnerabilities for IoT systems is “Insufficient privacy protection", a handshake key leakage attack was used to see what and how sensitive information were stored. The methodology follows from a security study on the August smart lock [23]. Since we need to access the xml files stored secretly a rooted/jailbreaked phone is needed. Afterwards the system files where found.

[5] Ajin Abraham, Magaofei, Matan Dobrushin and Vincent Nadal. "Mobile Security Framework (MobSF)" https://github.com/MobSF/Mobile-Security-Framework-MobSF Last accessed: 05May2021. 2021

5.6 Personal Information Leakage Attack

This attacks is executed exactly like the previous except the relevant system files where found under a different file path.

5.7 Bruteforcing passwords and looking at password resets

As seen in the previous work section the threat of brute force attacks was found on smart locks similar to the one investigated. Due to this it is reasonable to make sure that this risk is not also present in the specific locking unit investigated in this thesis. With the steps of the brute force attack as described below following the steps of the successful attempt in the research done by Arvid Viderberg [25].

Before a bruteforce attack is made various aspects of the access to the account is first explored. To allow for decisions to be made regarding what options are available and what vulnerabilities could potentially be exploited during the actual bruteforce attack.

With the limitations of the password being the first point to explore. Which would include testing the limitations and restrictions provided when creating or changing the password for an account. With the specific questions to look into being as follows:

- Is there a minimum/maximum amount of characters allowed?
- Are different character types required?
  - Uppercase letters
  - Lowercase letters
  - Numbers
  - Special character
- Are common password prohibited or discouraged?

With some additional points to explore being if multi factor authentication is available or required for accessing an account and if failed attempts to sign into an account signs out the actual user of the account or otherwise prevents the user from accessing it.

Secondly there is also a need to explore the potential of information being provided without proper access to an account. Such as username enumeration were a failed attempt to sign into an account provides information about the potential existence for the account the attempt is made towards. Similarly, the potential to gain information about existing accounts when an attempt to create an account with already used credentials is made should also be explored.

The last point to explore would the password reset feature. Were the specific questions to look into would be as follows:

- Can multiple password resets be requested?
- If multiple password resets can be made, how many are valid at one time?
- For how long are password reset requests valid?
- Are any additional information about the account provided with the password reset?

5.8 State consistency

In the same research as in the above section it was also found that both of the locks was vulnerable to state consistency attacks. With the lock being possible to access when access had been manually revoked by the owner of the lock. As such testing the locking unit that is investigated in this thesis with equivalent steps [25] would be reasonable. To make sure that the vulnerability is not something present within this specific locking unit.

To attempt a state consistency attack two mobile devices with the app for accessing the locking unit installed is required. With one device being set as the owner of the lock and the other being set as a guest with limited access.

After the guest has made sure that they are able to unlock and lock the door they then disable their network connection in an attempt to prevent their access being revoked, leaving only the Bluetooth connection to the lock active. The owner then revokes the guest users' access to the lock after which the guest tries to unlock and lock the door once more whilst keeping only the Bluetooth connection active.

If the guest is still able to access the lock although their access has been revoked by the owner the state consistency attack has been a success.

Chapter 6

Results

6.1 SecuriCAD

The threats that were found during the simulation process, using the model provided in the method are as follows:

Applications

The main threat documented with the access point being the application is that any data sent out or received by the application is at risk for being read and deleted with the services and access given by the application also being at risk for being denied.

With the data stored within the application mainly being at risk of having the access to it denied and attempts at access to the data being possible by the attacker.

Systems

With the system of either application as the access point for the attack the main threat documented is that the application hosted on the system is at risk of being read, denied and modified as well as the network connection held by the application being accessible. Furthermore information about the owner of the system is at risk with the owner being possible to assume.

Additionally with the access point being either the smartlock of wi-fi bridge the data sent from or received by the application hosted on the system is at risk at being denied, deleted, accessed and read. With the data stored on the applications only being at risk of having the access to it denied. Lastly with access to the wi-fi bridge specifically there is also the threat of the connection and network connected to the system being denied or having the network forwarded. Both a DoS and MITM attack are also a possible risk at an about 50% of being successfully.

Connection rules and network

With access to either of the various network and device connections DoS was found to be the main threat to affect not only the point of access for the attacker but also the connected network or connection depending on the access point.

Furthermore the data sent over the network was found to be at risk for deletion, reading and eavesdropping. Whilst data sent between the different applications or stored on connected applications is only at 50% risk of being successfully read or eavesdropped.

Data

Only access through the data in transit either over the network or device connection was found to produce a risk. With the documented threat being the data either being read or deleted.

6.2 Penetration testing

6.2.1 Preparation

In order to perform some of the different attacks described in the methodology section the IP addresses of the relevant devices are required. To accomplish this the commands nmap and ip was used in combination with unplugging and plugging in the smart hub. The results can be seen in table 6.1.

Device	IP address	Role
Kali Linux	192.168.53.145	Attacker
Smart hub	192.178.53.120	Victim
Router	192.168.53.1	Local network gateway

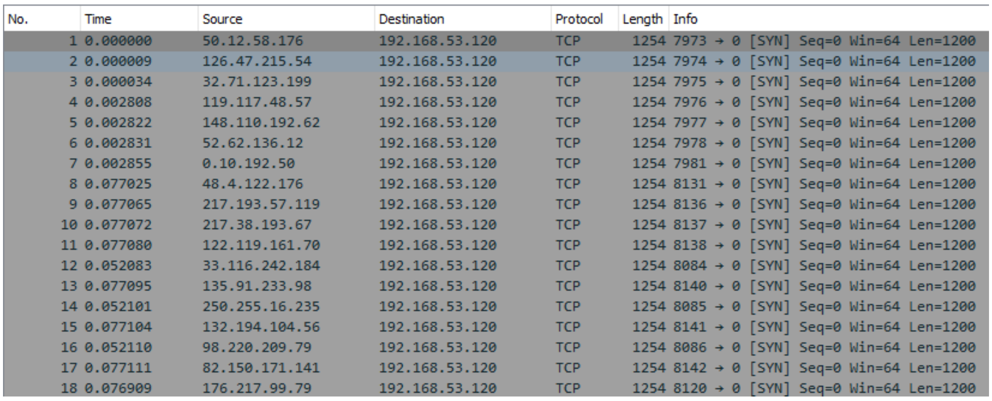
Table 6.1: Table of the different IP addresses and the role of the devices.

6.2.2 Denial of service

All commands mentioned earlier from the methodology section tried were executed successfully. Figure 6.1 and 6.2 displays how hping3 sent from random IP addresses to the target and how different protocols are being sent from the “-SARFU" flag respectively. Most of the commands resulted in high package lose rate, with higher being better. Some of the attacks resulted in the app not being able to communicate with the smart hub over a internet connection. Meaning the lock could not be locked/unlocked except while using Bluetooth.

