EXAMENSARBETE INOM DATATEKNIK,
GRUNDNIVÅ, 15 HP

STOCKHOLM, *SVERIGE 2020*

IoT Pentesting: Obtaining the Firmware of a Smart Lock

ALEXANDER BORG

CARL ASTON FRANCKE

KTH
SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP

IoT Pentesting: Obtaining the Firmware of a Smart Lock

CARL ASTON FRANCKE

ALEXANDER BORG

Degree Programme in Computer Engineering
Date: June 11, 2020
Supervisor: Pontus Johnson
Examiner: Mathias Ekstedt
School of Electrical Engineering and Computer Science

Abstract

Consumer Internet of Things (IoT) has become increasingly popular over the past years and continues to grow with virtual assistants, wearable devices and smart home appliances. Within the consumer IoT market, smart locks have gained popularity.

Smart locks offer the consumers a convenient way of handling keys and access to their home. Enabling your front door to be controlled over the internet however, introduces new possibilities for an adversary to brake in. Therefore, the integrity and authenticity of the product must be ensured.

This thesis covers a security assessment of a smart lock, focusing on the firmware of the embedded devices as the main assets. Potential threats against obtaining and abusing the firmware are identified by threat modeling. Based on the identified threats, penetration tests are conducted to demonstrate the security of the firmware.

The results show that the firmware could not be obtained and that the product constitutes a good example within consumer IoT for how to manage the firmware of embedded devices.

Keywords

Internet of Things, Penetration testing, Threat modelling, Firmware, Hardware

Sammanfattning

Sakernas internet (IoT) har blivit allt mer populärart under de senaste åren och fortsätter att växa med produkter som virtuella assistenter, bärbara enheter och smarta hushållsapparater. Inom marknaden för IoT-produkter riktat mot konsumenter har smarta lås blivit vanligare.

Smarta lås erbjuder konsumenter ett bekvämt sätt att hantera nycklar och tillgång till sina hem. Genom att göra det möjligt att styra ytterdörren via internet introduceras dock nya möjligheter för en attackerare att bryta sig in. Därför måste intergriteten och autenticiteten av produkten säkerställas.

Den här examensarbetet omfattar en säkerhetsbedömning av ett smart lås, med fokus på firmware för de inbyggda systemen som huvudtillgångar. Potentiella hot mot att få tag på och missbruka firmware identifieras genom hotmodellering. Baserat på de identifierade hoten genomförs penetrationstester för att utvärdera säkerheten för firmware.

Resultaten visar att firmware inte kunde erhållas och att produkten utgör ett bra exempel inom IoT-produkter riktat mot konsumenter för hur man hanterar firmware för inbyggda system.

Nyckelord

Sakernas internet, Penetrationstesting, Hotmodellering, Firmware, Hårdvara

Contents

Chapter 1

Introduction

Connected devices have become a large part of our daily lives. Smartphones, tablets and laptops are for many people necessities not only for communication but also for all kinds of consumption. With the emergence of the Internet of Things (IoT), we are taking the next step in the connected evolution, connecting all kinds of devices to the internet.

Consumer IoT has become increasingly popular over the past years and continues to grow with virtual assistants [27, 21], wearable devices [52] and smart home appliances [48]. When more information and services are moved to internet connected devices, the potential damage caused by cyber attacks increases. At the same time, the great variety of IoT device implementations results in a larger attack surface [15], making efficient and well implemented security measures on these devices essential.

Within the consumer IoT market, smart door locks have gained popularity [42]. Smart door locks replace the mechanical deadbolt with one that can be controlled digitally, usually via a smartphone app. This offers the consumers a convenient way of handling keys and access to their home. Enabling your front door to be controlled over the internet however, introduces new possibilities for an adversary to brake in. Therefore, the integrity and authenticity of the product must be ensured. A common way to ensure these security properties is to conduct penetration tests. An IoT penetration test is the assessment and exploitation of various components present in an IoT device solution to help make the device more secure [20].

This thesis covers a security assessment of Glue Smart Lock, focusing on the firmware of the embedded devices of the system as the main assets. The assessment follows the procedure where threat modeling and penetration testing is conducted to evaluate the security of the chosen assets.

1.1 Thesis objectives

The objective of this thesis is to assess some aspects of the security of Glue Smart Lock. In a previous security assessment of the same system conducted by Viderberg [51], the author investigates attack vectors related to the technologies available over the internet. This thesis aims to extend his work and demonstrate whether the firmware of the Glue Lock and Glue WiFi Hub, regarded as the main assets, is secure or not given certain delimitations. This thesis aims to answer the research question: *Can the firmware of the Glue Lock and Glue WiFi Hub be obtained and in that case used for further exploitation of the system?* The outcome of this project will add further knowledge about the overall security of Glue Smart Lock and contribute to the IoT security research field.

1.2 Delimitations

Due to the time constraints of the degree project, this thesis will focus on the security of the firmware of the Glue Lock and Glue WiFi Hub. Viderberg suggested proposals for future work by looking at the vectors hardware and the bluetooth communication of Glue Smart Lock. This thesis will investigate the hardware however, it will not investigate the bluetooth communication due to time constraints. Other attack vectors and attacks already investigated by Viderberg will not be covered by this thesis.

1.3 Disposition

Following chapters are included in this report:

- Chapter 2 Background: This chapter covers an introduction to the subject and relevant theory.
- Chapter 3 Methodology: This chapter covers the selected methods for threat modeling and penetration testing.
- Chapter 4 The System Under Consideration: This chapters presents the system under consideration.
- Chapter 5 Threat model of the system under consideration: This chapter covers the threat modeling of the system under consideration.

- Chapter 4 Penetration testing: This chapter covers the penetration tests conducted.
- Chapter 7 Ethics and sustainability: This chapter covers the ethics and sustainability aspects of the thesis.
- Chapter 8 Results: This chapter covers the results of the penetration tests.
- Chapter 9 Discussion: This chapter covers a discussion of the results.
- Chapter 10 Conclusion: This chapter covers a conclusion of the thesis and suggestions for future work.

Chapter 2

Background

This chapter covers related security research on smart locks and the relevant theoretical background. Previous attacks and known vulnerabilities in similar systems are presented as well as the previous work done on the system under consideration. The final sections cover a theoretical background of firmware, firmware vulnerabilities and different firmware attacks.

2.1 Smart locks vulnerabilities

Smart locks has been subjects for several successful penetration tests. Both in proving security and exposing vulnerabilities.

Smart lock developers add features to their products to enhance the user experience and to create an additional value for their customers. This includes features such as temporary access keys, automated unlocking and access logs to give just a few examples. Adding features however, increases the complexity of the system and can potentially increase the difficulty to make it secure. This was demonstrated by Ho et al. [26], where an adversary could retain access to a lock after the temporary access key was revoked by disabling the internet connection on his or her smartphone. This is also known as a state consistency attack. In another security assessment of a smart lock, software engineer and security researcher Jmaxxz found a number of critical vulnerabilities when testing the August Smart Lock [12, 18]. These vulnerabilities, enabled unauthorized users to access functionality requiring higher privilege.

2.1.1 Glue Smart Lock

As stated in the delimitations in section 1.3, some aspects of the security of Glue Smart Lock has been assessed by Viderberg [51]. In his assessment, the scope of attacks were limited to attacks proven successful in similar systems. The scope of attacks consisted of the following attacks.

1. State consistency attack
2. Brute force attacks
3. Man-in-the-middle attack

One vulnerability was found using the state consistency attack described in section 2.1 [51].

2.2 Firmware

Connected devices are often constrained in terms of resources. Their processors and operating systems are less costly, less capable, and less sophisticated than those found in personal computers or servers [29]. Systems running on these devices will therefore have to be adapted and implemented efficiently.

Firmware is the software running on these constrained devices, enabling them to access hardware functionality and run application logic. It can be seen as the component that holds the key to the kingdom. Anything that could be extracted from the device can be found in the firmware [23].

Firmware usually consists of a bootloader, kernel, file system and other resources. The main objective of the bootloader is to initialize the hardware and start the kernel. It can be seen as the BIOS of a normal PC. The kernel handles communication with the hardware of the device and allocates resources such as memory and access to shared resources e.g. I/O. The file system contains specific code for the device such as programs for handling requests, unlocking or locking the smart lock. The programs are written in low level code such as C or C++.

Firmware is commonly developed by the Original Design Manufacturers (ODM) or in-house developers working with the Original Equipment Manufacturer (OEM). The firmware is compiled into binary files which are then loaded into non-volatile memory such as Electrically Erasable Programmable Read-Only Memory (EEPROM) and flash memory on the device [24].

To ensure security, being able to patch functionality and potential vulnerabilities is crucial. Therefore, the device needs to support updates of the

firmware either manually or Over-The-Air (OTA). Depending on the implementation, the security of the update process can vary.

2.3 Firmware vulnerabilities

As for any other type of software, bugs and bad design choices in firmware implementations can be exploited and used by an adversary to take control or gain unauthorized access to a system. In a white paper published by the firmware security company Refirm Labs, eight common firmware vulnerabilities and their implications are listed [29].

Begin of Table 2.1

Vulnerabilities	**Implications of the vulnerability**
Unauthenticated access	Allows unfettered access, making it trivial for attackers to gain access and controls of the device.
Weak authentication	Simple password-based authentication or weak cryptographic algorithms than can be broken by brute force attacks.
Hidden back-doors	While potentially helpful for customer support, they are staple features that hackers quickly identify and exploit, often with severe consequences.
Password hashes stored in firmware	Hard-coded passwords that users are unable to change, and default passwords that users rarely change, result in devices that are trivial to exploit. Exploited by the Mirai malware to create a botnet of roughly 2.5 million IoT devices. [22]
Encryption keys stored in firmware	Encryption keys are critical, but when stored in firmware, they can result in easily compromised devices.
Buffer overflow vulnerabilities	Use of insecure string handling functions such as strcpy, strcat, etc., instead of their more secure strncpy, strncat counterparts, may result in buffer overflows that can be exploited, creating denial of service and code injection attacks.

Continuation of Table 2.1	
Vulnerabilities	**Implications of the vulnerability**
Use of open source solutions with known vulnerabilities	Automated hacking tools include exploits targeting known vulnerabilities in open source platforms and libraries. The latest versions will frequently include fixes, yet many devices are released unpatched.
Debug services in production systems	While critical during development and testing, they provide unfettered access and control over the device.
End of Table 2.1	

Table 2.1: Common firmware vulnerabilities.

Firmware related vulnerabilities similar to the ones presented in table 2.1 can also be found in OWASP top 10 2018 [39] and are listed by several other security researchers [23, 7, 24].

2.4 Firmware attacks

Firmware is essential for any IoT ecosystem. By providing functionality to a device as well as storing sensitive information, it is an attractive target for potential adversaries. The following attacks are some of the most common ways to abuse and exploit potential vulnerabilities in firmware or to obtain firmware.

2.4.1 Man-in-the-middle attack

IoT systems require communication between endpoints. These endpoints typically include smartphone apps, cloud APIs, and the devices themselves. Using a man-in-the-middle (MITM) attack, an adversary can intercept the communication between two endpoints and listen to or modify data being transmitted [32]. An example of a MITM attack was shown by Jmaxxz [12], where he could control the communication between the smartphone app and the cloud API. This attack can also be used in various ways to obtain the firmware image as presented by Gupta et al. [24, 23].

2.4.2 Hardware attacks

The constraints of IoT devices can limit the software attack surface of the device. Some IoT devices serves a single purpose such as receiving a bluetooth signal or sending an alert via a trusted gateway to a cloud server. An alternate attack surface for IoT and embedded devices is attacking the hardware [11, 8]. For this attack surface, two of the most common entry points are the memory components and debug interfaces of the device. Both of these entry points can be used to obtain the firmware image which is shown by Gupta et al. [24, 23].

Memory components

Memory components are located on the printed circuit board (PCB) of the device and communicate using protocols such as Serial Peripheral Interface (SPI) or Inter-Integrated Circuit (I2C). It is possible for an adversary to gain physical access to these memory components and read the data stored on them.

Debug interfaces

Debug interfaces are intended to be used during development of the device. These interfaces are also exposed on the PCB of the device and can sometimes be connected to directly or with little effort using protocols such as Joint Test Action Group (JTAG) or Universal Asynchronous Receive Transmit (UART). It is possible for an adversary to gain physical access to these debug interfaces and connect to the device.

2.4.3 Firmware update attack

A firmware update attack targets the firmware update operation of a device [4]. In a report presented by Mansor et al. [33], firmware update attacks are divided into different categories. These categories are further elaborated on and put in a general context in [4].

- **Obtaining firmware**: Firmware is the main asset in the firmware update process. Without an adequate security, the firmware can be easily obtained.
- **Reverse engineering**: If an adversary is able to get the firmware image or the corresponding binaries, he or she might attempt to reverse engineer the operations of the firmware.