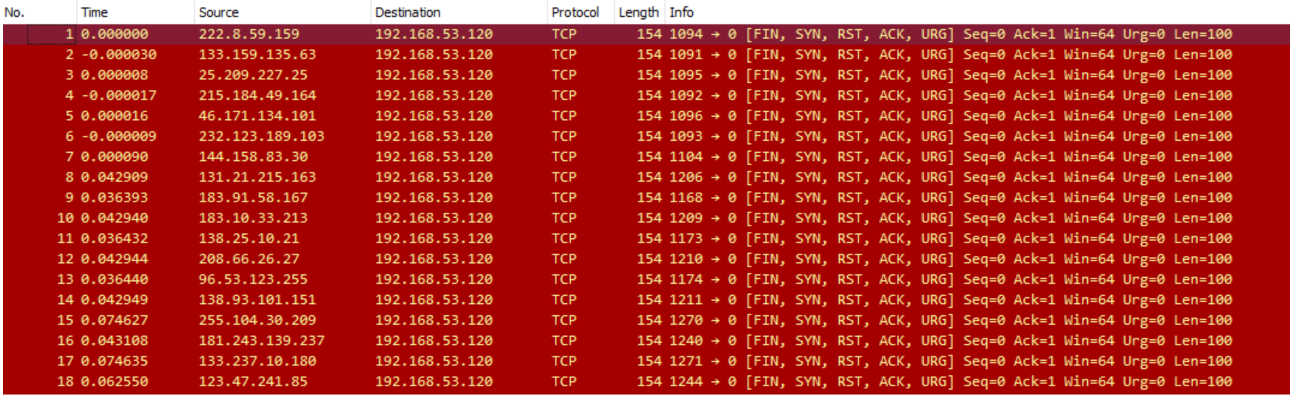
Most of the attacks caused the connection to be slower than usual. At the same time unlocking the device either through Bluetooth or manually always worked regardless of attack attempted. The lock being operable even though under attack is expected since the DoS attacks are actually targeting the smart hub and not the smart lock itself.

As seen in table 6.2 some of the commands made the app not function properly, thus making the lock uncontrollable from a distance. Both the simple syn command and the 1800 bytes one had a really high package lose rate, with the 1800 bytes one being 100%. This could explain why the lock was inoperable as essentially close to no response from the smart hub was given. The TCP command also yielded a high package lose rate. However the app still worked which could suggest a package lose of nearly 100% is needed to fully mess with the connection. As previous research has indicated a high package size is more likely to be successful. Which we believe gains further credibility based on our results.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	50.12.58.176	192.168.53.120	TCP	1254	7973 → 0 [SYN] Seq=0 Win=64 Len=1200
2	0.000009	126.47.215.54	192.168.53.120	TCP	1254	7974 → 0 [SYN] Seq=0 Win=64 Len=1200
3	0.000034	32.71.123.199	192.168.53.120	TCP	1254	7975 → 0 [SYN] Seq=0 Win=64 Len=1200
4	0.002808	119.117.48.57	192.168.53.120	TCP	1254	7976 → 0 [SYN] Seq=0 Win=64 Len=1200
5	0.002822	148.110.192.62	192.168.53.120	TCP	1254	7977 → 0 [SYN] Seq=0 Win=64 Len=1200
6	0.002831	52.62.136.12	192.168.53.120	TCP	1254	7978 → 0 [SYN] Seq=0 Win=64 Len=1200
7	0.002855	0.10.192.50	192.168.53.120	TCP	1254	7981 → 0 [SYN] Seq=0 Win=64 Len=1200
8	0.077025	48.4.122.176	192.168.53.120	TCP	1254	8131 → 0 [SYN] Seq=0 Win=64 Len=1200
9	0.077065	217.193.57.119	192.168.53.120	TCP	1254	8136 → 0 [SYN] Seq=0 Win=64 Len=1200
10	0.077072	217.38.193.67	192.168.53.120	TCP	1254	8137 → 0 [SYN] Seq=0 Win=64 Len=1200
11	0.077080	122.119.161.70	192.168.53.120	TCP	1254	8138 → 0 [SYN] Seq=0 Win=64 Len=1200
12	0.052083	33.116.242.184	192.168.53.120	TCP	1254	8084 → 0 [SYN] Seq=0 Win=64 Len=1200
13	0.077095	135.91.233.98	192.168.53.120	TCP	1254	8140 → 0 [SYN] Seq=0 Win=64 Len=1200
14	0.052101	250.255.16.235	192.168.53.120	TCP	1254	8085 → 0 [SYN] Seq=0 Win=64 Len=1200
15	0.077104	132.194.104.56	192.168.53.120	TCP	1254	8141 → 0 [SYN] Seq=0 Win=64 Len=1200
16	0.052110	98.220.209.79	192.168.53.120	TCP	1254	8086 → 0 [SYN] Seq=0 Win=64 Len=1200
17	0.077111	82.150.171.141	192.168.53.120	TCP	1254	8142 → 0 [SYN] Seq=0 Win=64 Len=1200
18	0.076909	176.217.99.79	192.168.53.120	TCP	1254	8120 → 0 [SYN] Seq=0 Win=64 Len=1200

Figure 6.1: A screenshot from wireshark showing how the command successfully send from different IP addresses



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	222.8.59.159	192.168.53.120	TCP	154	1094 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
2	-0.000030	133.159.135.63	192.168.53.120	TCP	154	1091 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
3	0.000008	25.209.227.25	192.168.53.120	TCP	154	1095 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
4	-0.000017	215.184.49.164	192.168.53.120	TCP	154	1092 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
5	0.000016	46.171.134.101	192.168.53.120	TCP	154	1096 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
6	-0.000009	232.123.189.103	192.168.53.120	TCP	154	1093 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
7	0.000090	144.158.83.30	192.168.53.120	TCP	154	1104 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
8	0.042909	131.21.215.163	192.168.53.120	TCP	154	1206 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
9	0.036393	183.91.58.167	192.168.53.120	TCP	154	1168 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
10	0.042940	183.10.33.213	192.168.53.120	TCP	154	1209 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
11	0.036432	138.25.10.21	192.168.53.120	TCP	154	1173 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
12	0.042944	208.66.26.27	192.168.53.120	TCP	154	1210 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
13	0.036440	96.53.123.255	192.168.53.120	TCP	154	1174 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
14	0.042949	138.93.101.151	192.168.53.120	TCP	154	1211 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
15	0.074627	255.104.30.209	192.168.53.120	TCP	154	1270 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
16	0.043108	181.243.139.237	192.168.53.120	TCP	154	1240 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
17	0.074635	133.237.10.180	192.168.53.120	TCP	154	1271 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100
18	0.062550	123.47.241.85	192.168.53.120	TCP	154	1244 → 0 [FIN, SYN, RST, ACK, URG] Seq=0 Ack=1 Win=64 Urg=0 Len=100

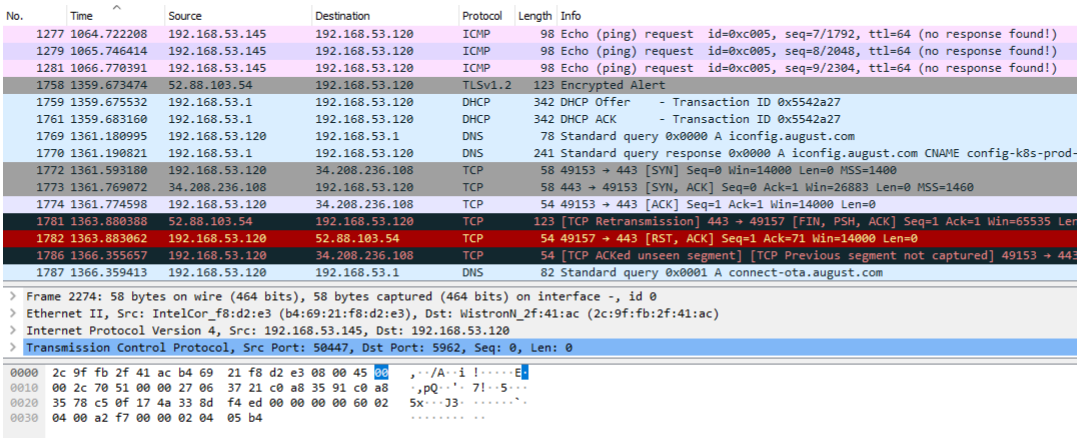
Figure 6.2: A screenshot from wireshark showing how the command successfully send from different protocols

6.2.3 Man in the middle attack

The traffic between the router and the targeted device was sniffed as illustrated by figure 6.3. All the traffic was encrypted which meant no sensitive information like usernames or passwords could be sniffed.