- **Firmware modification**: By analyzing the firmware, an adversary might attempt to modify its contents in order to introduce unauthorised functionality.
- **Obtaining access authorization**: Some devices need authorization for external devices to communicate with them. If an adversary can gain that authorization, he or she will be able to conduct different types of attacks on the victim devices.
- **Installing unauthorized firmware**: An adversary can install unauthorized firmware on the device. Once the unauthorized firmware is installed on the device, the adversary can potentially conduct additional attacks.
- **Unauthorized device**: An illegal device may pretend to be a legal device and get an authentic copy of the firmware. This can possibly result in privacy issues and losses to the device's manufacturer, along with the potential for further malicious activity.

Firmware update attacks have been successfully conducted in many cases. In [43] the author successfully performs a firmware modification attack by injecting code into the downloaded firmware. The author obtains the firmware by using a MITM attack described in section 2.4.1. Another example of a firmware update attack performed on the August Smart Lock was conducted by Jmaxxz [12]. Jmaxxz was able to install unauthorized firmware on the smart lock since the firmware was not signed.

Chapter 3

Methodology

The following chapter introduces the methodology used for this thesis. The initial part covers the selected approaches for threat modeling and identifying the potential threats. The second part focuses on the penetration testing methodology.

3.1 Threat modeling methodology

Threat modeling is a well known tool for identifying potential security threats against a system. One way of threat modeling is to first identify the assets and map out the architecture of the system. This is done in order to gain knowledge of any security flaws and how they can be exploited. Using a well-recognized threat modeling methodology ensures the scientific validity of the security assessment and assists in finding and evaluating potential attack surfaces.

The threat modeling methodology in this thesis is based on a combination of the methodology presented by Gupta et al. [24] and Clinton et al. [7]. The methodology takes a top down approach, starting from a high level of abstraction mapping out the system architecture, working its way down to specific parts of the system. This enables an iterative process for finding and investigating different threats. The combination of methodologies consists of the following steps.

1. Identify the assets of the system under consideration.
2. Identify potential adversaries.
3. Categorize possible attack surfaces.

4. Create a diagram to decompose the system architecture and analyze protocols and data flows.

5. Identify threats based on STRIDE for chosen asset.

3.1.1 STRIDE

STRIDE is a widely used threat modeling methodology created by Praerit Garg and Loren Kohnfelder at Microsoft. It assists in identifying threats to the Target of Evaluation (ToE) and categorizing them in order to understand if they violate any security properties of a system [47]. The six categories and the property they violate is presented as following.

Begin of Table 3.1		
Property violated	**Threat**	**Threat definition**
Authentication	**S**poofing	Pretending to be something or someone other than yourself.
Integrity	**T**ampering	Modifying something on disk, on a network or in memory.
Non-repudiation	**R**epudiation	Claiming that you didn't do something or were not responsible.
Confidentiality	**I**nformation Disclosure	Providing information to someone not authorized to see it.
Availability	**D**enial Of Service	Absorbing resources needed to provide service.
Authorization	**E**levation Of Privelege	Allowing someone to do something they're not authorized to do.
End of Table 3.1		

Table 3.1: STRIDE

3.2 Penetration testing methodology

Security assessments of a system can take different approaches. Depending on the prior knowledge of the system, these approaches are usually categorized as

Black box, White box and Grey box [24]. For this thesis, a black box approach is used with no prior knowledge of the system.

As is suggested by Gupta et al. [24], a comprehensive test should be performed on the entire IoT system and infrastructure. However due to previously mentioned scope of this thesis, conducting penetration tests to assess the security of the entire system will not be feasible. Instead, the penetration tests will be performed to evaluate the security of the firmware of the Glue Lock and Glue WiFi Hub.

3.3 Firmware

The security assessment of the firmware in this thesis follows the recipe suggested by Gupta et al. [24] with some additions presented by Gupta in [23]. The recipe contains a step-by-step methodology with suggested tools and approaches on how to perform penetration tests on firmware in IoT devices.

3.3.1 Obtaining firmware

The first step when penetration testing the firmware of an IoT device is to actually get hold of the firmware. Depending on vendor and type of device, finding a suitable method can be more or less difficult. A number of possible methods for different scenarios are presented in [24].

Downloading from vendor's website

Some product manufacturers make their firmware available for download from their websites to enable customers to update their devices locally e.g. via USB or micro SD card.

Proxying or mirroring traffic during device updates

If the firmware of the device is updated remotely, it could be obtained by intercepting the traffic using a MITM attack as described in section 2.4.1.

Dumping firmware directly from the device

By having physical access to the device, the firmware could be obtained by using the hardware attacks as described in section 2.4.2.

Analyzing the APK of the Android app

Firmware related information can sometimes be found in the code of the web or smartphone app that is used to control the device. By extracting the Android Package Kit (APK) the app can be decompiled and analyzed.

Googling

If none of the steps presented above succeeds in obtaining the firmware, searching for information online is the last resort.

3.3.2 Analyzing firmware

Once the firmware is obtained, the binary file can be analyzed using open source tools. Most of the information is stored in the file system as described in section 2.2. The file system can be extracted to analyze its content which is where most of the vulnerabilities described in table 2.1 exists.

Chapter 4

The system under consideration

This chapter describes the system under consideration. The first section provides information about supported features of the Glue Smart Lock. The sections that follows describes the main use case and the hardware of the Glue Lock and Glue WiFi Hub.

4.1 Glue Smart Lock

The Glue Smart Lock is an IoT consumer product that lets the user control door locks through the use of a smartphone app [19]. The system consists of three major components:

- Glue Lock
- Glue WiFi Hub
- Glue smartphone app
  - Glue (User app)
  - Glue Driver (InHome delivery partner app)

Each component will be described further in the following subsections.

4.1.1 Glue Lock

The Glue Lock replaces the mechanical deadbolt of the lock with one that can be controlled wirelessly over Bluetooth Low Energy (BLE) communication. It is mounted on the inside of the door, meaning physical keys can still be used

to unlock the door from the outside. The door can also be locked manually from the inside by using the thumb turn by hand or pressing it twice. When pressing the thumb turn twice, an eight second delay is added before locking the door. The lock is powered by four AA batteries that can easily be replaced.

4.1.2 Glue WiFi Hub

The Glue WiFi Hub is used to enable communication with the lock over the internet. The hub enables the user to control the lock on distances greater than the range of BLE. The hub supports communication over BLE to communicate with the lock, and TLS encrypted communication with the cloud API.

4.1.3 Glue smartphone apps

There are two separate Glue smartphone apps.

Glue user app

The Glue user app lets the user control the Glue Lock and Glue WiFi Hub. The user app is available for both Android and IOS and has support for multiple locks and hubs. Besides being able to lock and unlock, the smart phone app provides a number of handy features such as temporary guest keys, activity logs and InHome delivery [19].