Attack method	Does the app work	Time for success of attack	Package lose rate	Average response time
600 bytes	yes	0.94s	92%	534ms
1200 bytes	yes	0s	94 %	763ms
1800 bytes	no	0s	100%	–
Simple Syn	no	19s	99 %	19193ms
TCP	yes	70s	97 %	17456ms
TCP connect flood	yes	–	0 %	428ms

Table 6.2: Table showing the different attack methods, if the app was operable and the communication information between the systems.



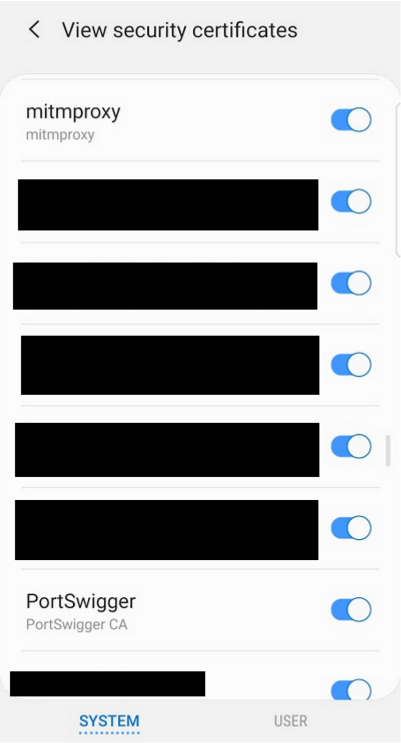
```
No.       Time          Source            Destination       Protocol  Length Info
     1277 1064.722208   192.168.53.145    192.168.53.120    ICMP          98 Echo (ping) request  id=0xc005, seq=7/1792, ttl=64 (no response found!)
     1279 1065.746414   192.168.53.145    192.168.53.120    ICMP          98 Echo (ping) request  id=0xc005, seq=8/2048, ttl=64 (no response found!)
     1281 1066.770391   192.168.53.145    192.168.53.120    ICMP          98 Echo (ping) request  id=0xc005, seq=9/2304, ttl=64 (no response found!)
     1758 1359.673474   52.88.103.54      192.168.53.120    TLSv1.2      123 Encrypted Alert
     1759 1359.675532   192.168.53.1      192.168.53.120    DHCP         342 DHCP Offer    - Transaction ID 0x5542a27
     1761 1359.683160   192.168.53.1      192.168.53.120    DHCP         342 DHCP ACK      - Transaction ID 0x5542a27
     1769 1361.180995   192.168.53.120    192.168.53.1      DNS           78 Standard query 0x0000 A iconfig.august.com
     1770 1361.190821   192.168.53.1      192.168.53.120    DNS          241 Standard query response 0x0000 A iconfig.august.com CNAME config-k8s-prod-
     1772 1361.593180   192.168.53.120    34.208.236.108    TCP           58 49153 → 443 [SYN] Seq=0 Win=14000 Len=0 MSS=1400
     1773 1361.769072   34.208.236.108    192.168.53.120    TCP           58 443 → 49153 [SYN, ACK] Seq=0 Ack=1 Win=26883 Len=0 MSS=1460
     1774 1361.774598   192.168.53.120    34.208.236.108    TCP           54 49153 → 443 [ACK] Seq=1 Ack=1 Win=14000 Len=0
     1781 1363.880388   52.88.103.54      192.168.53.120    TCP          123 [TCP Retransmission] 443 → 49157 [FIN, PSH, ACK] Seq=1 Ack=1 Win=65535 Len
     1782 1363.883062   192.168.53.120    52.88.103.54      TCP           54 49157 → 443 [RST, ACK] Seq=1 Ack=71 Win=14000 Len=0
     1786 1366.355657   192.168.53.120    34.208.236.108    TCP           54 [TCP ACKed unseen segment] [TCP Previous segment not captured] 49153 → 443
     1787 1366.359413   192.168.53.120    192.168.53.1      DNS           82 Standard query 0x0001 A connect-ota.august.com

> Frame 2274: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface -, id 0
> Ethernet II, Src: IntelCor_f8:d2:e3 (b4:69:21:f8:d2:e3), Dst: WistronN_2f:41:ac (2c:9f:fb:2f:41:ac)
> Internet Protocol Version 4, Src: 192.168.53.145, Dst: 192.168.53.120
> Transmission Control Protocol, Src Port: 50447, Dst Port: 5962, Seq: 0, Len: 0

0000  2c 9f fb 2f 41 ac b4 69  21 f8 d2 e3 08 00 45 00   ,··/A··i !·····E·
0010  00 2c 70 51 00 00 27 06  37 21 c0 a8 35 91 c0 a8   ·,pQ··'· 7!··5···
0020  35 78 c5 0f 17 4a 33 8d  f4 ed 00 00 00 00 60 02   5x···J3· ······`·
0030  04 00 a2 f7 00 00 02 04  05 b4                     ········ ··
```

Figure 6.3: Wireshark traffic captued between the router and the smart lock.

The certificates was installed successfully as can be seen in figure 6.4 and using mitmproxy traffic could be sniffed. However since the accompanied app uses http2 requests the following error message was given; “Initiating HTTP/2 connections with prior knowledge are currently not supported.". Which caused the connection to proxy to disconnect and thus the app not to work. The reason being that mitmproxy does not support changing the request destination with http2.

This effectively means that it was not possible to sniff the traffic coming from the app using mitmproxy. However support for this feature may be implemented in later implementations of mitmproxy. If it does get implemented redoing this experiment could be relevant as previous works has shown that user information such as password were able to sniffed in plain text.

Similar results were found when using burp suite. No error message was given however the lock could not be controlled through the app. Burp suite has recently implemented http2 support although our results may suggests it might need further improvements.



View security certificates

mitmproxy
mitmproxy

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

PortSwigger
PortSwigger CA

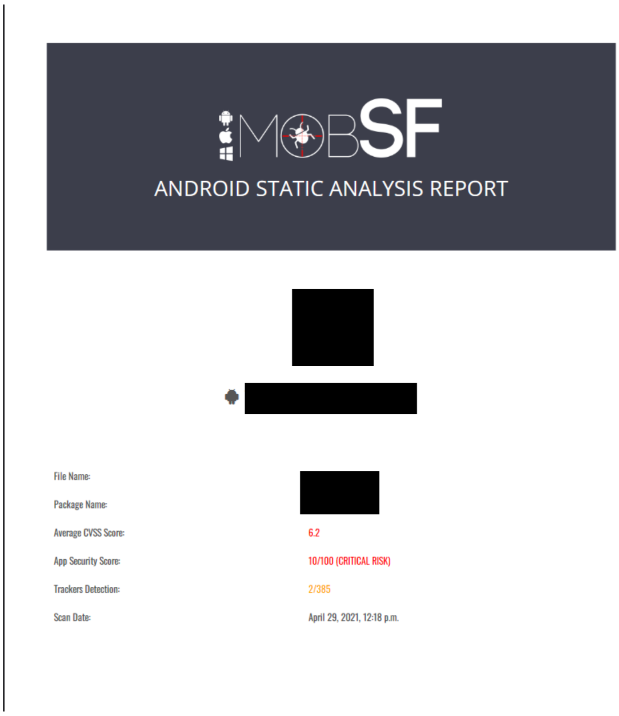
[redacted]

Figure 6.4: System certificate for both Mitmproxy and Burp suite installed. All other certificates are crossed out.