Glue Driver app

The Glue Driver app is used by approved third party delivery companies. The Courier receives a one-time key in order to drop the goods inside the home of the Glue Smart Lock owner. Upon entering the home, the Glue Driver app starts a video recording that is sent to the Glue Smart Lock owner as a receipt. The Glue Smart Lock owner also receives an activity timeline in the case of InHome deliveries. The timeline lists all activities from the point in time where the courier receives the temporary key. This way the Glue Smart Lock owner can see timestamps of the courier unlocking and locking the door.

4.1.4 User privileges

The Glue Smart Lock divides the users into three categories with different privileges. The different categories are Owner, Resident and Guest.

Owner

The owner is the most privileged user. With owner access, a user is able to

- Lock and unlock doors (within BLE range)
- Remote lock and unlock doors (WiFi)
- View all recent user activity
- Invite and remove users

Resident

A user with resident privileges has the second highest level of privileges. This includes

- Lock and unlock doors (within BLE range)
- View their own recent activity

Guest

The guest user has the lowest level of privileges. This category is most often used for people that need temporary access to the home. This could be one time only or repeated, e.g. every week. With guest privileges, a user can

- Lock and unlock doors (within BLE range) for a limited time

4.2 Use case

There is one main use case of the Glue Smart Lock

1. User installs the lock on the door and plugs in the hub to a socket.
2. User registers an account in the smartphone app.
3. User registers the lock in the smartphone app.
4. User registers the hub in the smartphone app and pair it with the lock.
5. The system is now installed and the user have owner privileges

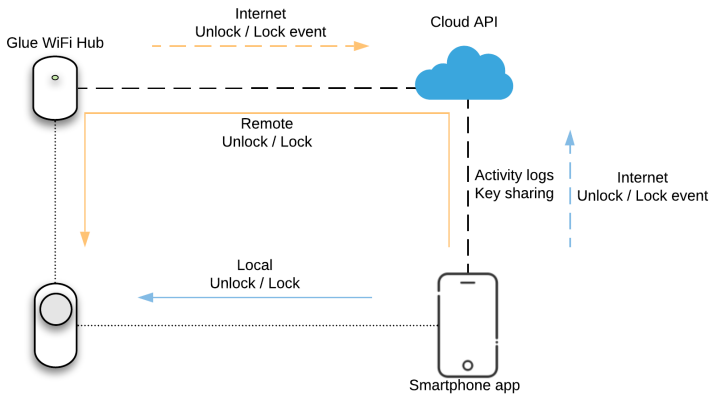


Figure 4.1: Use case architecture

When the user registers the lock in the smartphone app, he or she needs to calibrate the lock. The calibration tells the lock how much to turn the thumb turn in both directions to lock and unlock the door.

The hub requires an internet connection to communicate with the cloud API. Therefore, the hub is connected to the home WiFi during the registration step 4. The WiFi password is sent to the hub via BLE. If there is no internet connection between the hub and the cloud API the lock can only be controlled locally via the smartphone app.

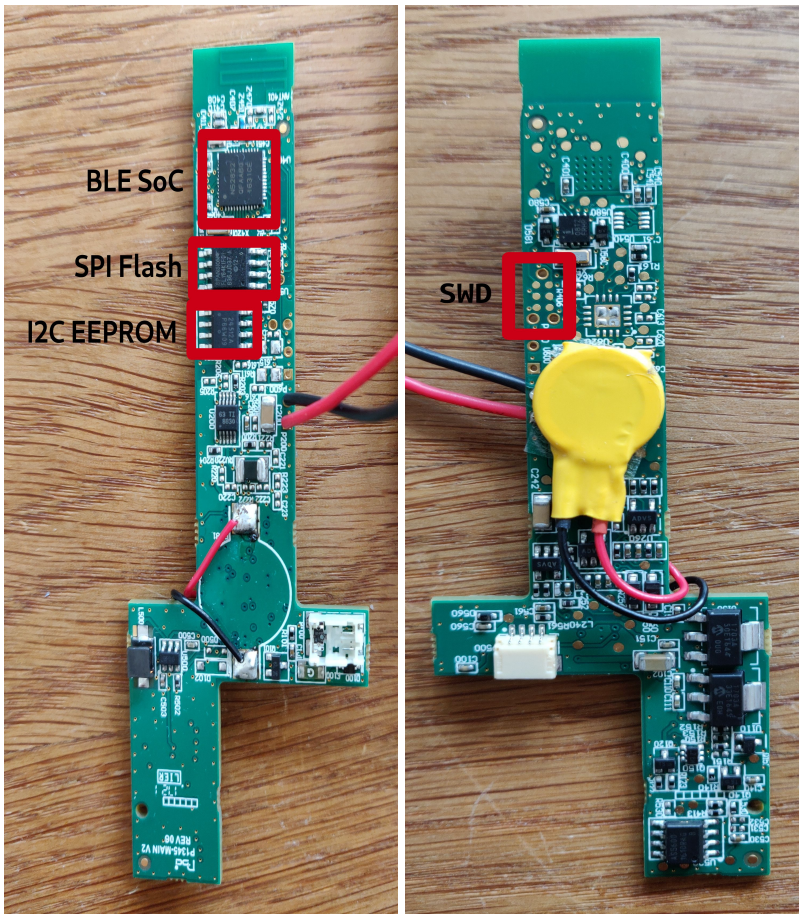
In the case where the smartphone app has no internet connection, the activity logs are not sent to the cloud API and the key sharing functionality is disabled. Once the smartphone app has an internet connection again, all the events that occurred during the time it was not connected are sent to the cloud API.

4.3 Hardware

An external inspection of the Glue Lock and the Glue WiFi Hub reveals no available ports such as USB ports or micro SD card slots. The internal inspection of the devices is presented below.

4.3.1 Glue Lock

The casing of the lock is opened by unscrewing five security bits. On the inside, a PCB with four components of interest can be identified.