6.2.4 Reverse engineering with MobSF

According to MobSF the overall security of the app is quite poor as it was given the lowest score possible of 10/100 as seen figure 6.5. A lot of different potential security issues were found. However it is outside the scope of this thesis to analyse all of them. One issue found by MobSF might by confirmed which is that files may contain hardcoded sensitive information. As the handshake key and some personal information was found in the coming sections.

Figure 6.6 displays aforementioned issue.



MobSF

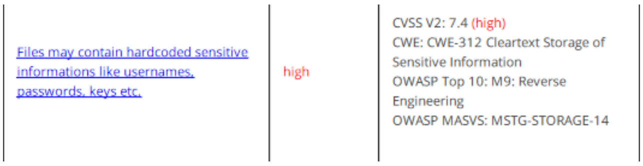
ANDROID STATIC ANALYSIS REPORT

[redacted]

[redacted]

File Name:	[redacted]
Package Name:	[redacted]
Average CVSS Score:	6.2
App Security Score:	10/100 (CRITICAL RISK)
Trackers Detection:	2/385
Scan Date:	April 29, 2021, 12:18 p.m.

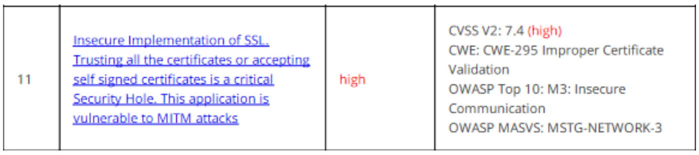
Figure 6.5: Screenshot displaying the front page from MobSF.



Files may contain hardcoded sensitive informations like usernames, passwords, keys etc.	high	CVSS V2: 7.4 (high) CWE: CWE-312 Cleartext Storage of Sensitive Information OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14

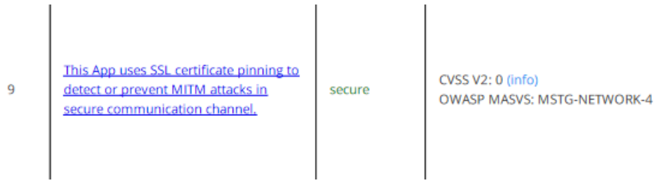
Figure 6.6: Screenshot displaying how sensitive information may be stored in files.

Another concern raised is how the app is vulnerable to MITM attacks as illustrated by figure 6.7. Although this could not be confirmed as the MITM attacks in this report ultimately was unsuccessful. MobSF also seemingly contradictory claims the app is secure against MITM attacks. Figure 6.8 illustrates this. One possible explanation why is that the app contains libraries not used. Therefore the weakness may or may not lay in the application itself.



11	Insecure Implementation of SSL. Trusting all the certificates or accepting self signed certificates is a critical Security Hole. This application is vulnerable to MITM attacks	high	CVSS V2: 7.4 (high) CWE: CWE-295 Improper Certificate Validation OWASP Top 10: M3: Insecure Communication OWASP MASVS: MSTG-NETWORK-3

Figure 6.7: MobSF report claiming the app is vulnerable to MITM attacks.

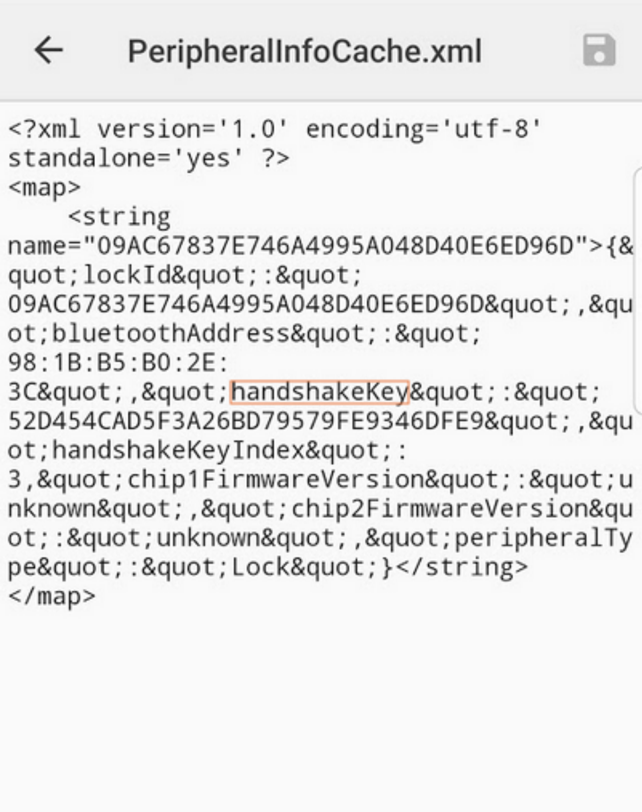


9	This App uses SSL certificate pinning to detect or prevent MITM attacks in secure communication channel.	secure	CVSS V2: 0 (info) OWASP MASVS: MSTG-NETWORK-4

Figure 6.8: MobSF report claiming the app is secure to MITM attacks.

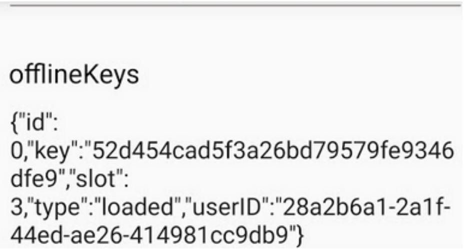
6.2.5 Handshake key leakage attack

As can been seen from figure 6.9 the handshake key is stored without any encryption. The handshake key could also be located in the database folder as seen in figure 6.10. Previous work has shown that with the handshake key one can control the lock without using the official app. This however was not verified due to time constraints.



```
<?xml version='1.0' encoding='utf-8'
standalone='yes' ?>
<map>
    <string
name="09AC67837E746A4995A048D40E6ED96D">{&
quot;lockId&quot;:&quot;
09AC67837E746A4995A048D40E6ED96D&quot;,&qu
ot;bluetoothAddress&quot;:&quot;
98:1B:B5:B0:2E:
3C&quot;,&quot;handshakeKey&quot;:&quot;
52D454CAD5F3A26BD79579FE9346DFE9&quot;,&qu
ot;handshakeKeyIndex&quot;:
3,&quot;chip1FirmwareVersion&quot;:&quot;u
nknown&quot;,&quot;chip2FirmwareVersion&qu
ot;:&quot;unknown&quot;,&quot;peripheralTy
pe&quot;:&quot;Lock&quot;}</string>
</map>
```

Figure 6.9: Screenshot displaying how the handshake key is stored in plain text



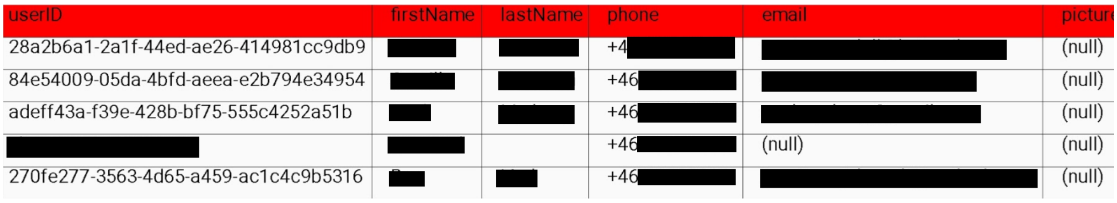
offlineKeys

```
{"id":
0,"key":"52d454cad5f3a26bd79579fe9346
dfe9","slot":
3,"type":"loaded","userID":"28a2b6a1-2a1f-
44ed-ae26-414981cc9db9"}
```

Figure 6.10: Screenshot displaying how the handshake key is stored in plain text

6.2.6 Personal information leakage attack

Figure 6.11 shows how some personal information such as name, phone number and email address was found. This may create a vulnerability to brute-force attacks against the password since both the phone number and the email address are used as usernames when logging in.



userID	firstName	lastName	phone	email	pictur
28a2b6a1-2a1f-44ed-ae26-414981cc9db9	[redacted]	[redacted]	+4[redacted]	[redacted]	(null)
84e54009-05da-4bfd-aeea-e2b794e34954	[redacted]	[redacted]	+46[redacted]	[redacted]	(null)
adeff43a-f39e-428b-bf75-555c4252a51b	[redacted]	[redacted]	+46[redacted]	[redacted]	(null)
[redacted]	[redacted]		+46[redacted]	(null)	(null)
270fe277-3563-4d65-a459-ac1c4c9b5316	[redacted]	[redacted]	+46[redacted]	[redacted]	(null)

Figure 6.11: The found personal info including UserID, full name, phone number and email address for each user of the smartlock. Sensitive information is crossed out.

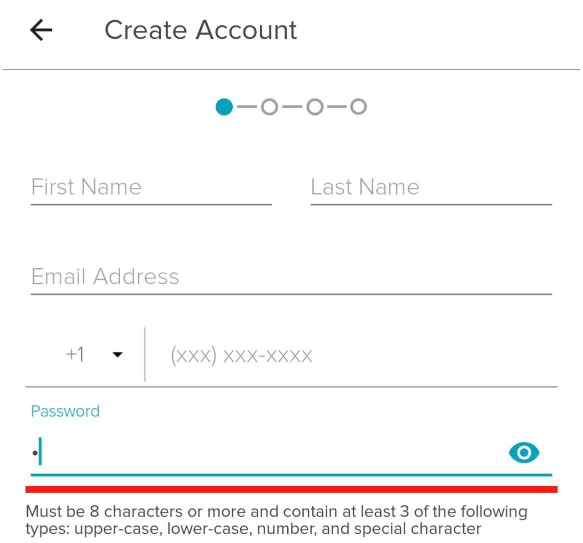
6.2.7 Bruteforcing

Creating an account

When creating an account both a valid phone number and email address is required both of which will be sent a verification code that is required to verify the account in the final step of the creation process. The Password set for the account requires at least eight character out of which three different types of characters has to be included as can be seen in figure 6.12. With the different character types being lower-case, upper-case, numbers and special characters. There does not seem to be any limitation on the most amount of characters permitted when creating a password with over 200 characters being allowed when setting the password for the account.

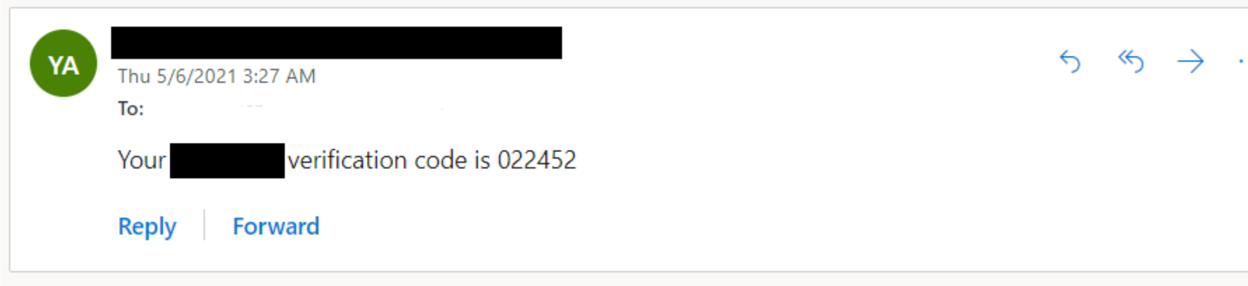
When finalising the account creating a verification code is sent firstly to the phone number and then email associated with the account. With the verification code consisting of six digits. During the setup process the verification code sent to the phone briefly flashes on screen, something that seems to come from the fact that the verification code sent through the phone can be automatically verified if you attempt to login on the same device the message was sent to. The same did not occur during the email verification even though the email is connected to the same phone the app is used through.

All of the verification messages sent either to the phone number or email during account creation, signing in or when forgetting the password are all very simplistic with no additional information to be acquired from them. Neither the text message nor email give any additional detail on what action is being taken to require the verification. With an example of these simple verification messages being seen in figure 6.13.



Must be 8 characters or more and contain at least 3 of the following types: upper-case, lower-case, number, and special character

Figure 6.12: A screenshot of the account creation page on the mobile application showing the requirements for the password



[redacted]

Thu 5/6/2021 3:27 AM

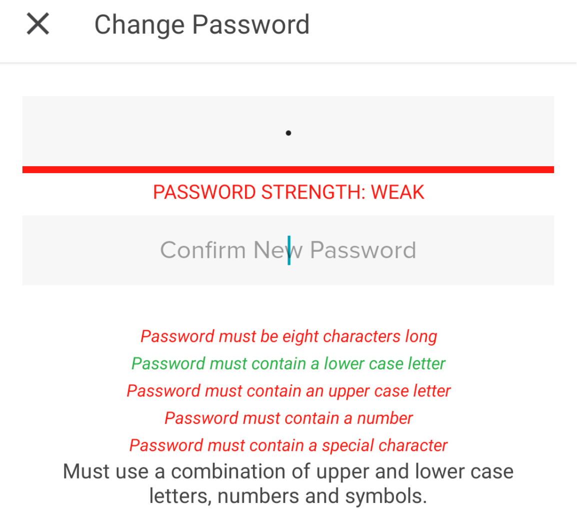
To:

Your [redacted] verification code is 022452

Figure 6.13: A screenshot of an example verification email that is sent out to verify during account creation, signing in or resetting the password.

Multi factor authentication is required when accessing the lock, both when signing in as well as creating an account and requesting a password reset. With the latter two requiring three factors of authentication, password, phone number and email. Whilst signing in only requires two factors of authentication, password and either a phone number or email.

When changing the password of an existing account the same requirements as when creating an account are present. Although the requirements as seen in figure 6.14 are presented in such a way that it is not made clear how many different special characters are required. The amount required remains the same with three different character types being required. Any attempt with less than eight characters or less than three different character types does not allow the user to confirm the password change.