(a) Side 1 PCB Glue Lock (b) Side 2 PCB Glue Lock

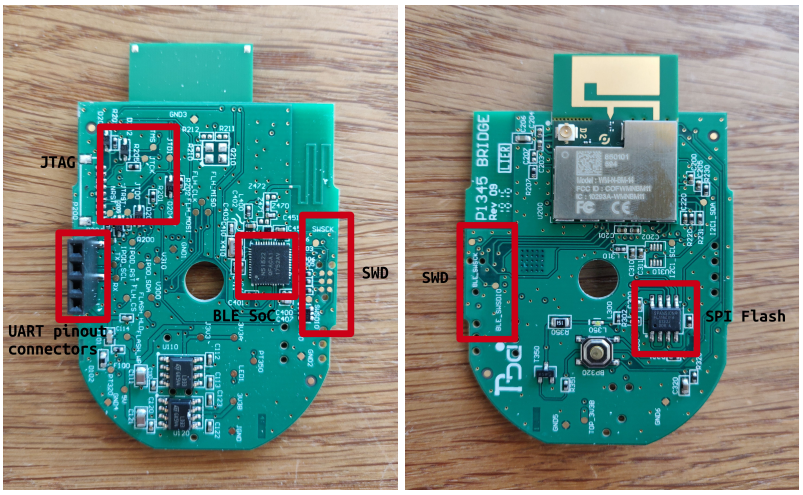
Figure 4.2: PCB Glue Lock

Begin of Table 4.1		
Component of interest	**Size**	**Protocols supported**
N52832 System on a chip (SoC) with built in Flash [41]	512/256 Kbyte	SPI / 2-wire / I2S / UART / PDM / QDEC
FL164KIF01 Flash memory [10]	8 Mbyte	SPI
24512A EEPROM [9]	64 Kbytes	I2C
Test points	-	SWD (Not verified)
End of Table 4.1		

Table 4.1: Components of interest on Glue Lock PCB

4.3.2 Glue WiFi Hub

The casing of the lock is opened by unscrewing one security bit and pushing out the sides. On the inside, a PCB with six components of interest can be identified.



(a) Side 1 PCB Glue WiFi Hub (b) Side 2 PCB Glue WiFi Hub

Figure 4.3: PCB Glue WiFi Hub

Begin of Table 4.2		
Component of interest	**Size**	**Protocols supported**
N51822 SoC with built in Flash [40]	256/128 Kbyte	SPI / 2-wire / UART
FL116KIF4 Flash memory [10]	2 Mbyte	SPI
Pinout connector	-	UART
Test points	-	JTAG, 2xSWD
End of Table 4.2		

Table 4.2: Components of interest on Glue WiFi Hub PCB

Chapter 5

Threat model of the system under consideration

This chapter covers the threat modeling of the Glue Smart Lock. The methodology described in section 3.1 is applied on the firmware of the Glue Lock and Glue WiFi Hub which are considered the ToE for this thesis and presented in the following sections.

5.1 Identify the assets

The term asset in a context of threat modeling refers to something an attacker wants to access, control or destroy [47]. Given this definition, assets are parts of the system that need to be protected. In the previously conducted threat model of the system [51], Viderberg identifies the following assets.

- Firmware
  - Firmware of the Glue WiFi Hub
  - Firmware of the Glue Lock
- Certificates and encryption keys
- Credentials
- Event logs
- Communication channels

What can be added to the list Viderberg provides are the video recordings that are created when a third party delivery company uses the lock as described in section 4.1.3. Other important assets not brought up by Viderberg due to scope limitations are hardware resources such as debug interfaces and storage. These assets will be of importance for this thesis in the sense of acting as entry points to the firmware.

All the above mentioned assets need to be stored securely as they could pose a serious threat to the system if disclosed. The attacks chosen in the previously conducted security assessment presented in section 2.1.1, smartphone app credentials and the communication channels are considered the main assets. To provide further knowledge about the security of the system, the main assets for the threat model in this thesis are the firmware of the Glue Lock and the Glue WiFi Hub. These assets are of most importance as they may also contain other assets.

5.2 Identify potential adversaries

In his threat model, Viderberg also identifies potential adversaries [51]. The adversaries are based on a generic adversary model presented by ARM [7] and are as follows.

- **Remote software attacker** - Has no direct access to the ecosystem of the lock or the hub.
- **Network attacker** - Has direct access to the ecosystem of the lock and the hub and can possibly eavesdrop or intercept traffic between endpoints.
- **Malicious insider attacker** - Employee inside the organisation with malicious intent and access to the device, e.g. Glue, part of the OEM or ODM supply chain.
- **Hardware attacker**
  - **Simple hardware attacker** - Has physical access to the lock or the hub but limited resources e.g. debug ports, tools to break casing of the lock or the hub.
  - **Advanced hardware attacker** - Has physical access to the lock or the hub and unlimited resources e.g. ion-beam lithography or microscopy probing.

In Viderbergs threat model, the hardware attacker is not considered due to being out of scope. In this thesis however, the simple hardware attacker is of interest due to the fact that the hardware is viewed as an entry point to the firmware.

5.3 Attack surfaces

The attack surfaces of the ToE can be divided into four main categories. Each category represents a possible entry point to the ToE that can be used as a way in for an adversary [7].

5.3.1 Communication attacks

Communication attacks exploit weaknesses in the communication between the endpoints of a system. Typical attacks are MITM attacks as described in section 2.4.1. This attack surface can be exploited by an adversary to obtain the firmware as proven by Gupta et. al [24] and applies to both the Glue Lock and the Glue WiFi Hub.

5.3.2 Software attacks

Software attacks are attacks that does not require physical access to the the lock or the hub [30]. The attack surface category includes attacks towards open ports of the devices e.g. if the firmware of the device includes hosting a web server or if a user can connect to the device via Secure Shell (SSH) or telnet. Software attacks are generally the most accessible attack surface and can target large numbers of devices anywhere in the world [30].

5.3.3 Lifecycle attacks

In his security assessment [51], Viderberg divides lifecycle attacks into two subcategories, User lifecycle and Product lifecycle.

User lifecycle

User lifecycle attacks exploit the moment in time where the ownership of the Glue Smart Lock changes hands e.g. user buys the system, user send the lock or hub to Glue for maintenance or user sells the system to another user [31]. The device could contain sensitive information about the previous user stored

in firmware, and the information could be disclosed by obtaining it. An adversary could also upload malicious firmware to attack the user in the next step of the lifecycle.

Product lifecycle

Product lifecycle attacks exploit vulnerabilities in systems that are no longer supported by the manufacturer. If the version of the lock or the hub is no longer supported and no more updates of the firmware are released. An adversary could possibly attack the device by applying newly found vulnerabilities to the unpatched Glue Lock or Glue WiFi Hub.

5.3.4 Hardware attacks

As opposed to software attacks, hardware attacks require physical access to the Glue Lock or Glue WiFi Hub. By accessing any of the two devices' hardware, an adversary could tamper with or access the firmware of the device via debug interfaces or storage components on the PCB as described in section 2.4.2.

5.3.5 Attack surfaces of the ToE

By compiling the information presented in chapter 4, a diagram over the system architecture can be created as a visual representation of the potential attack surfaces.