Change Password

•

PASSWORD STRENGTH: WEAK

Confirm New Password

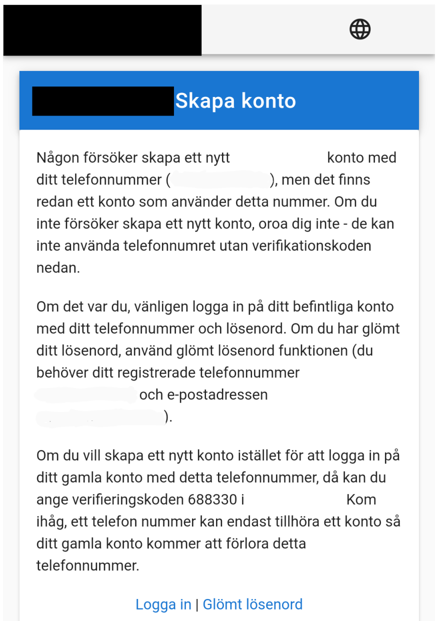
Password must be eight characters long
Password must contain a lower case letter
Password must contain an upper case letter
Password must contain a number
Password must contain a special character
Must use a combination of upper and lower case letters, numbers and symbols.

Figure 6.14: A screenshot of the listed requirements for changing the password of an existing account.

Attempting to create a new account with the same email or phone number as an already existing account does not yield any information about the account already existing. With the creation process seemingly proceeding as normal up until the verification stage. During which instead of the normal verification message the account owner instead gets sent a link to a message seen in figure 6.15 informing the owner of there being an attempt to create a new account using their credentials which are listed within the message. The message also includes information on what to do and if it is an legitimate attempt to create a new account by the owner of the account the verification code needed to finalise this creation is also given.

Signing in

When signing in either the email or phone number associated with the account is required alongside the password to be able to sign in. With a verification code being sent through the chosen option to verify the user signing in. When attempting to sign in using a valid email or phone number that does not yet have an account associated with them an error message is presented that reads



[redacted]

[redacted] Skapa konto

Någon försöker skapa ett nytt konto med ditt telefonnummer ( ), men det finns redan ett konto som använder detta nummer. Om du inte försöker skapa ett nytt konto, oroa dig inte - de kan inte använda telefonnumret utan verifikationskoden nedan.

Om det var du, vänligen logga in på ditt befintliga konto med ditt telefonnummer och lösenord. Om du har glömt ditt lösenord, använd glömt lösenord funktionen (du behöver ditt registrerade telefonnummer och e-postadressen).

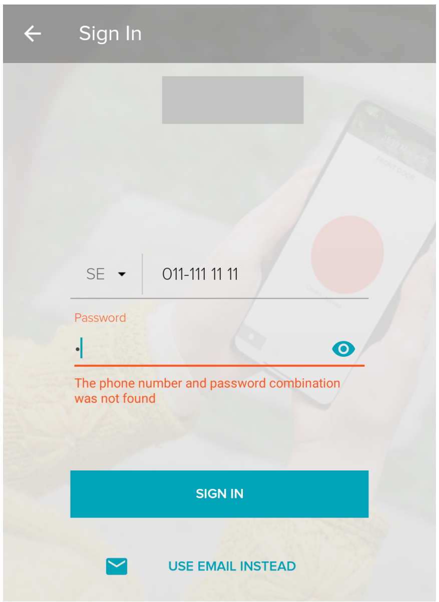
Om du vill skapa ett nytt konto istället för att logga in på ditt gamla konto med detta telefonnummer, då kan du ange verifieringskoden 688330 i Kom ihåg, ett telefon nummer kan endast tillhöra ett konto så ditt gamla konto kommer att förlora detta telefonnummer.

Figure 6.15: A screenshot of the linked message sent to the account holder when attempting to create a new account with the email or phone number of the already existing account.

“The email/phone number and password combination was not found" as seen in figure 6.16. The same message is presented if the incorrect password is given to an existing account. Something that does not allow anyone to neither confirm nor deny the existence of an account.

Multiple attempts to sign in was done both with an incorrect email and phone number as well as the correct ones for an existing account. To test whether the app blocks repeated failed attempts to sign into an account more than 25 attempts was done during a short period of time for each of the instances mentioned above. For all instances no attempts were blocked by the application, allowing for the potential to brute force the password without additional delay being added.

When signing into an existing account using the associated phone number the verification code sent through a text message is automatically verified. Something that was mentioned during the account creation earlier in this section.



SE 011-111 11 11

The phone number and password combination was not found

Figure 6.16: A screenshot of the error message given when attempting to sign in using the incorrect credentials or password for any existing account. With the same message being given for both situations.

Password resets

When requesting a password reset both the correct email and phone number associated with the account is required to be allowed to set a new password for the account. The verification process is done in two steps with the option to chose between first inputting and verifying either the phone number or email and then the other. With the verification message being sent first to the option first chosen and only after the first option is verified does it move on to verifying the second option.

The verification message sent during a password reset are the same as in previous action as seen earlier in the section. No further information is given in these messages about the verification being given for an attempted password reset.

When attempting to verify either the email or the phone number multiple verification messages can be requested. At most seven seemed to be able to come through at a time with any further requests not reaching the destination. With each new request the previous verification code was made invalid with the exception being for one instance during the multiple attempts that allows for the second most recent verification code to be valid at the same time as the most recent requested verification code was. However in the general case only one code was ever valid at one time.

In certain instances the most recent verification code was made invalid although it theoretically should have been valid. Something that was the case in most of the instances were multiple requests was made after the most recently acquired verification code. With the latter attempted requests never arriving at the destination and thus most likely the most recently arrived verification code would no longer be the correct code requested by the application for the verification process. When the verification code was made invalid after 15 minutes.

6.2.8 State consistency

Both the owner and guest account was set up with the ability of the guest to lock and unlock the door using both wi-fi and Bluetooth was confirmed. The graphical password of the phone was required for the guest to operate the lock.

With the network disabled on the device of the guest user their access was revoked. Multiple attempts to lock and unlock the door after this step was successful. Although updating the page in any way, through either switching between the tabs of the app or closing down the app temporarily resulted in an authorisation failure. Making the app no longer being able to connect to the locking unit.

Enabling the network connection in this state caused the app to pull the user to a welcome page, asking the user to set up a new device.

When revoking owner privileges from an account whilst the user is in the app, the now guest user was still able to access all the tabs that the owner user has access to. However only the history tab was possible to be viewed as the other tabs simply resulted in a blank page with an error message stating something went wrong

When the guest user was revoked access and then allowed access back as a guest once again the app showed only the 2 tabs that were to be present for a guest user, the lock tab to allow the user to lock and unlock the door and the account tab that for a guest user shows only the information of the user's own account rather than a list of all accounts with access to the lock.

Chapter 7

Discussion

7.1 Denial of service

A DoS attack as presented here might be used by a hacker in a exploitative way for the owner. Such a scenario could be when delivery personal is let in by the owner through a remote connection when the owner themselves are not at home. If the attacker performs the DoS attacks once the lock is unlocked there is no way for the delivery personal nor the owner to lock the door. Of course assuming the delivery personal does not have a key, which they are unlikely to have. Unless whoever entered the house has the time to wait for the owner to come back there now exists a time frame between when they leave and the owner comes home for the attacker to access whatever is inside. Additionally the owner may mistake the DoS attack for a software bug and may thus not be aware that something is wrong.

Having said that there are difficulties with the attack and there already exists countermeasure features in the smart lock. Auto locking means the door always is locked when closed, effectively neutralising the proposed exploit. Meaning the attack as proposed can only be executed when the auto locking feature is turned off, which is an option in the smart lock device.

Another problem is the need for the network password as access to the network is needed to perform the DoS attack done in this thesis. However not all network has password. As shown in a previous study some IoT devices sent network secrets like password in plain text [18]. That is to say in a smart home other devices vulnerabilities may come to be security issues in others. Passwords can also be hacked through brute force attacks.