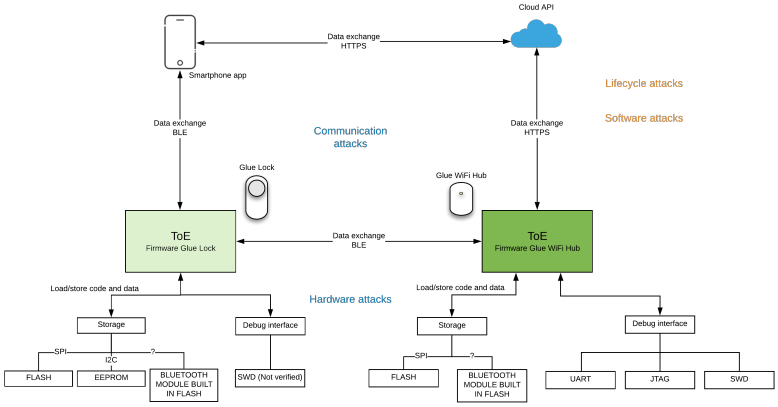


Figure 5.1: Diagram of potential attack surfaces

5.4 STRIDE

The threats are categorized using STRIDE. Table 5.1 lists the threats regarding firmware as the main asset and ways of obtaining it. To further illustrate the importance of a security assessment focusing on the firmware, table 5.2 lists the threats regarding firmware as an entry point into the system where other important assets may be affected. For this thesis however, the threats listed in table 5.1 will pose as the foundation of the penetration test presented in chapter 6.

Begin of Table 5.1		
Threat	**Threat definition**	**Related attacks**
Spoofing	- Spoof the hub in the communication with the cloud API to receive firmware related information	MITM
Tampering	- Dump firmware by tampering with device storage - Dump firmware by tampering with debug interface - Trigger OTA firmware update by modifying request/response to cloud API and eavesdrop traffic	MITM, Hardware attacks
Repudiation		
Information disclosure	- Firmware related information stored in smartphone app - Firmware related information disclosed in debug interface developer console (cf. Tampering) - Firmware publicly available - Eavesdrop on firmware related information on the local network	MITM, Hardware attacks
Denial of service		
Elevation of privilege	- Dump firmware by gaining admin access to the device via debug interface(cf. Tampering) - Dump firmware by gaining admin access to the device via network service	Hardware attacks, Software attacks
End of Table 5.1		

Table 5.1: Security threats with firmware as an asset

Begin of Table 5.2		
Threat	**Impact**	**Affected assets**
Spoofing	- Spoofing the firmware version to access or modify functionality based on the firmware version	Firmware
Tampering	- Execute malicious code - Bypass secure boot of the hub and run code that has not been validated by OEM - Physically install malware to permanently change device behaviour - Exploit known vulnerabilities in libraries used in frimware	Firmware, Intellectual property (IP), Encryption keys, Credentials, Certificates, Event logs, Communication channels
Repudiation	- Deny actions performed on the hub, e.g. lock/unlock	Event logs
Information disclosure	- Reverse engineer IP - Explore vulnerabilities in firmware - Extract sensitive information stored in firmware file system	IP, Encryption keys, Credentials, Certificates
Denial of service	- Permanent bricking of system - Block BLE communications - Block network communications	Communication channels
Elevation of privilege	- Gain admin access to the device and execute malicious code (cf. Tampering)	IP, Encryption keys, Credentials, Certificates, Communication channels
End of Table 5.2		

Table 5.2: Security threats with firmware as an entry point

Chapter 6

Penetration testing

This chapter covers the actual penetration testing of Glue Smart Lock. The penetration testing is performed according to the recipe presented by Gupta et al. [24], described in section 3.3. Each penetration test will be presented in detail, providing a short introduction followed by a background and method. Finally the results will be given followed by a discussion.

6.0.1 Scope of attacks

Due to scope limitations, the penetration tests conducted in this thesis will focus on the threats presented in table 5.1. This includes different ways of obtaining the firmware using the related attacks, also stated in table 5.1.

6.1 Reconnaissance

Reconnaissance is the first step of any penetration test, whether it be on an IoT product, web or mobile application or any other kind of system. The process involves information gathering of the ToE using different techniques. The reconnaissance is important as it further exposes potential threats and attack vectors of the ToE that was listed during the threat modeling.

6.1.1 Method

External inspection

The external inspection objective is to identify exposed interfacing options such as UBS-ports and SD card slots. It also includes identifying hardware

tampering prevention mechanisms such as security bits and adhesives on the Glue Lock and Glue WiFi Hub.

Internal inspection

Following the external inspection, an internal inspection is carried out by opening the case of the devices. The internal inspection includes identifying interesting components on the PCB such as memory components and debug interfaces.

Communication

To identify communication and data flows, the Glue WiFi Hub and smartphone app are connected to a controlled hotspot. Once connected, the network traffic analysis tool, Wireshark [53], is used to analyze the traffic.

Network services and port scanning

Network services running on the Glue WiFi Hub are identified using the network discovery tool Nmap [36].

Public firmware information

Public firmware information is gathered by browsing the vendors website and searching for information on Google.

6.1.2 Results

External inspection

The external inspection showed no exposed interfacing options apart from the reset button on the hub. Security bits to prevent the case from being opened were identified hidden behind stickers and protection film.

Internal inspection

The internal inspection required removing the stickers, removing the protection film and unscrewing of the security bits. Once the case was opened, the PCB could be obtained for inspection. For the Glue Lock four components of interest could be identified. These components are presented in section 4.3.1. For the Glue WiFi Hub six components of interest could be identified. These components are presented in section 4.3.2.

Communication

Analyzing the traffic showed that all packets sent between the cloud API and both the smartphone app and the Glue WiFi Hub were TLS encrypted. Tracing the packets also revealed the flow of data between the cloud API, the smartphone app and the hub as described in section 4.2.

Network services and port scanning

Running nmap on the Glue WiFi Hub showed no ports were open.

```
Nmap 7.80 scan initiated Tue Apr 7 11:27:36 2020 as
   : nmap -A -T4 -Pn -p- -o glue.nmap 192.168.0.33
Warning: 192.168.0.33 giving up on port because re-
   transmission cap hit (6).
Nmap scan report for 192.168.0.33
Host is up (0.0070s latency).
All 65535 scanned ports on 192.168.0.33 are closed
   (48159) or filtered (17376)
```

This eliminates the software attack surface and the potential to exploit the firmware of the devices without having physical access to the hub.

Public firmware information

No downloadable firmware image or information about the firmware could be found on the vendor's website or by searching for it on Google. An indication about how the firmware is developed and technologies used could be derived from a public job ad [16]. The firmware version of the Glue Lock and Glue WiFi Hub is included in the smartphone app.