Previous work has demonstrated a DoS attack as an example of the insecurity present in smart locks [17]. We believe the DoS attacks as performed within this report shows vulnerability like previous work also has concluded. Since the smart lock device uses a smart hub however this vulnerability is lessened. At least to DoS attacks over Wi-fi. Since such an attack does not target the device itself and thus the system itself is not under attack.

7.2 Man in the middle

Comparing our results to previously found weaknesses the unit does seems secure to MITM attacks. Since none of the attacks performed found any real vulnerabilities. This could be explained by not properly executing said attacks. We believe http2 to be the protocol who prevented success. There likely exists workarounds to this problem and thus it is hard to conclude any rigid conclusion based on our methodology and results.

7.3 Reverse engineering with MobSF

Sensitive information may be hardcoded in files according to the report from MobSF which in this case turned out to be true since both the handshake key and personal information was found. It might therefore be worth looking at the other vulnerabilities found within the report. The potential vulnerability to MITM attacks being a relevant candidate.

In order to find more conclusive results performing a manual static code analysis or a dynamic one might have been a good idea. Using tools such as Jadx and APKTool. However as stated before that process can become quite time consuming and as such is not necessarily the best approach depending on the time scope.

7.4 Handshake key leakage

The handshake key could be found in plain text in the system files and has shown to previously allow unofficial communication with a smart lock unit [23]. We believe this is a relatively serious vulnerability since the exploits can be quite devastating. Although there also exists some drawbacks to the attack as tested in this thesis. To read the system files the device needs to be rooted/jailbroken. Furthermore the attacker needs access to the victims phone.

With that being said we believe the protection is lacking. Specifically sophisticated encryption should be implemented. As the study our methodology is based on suggests [23]. Additionally as the study mentions such an encryption system could take form as the secret handshake scheme proposed by D. Balfanz et.al [30].

7.5 Personal information leakage

As has been previously found there is a lack of encryption in system files [23]. With user information such userID, name, phone number and email being stored in plain text. Countermeasures would be similar to the handshake key leakage as a state-of-the-art encryption method would be advised to implement. Wang et.al has implemented an encryption design that is able to protect rooted system files and we believe something similar should be implemented [31].

7.6 Brute forcing

Compared to what had been seen in the mentioned previous research the passwords for the application tested in this thesis follows a majority of the password conventions that are recommended by OWASP and similar sources. With a minimum amount of characters and character types being required as well as multi factor authentication being required when signing into an account.

However a vulnerability that remains the same as previous research has found is that common passwords such as “Password1" are not discouraged. Which would make brute forcing the password using a dictionary of common passwords have the potential to be highly successful.

7.6.1 Signing in

Another vulnerability that is present in previous research that has carried over is the fact that passwords attempts are not restricted after a set number of failed attempts. Two factor authentication is however required when signing in which may be a reason for this. Since successfully brute forcing the password does not yield immediate access to the account. The lack of restriction does make it so that the actual user cannot get blocked out of their account and is a probable reason for the lack of restricted attempts. However since this could instead be solved by requiring both the phone number and email verification to sign into a blocked account it does not excuse the risk that this choice results in.

With the lack of restriction allowing for the brute forcing process to run uninterrupted. Which is a risk for users, with either other accounts connected to the same email or phone number being at risk if the user keeps the same password for multiple accounts. Additionally due to the potential of either the phone or email also being compromised this vulnerability is still a risk for the user. Especially with the potential to brute force the verification code that is limited to six digits. Something that along side the brute forced password would allow unauthorised access to the account.

7.6.2 Password reset

At most two codes were active at once even if that only occurred rarely with most cases only allowing the most recent code to be valid. Could be due to timing.

At times neither of the codes received were valid. Most likely due to how the amount of codes received being limited to around seven at a time. With it being possible that an new attempt was able to be confirmed without the code being received.

No information could be taken from the validation message with only the code being given and nothing else.

7.7 State consistency

The problem seen in previous research in terms of state consistency attacks has been partially fixed. With the access to the lock not being fully functional after following the steps mentioned in the method. With any page changes done within the app or having the app closed down briefly, resulting in an error message. However with this there is still the potential for a successful state consistency attack as long as the page and connection is left untouched.

The locking unit has a built in clock and calendar according to the home page of the locking unit. Something that seemingly was the cause for one of the locks mentioned in the previous study being able to prevent this kind of attack. Although this difference could be caused by the privilege being revoked manually. Something that for this application was more intuitive in terms of allowing temporary access to users.

Additionally testing this extended access when moving outside of the range of the Bluetooth connection could also be something that would yield a different result. Making this threat less likely to be used maliciously due to the limitations it would provide.

Chapter 8

Future work

As stated before “fail open" is essentially necessary for smart lock systems but it also presents a serious vulnerability to DoS attacks. A Bluetooth based DoS for example might be interesting to try in future works to see if lock would then be left in a unlocked state.

Since the http2 protocol seemingly prevented the MITM attack from working redoing this experiment if support for http2 in mitmproxy is implemented or http2 is further improved upon in Burp suite might be relevant. Also other workarounds might be worth trying.

Since previous work has controlled smart lock units with the discovered handshake key doing such an experiment could be worth trying. Retesting found vulnerabilities in other smart lock units might also be interesting to see if the devices becomes more secure with time.

In terms of further work required for the brute force attack. There is an interest in performing an actual attempt at password cracking. To test if there are any limitations put in place when using an automatic process that are able to make many more attempts in a single second than what a manual process could. With this being timed to determine how viable the attack is in an actual instance or if the service gets affected in any way from the multiple attempts made towards it.

Additionally there is also an interest in attempting a brute force attack similar to the above on the verification code. To test the vulnerability of the relatively small series of digits as well if the combination of the two is possible to allow full access to the account. Which would be a greater risk for the user of the account.

Lastly in terms of the state consistency attack further research could be done towards testing all the limitations of the state of extended access that was achieved. With limits to test being if the access is fully revoked by moving out of the range for the Bluetooth connection. As well as if similar results are achieved when the access of the user is revoked after a set time period. With the locking unit including a built in clock and calendar that should theoretically prevent access outside of the set time the user is allowed access.

Chapter 9

Conclusions

We performed several different attacks on the smart lock unit in order to find any vulnerabilities. Our methodology were largely based on previous work and it can be concluded that previously found vulnerabilities were also found within the device. The weaknesses found may however be difficult to exploit. Nevertheless we still believe there is a need to addresses them, and especially since it is very important for a smart lock to designed in a secure way.

We discovered that a handshake key could be found in the system files where the mobile application used to control the lock was stored in plain text. With potential implications being that the smart lock could be controlled without using the official application. Additionally the device was vulnerable to DoS attacks, specifically over a Wi-Fi connection. Resulting in the lock not being operable through a Wi-Fi connection and potential for it being exploited by an attacker.

Another found vulnerability like previous research has mentioned lies in how certain common passwords are not discouraged. Resulting in a higher success rate for brute force attacks which also is highlighted by the fact that there are no limits to login attempts. However the device implements two factor authentication which makes the vulnerability found less severe. A state consistency attack was also found as successful albeit less so than has been previously found.

Some further research has also been suggested such as trying to control the lock with the handshake key and expand upon the MITM attack performed.

This study further contributes to our understanding of how to design in a secure way but further research even for this specific device is needed.

Bibliography

[1] Anthony Rose and Ben Ramsey. *Picking Bluetooth Low Energy Locks from a Quarter Mille Away*. DEF CON. https://doi.org/10.5446/36217 $Lastaccessed : 14Mar$2021. 2016.

[2] Jmaxxz. *The August Smart Lock's not so smart password reset (Part 2)*. https://jmaxxz.com/blog/?p=498$Lastaccessed : 14Mar$2021. 2015.

[3] S. Patil et al. "Ethical hacking: The need for cyber security". In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. 2017, pp. 1602–1606. DOI: 10.1109/ICPCSI.2017.8391982.

[4] James Conrad. "Seeking help: the important role of ethical hackers". eng. In: *Network security* 2012.8 (2012), pp. 5–8. ISSN: 1353-4858.

[5] Michael Howard and David LeBlanc. *Writing Secure Code: Practical Strategies and Proven Techniques for Building Secure Applications in a Networked World*. eng. Redmond: Microsoft Press, 2014. ISBN: 0735617228.

[6] Adam Shostack. *Threat modeling : designing for security*. eng. 1st edition. 2014. ISBN: 1-118-82269-2.

[7] Ahmad Shabir Hussain Shafiq Kamal Asif and Iqbal Sajid. "Threat Modelling Methodologies: a survey". eng. In: *Sci Int. 2014;26(4):1607–1609* (2014). ISSN: 1013-5316.

[8] P. Johnson, R. Lagerstörm, and M. Ekstedt. "A Meta Language for Threat Modeling and Attack Simulations". In: *ARES 2018: Proceedings of the 13th International Conference on Availability, Reliability and Security*. 38. 2018, pp. 1–8. DOI: 10.1145/3230833.

[9] Mathias Ekstedt et al. “Securi CAD by Foreseeti: A CAD Tool for Enterprise Cyber Security Management”. In: *2015 IEEE 19th International Enterprise Distributed Object Computing Workshop*. 2015, pp. 152–155. DOI: 10.1109/EDOCW.2015.40.

[10] Tejasvi Alladi et al. “Consumer IoT: Security Vulnerability Case Studies and Solutions”. In: *IEEE Consumer Electronics Magazine* 9.2 (2020), pp. 17–25. DOI: 10.1109/MCE.2019.2953740.

[11] Brittany D. Davis, Janelle C. Mason, and Mohd Anwar. “Vulnerability Studies and Security Postures of IoT Devices: A Smart Home Case Study”. In: *IEEE Internet of Things Journal* 7.10 (2020), pp. 10102–10110. DOI: 10.1109/JIOT.2020.2983983.

[12] Miao Yu et al. “A Survey of Security Vulnerability Analysis, Discovery, Detection, and Mitigation on IoT Devices”. In: *Future Internet* 12.2 (2020). ISSN: 1999-5903. DOI: 10.3390/fi12020027. URL: https://www.mdpi.com/1999-5903/12/2/27.

[13] Aditya Gupta. “Internet of Things: A Primer”. In: *The IoT Hacker’s Handbook: A Practical Guide to Hacking the Internet of Things*. Berkeley, CA: Apress, 2019, pp. 1–16. ISBN: 978-1-4842-4300-8. DOI: 10.1007/978-1-4842-4300-8_1. URL: https://doi.org/10.1007/978-1-4842-4300-8_1.

[14] Marko Pavelić et al. “Internet of Things Cyber Security: Smart Door Lock System”. In: *2018 International Conference on Smart Systems and Technologies (SST)*. 2018, pp. 227–232. DOI: 10.1109/SST.2018.8564647.

[15] Lulu Liang et al. “A Denial of Service Attack Method for an IoT System”. In: *2016 8th International Conference on Information Technology in Medicine and Education (ITME)*. 2016, pp. 360–364. DOI: 10.1109/ITME.2016.0087.

[16] Qifeng Chen et al. “Denial of Service Attack on IoT System”. In: *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*. 2018, pp. 755–758. DOI: 10.1109/ITME.2018.00171.

[17] Belal Asad and Neetesh Saxena. “On the Feasibility of DoS Attack on Smart Door Lock IoT Network”. In: Feb. 2021, pp. 123–138. ISBN: 978-981-16-0421-8. DOI: 10.1007/978-981-16-0422-5_9.

[18] Yogeesh Seralathan et al. "IoT security vulnerability: A case study of a Web camera". In: *2018 20th International Conference on Advanced Communication Technology (ICACT)*. 2018, pp. 172–177. DOI: 10.23919/ICACT.2018.8323686.

[19] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. "Security Analysis of Emerging Smart Home Applications". In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 636–654. DOI: 10.1109/SP.2016.44.

[20] Yili Lu. "Research on authentication encryption mechanism based on intelligent door lock vulnerability risk". In: *MATEC Web of Conferences* 336 (2021), p. 08009. DOI: 10.1051/matecconf/202133608009.

[21] Raihana 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.

[22] Robin Gustafsson and Linus Janstad. "Säkerhetsutvärdering av smarta dörrlås utifrån ett kommunikationsperspektiv". B.S. Thesis. Malmö University, 2015.

[23] M. Ye et al. "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.

[24] Dong Wang et al. "Cracking IoT Device User Account via Brute-Force Attack to SMS Authentication Code". In: *Proceedings of the First Workshop on Radical and Experiential Security*. RESEC '18. Incheon, Republic of Korea: Association for Computing Machinery, 2018, pp. 57–60. ISBN: 9781450357579. DOI: 10.1145/3203422.3203426. URL: https://doi.org/10.1145/3203422.3203426.

[25] Arvid Viderberg. "Security evaluation of smart door locks". MA thesis. KTH, School of Electrical Engineering and Computer Science, 2019.

[26] Christopher Robberts and Joachim Toft. "Finding vulnerabilities in iot devices: Ethical hacking of electronic locks". B.S. Thesis. KTH, School of Electrical Engineering and Computer Science, 2019.

[27] Jiliang Wang et al. "BlueDoor: breaking the secure information flow via BLE vulnerability". In: *MobiSys '20: The 18th Annual International Conference on Mobile Systems, Applications, and Services*. New York, United States: Association for Computing Machinery, June 2020, pp. 286–298. DOI: 10.1145/3386901.3389025.

[28] Imano Williams and Xiaohong Yuan. “Evaluating the Effectiveness of Microsoft Threat Modeling Tool”. In: *Proceedings of the 2015 Information Security Curriculum Development Conference*. InfoSec ’15. Kennesaw, Georgia: Association for Computing Machinery, 2015. ISBN: 9781450340496. DOI: 10.1145/2885990.2885999. URL: https://doi-org.focus.lib.kth.se/10.1145/2885990.2885999.

[29] Hossain Shahriar, Md Arabin Islam Talukder, and Md Saiful Islam. “An Exploratory Analysis of Mobile Security Tools”. In: *2019 KSU Conference on Cybersecurity Education, Research and Practice*. 2019.

[30] D. Balfanz et al. “Secret handshakes from pairing-based key agreements”. In: *2003 Symposium on Security and Privacy, 2003*. 2003, pp. 180–196. DOI: 10.1109/SECPRI.2003.1199336.

[31] Zhaohui Wang, Rahul Murmuria, and Angelos Stavrou. “Implementing and Optimizing an Encryption Filesystem on Android”. In: *2012 IEEE 13th International Conference on Mobile Data Management*. 2012, pp. 52–62. DOI: 10.1109/MDM.2012.31.