6.2 Proxying or mirroring traffic during device updates

If the firmware is not publicly available, the first step is to investigate if a firmware image can be obtained by proxying or mirroring traffic.

6.2.1 Introduction

With the identified architecture there are two possible scenarios for an OTA firmware update of both Glue Lock and Glue WiFi Hub. The first scenario

is updating the lock firmware via a smartphone connected to the lock. The second scenario is updating the hub or lock firmware via the hub. The firmware or firmware update information, such as an URL where the firmware could be downloaded from, could be obtained by controlling the traffic between the devices and the cloud API [24]. This can be done by attacking the update process with a MITM attack described in section 2.4.1 and 2.4.3.

6.2.2 Background

It is possible for a network attacker described in section 5.2 to intercept the traffic between either the smartphone app and the cloud API, or the Glue WiFi Hub and the cloud API during an update. If he or she is in control of the network they are connected to, a proxy can be set up to mirror or forward the traffic between the endpoints and the cloud API via the network attacker.

In order for a device to be patched and secured against vulnerabilities, the firmware has to support updates, either locally or OTA. The implementation of the update process can vary. Glue does not disclose how the firmware of their devices are updated. One implementation is described by Rieck [43] where a device checks if it has the latest firmware version installed by sending its current version to the cloud API and asks if a newer version is available. If there is a newer version available, the cloud API responds with an URL endpoint from where the device can request the new firmware. The firmware image is then requested and sent to the device that updates its firmware. The MITM attack conducted by Viderberg indicates that this implementation is used by Glue Smart Lock. This is due to the json parameters "FirmwareVersion" and "AvailableFirmwareVersion" being included in the HTTP responses from the cloud API [51]. This update model is also described as "Pull mode" by the Cloud Security Alliance [1].

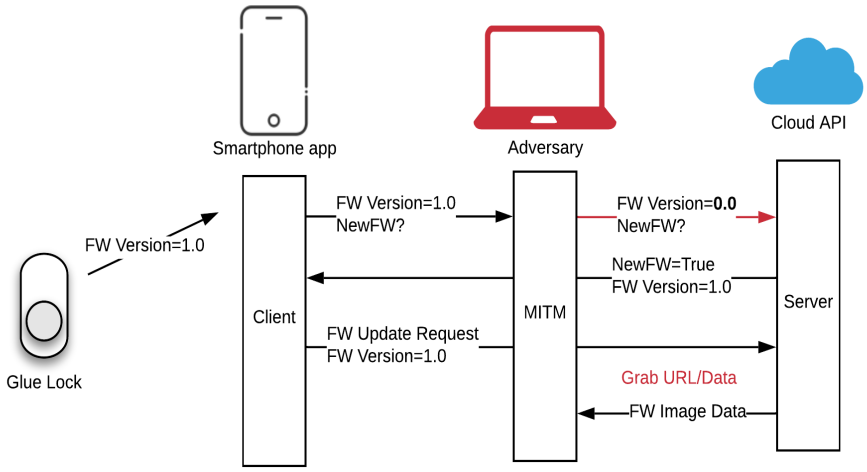


Figure 6.1: MITM Setup

To be able to use the app and send requests, a user has to be logged in. The result of the reconnaissance in section 6.1.2 showed that the traffic was encrypted and therefore has to be bypassed using decryption techniques.

6.2.3 Method

The methodology of proxying or mirroring traffic during device updates differs slightly between the Glue Lock and Glue WiFi Hub.

Glue Lock

To intercept the traffic between the smartphone app and the cloud API, the MITM attack can be set up by using the open source tool Mitmproxy [34]. To use it, a certificate must be manually installed on the smartphone to bypass the encryption of the traffic. Mitmproxy allows the user to add custom scripts to intercept and modify traffic. This functionality is used to modify the requests and responses between the smartphone app and cloud API. To test the hypothesis of the firmware update process described in section 6.2.2, the response parameters "FirmwareVersion" and "AvailibleFirmwareVersion" can be modified to potentially trigger a firmware update. A potential issue with applying this hypothesis on Glue Smart Lock is that the device can communicate either with the smartphone app or the Glue WiFi Hub. In order to prevent the firmware or information regarding firmware to be sent via the Glue WiFi Hub,

the hub is disconnected. This enforces any data to be transmitted to the lock via the smartphone app.

Glue WiFi Hub

Certificates cannot be installed on the Glue WiFi Hub manually to bypass the encrypted traffic between the Glue WiFi Hub and the cloud API. Therefore, no MITM attack can be set up and no attack will be conducted for the device.

6.2.4 Results

When setting up the Mitmproxy, it was noticed that the json data parameters was different from the results presented by Viderberg [51]. The parameter called "firmwareVersion" was used instead of the previous "FirmwareVersion", and the parameter "AvailableFirmwareVersion" was no longer included in the response.

By setting up the proxy, the traffic between the smartphone app and the cloud API was decrypted and analyzed. When the user launches the smartphone app, several GET requests are sent to the cloud API to fetch all the data available for the account. This includes registered properties, hubs and locks. By modifying the response parameter "firmwareVersion" using a script (See Appendix A), the user interface (UI) of the smartphone app displayed the modified data. As soon as the MITM attack was canceled, the UI displayed the actual firmware version again. However, no response or other action such as a firmware update was triggered. During the registration of the lock, the analyzed traffic showed a http PATCH request with information about the current firmware version of the lock. By modifying the "firmwareVersion" parameter in the request data, using a script (See Appendix A), it was possible to modify the firmware version stored in the cloud API for the lock. The response includes the modified firmware version and the UI now displayed the modified firmware version persistently without having to constantly modify the response during the GET requests. However, no other action such as a firmware update was triggered.

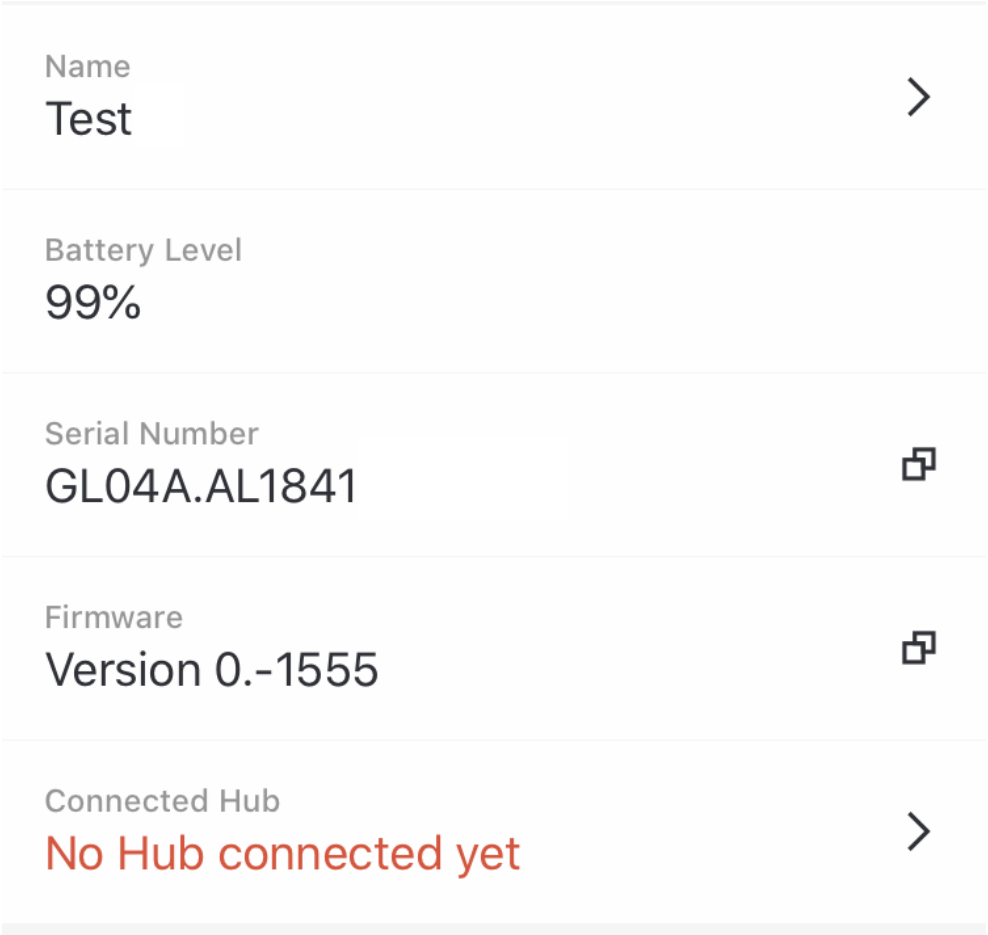


Figure 6.2: Modified firmware version in the smartphone application

6.2.5 Discussion

Securing the firmware update process is important in many aspects. One of them being preventing it from being downloaded and modified to break the authenticity and integrity of the code running on the devices. It is also important to prevent adversaries to analyze and find vulnerabilities in the code if the firmware is not publicly available. Since a firmware update was not triggered by modifying the communication between the smartphone app and the cloud API, it can confidently be said that the system withstood the MITM attack to dump the firmware image. There are two possible reasons for this result.

One reason is that the Glue Smart Lock uses what is described as a "Push mode" update model [1] and the firmware update is initiated server side. This would prevent adversaries from triggering the firmware update by sending re-

quests, unless they were in control of the firmware update server. This theory is strengthen by the fact that the parameter "AvailableFirmwareVersion" is no longer included in the API response.

The other reason is that the firmware image is sent via the Glue WiFi Hub. This would protect the communication being vulnerable to a MITM attack unless the TLS encryption of the communication could be decrypted by attacking the certificate validation process.

Implementing one or both of these countermeasures would make the MITM attack to dump the firmware image require a lot more effort, if even possible at all. For this thesis, an attack of the certificate validation process is out of scope due to time constraints.

What should be mentioned is that the possibility of conducting a MITM attack on the BLE communication between the devices and the smartphone app is not investigated due to scope limitations.

6.3 Dumping the firmware directly from the device

If the firmware cannot be obtained by mirroring or proxying traffic during firmware updates, attacking the hardware itself is a common approach.

6.3.1 Introduction

The compiled binary image of the firmware is stored in non-volatile memory as described in section 2.2. By locating the memory components on the PCB, an adversary can try to extract the firmware image by dumping the memory contents. During the reconnaissance phase of the penetrations testing, a SoC with built in flash, an EEPROM and a flash component was identified on the Glue Lock and a similar SoC and flash component on the Glue WiFi hub. The components are described in detail in table 4.1 and 4.2.

If the memory components cannot be accessed directly, debug or serial communication interfaces can sometimes be used for the same purpose. The identified interfaces include UART, JTAG and SWD. The debug interfaces are also listed in table 4.1 and 4.2.

6.3.2 Background

Having access to the device hardware, it is possible to connect to the pins of the flash or EEPROM and dump the contents using an external device. The external device is used to enable communication using the supported communication protocol, e.g. SPI or I2C. Among the more popular external devices are Bus pirate [6], Shikra [50] and Attify badge [3].

The pins on the memory components can be accessed directly on the PCB using an SOIC clip or by desoldering and removing the component from the PCB. While the former approach is more accessible and less invasive, the latter is sometimes preferred. This is due to the fact that the supply voltage provided through the SOIC clip to the component sometimes powers other components on the PCB as well. In this case, some of the other components might try to communicate with the memory component and thus block the communication between the memory component and the external device [11].

Another approach is to connect to a debug interface that are exposed on the PCB. Having access to a debug interface, an adversary can read and/or write data to the device using serial communication. Accessing the device using UART could allow an adversary to see logs of the booting sequence and sometimes even provide a shell [23]. JTAG and SWD will allow for an adversary to debug running processes [24]. However, debug interfaces such as JTAG will sometimes be deactivated, either by software or hardware, which makes it more difficult to exploit.

6.3.3 Penetration testing methodology of the Glue Lock

The SPI Flash and I2C EEPROM identified during the reconnaissance are selected as initial attack vectors to obtain the firmware image. This is due to the fact that one of these components will likely contain the firmware image and that they are the most accessible components on the PCB.

SPI Flash

To dump the contents of the SPI flash, an SOIC clip is used to connect the flash memory to a Bus pirate using the connections as shown in table 6.1. The Bus pirate is connected to a computer running a serial terminal. The SOIC clip is used both with the SPI flash remaining on the PCB and removed from the PCB.

Begin of Table 6.1	
Bus pirate	**FL164KIF01**
CS	CS#
MISO	SO/IO1
MOSI	SI/IO0
GND	Vss
+ 3.3V	Vcc
End of Table 6.1	

Table 6.1: Connections between Bus pirate and SPI flash

Having all the connections setup, Flashrom is used to identify and read the contents of the flash chip. Flashrom is a utility for identifying, reading, writing, verifying and erasing flash chips [17].

I2C EEPROM

To dump the contents of the EEPROM, a similar approach is used. The Bus pirate is connected to the chip using the SOIC clip and a serial terminal on a computer. The connections can be seen in table 6.2.

Begin of Table 6.2	
Bus pirate	**24512A**
CLK	SCL
MOSI	SDA
GND	Vss
+ 5V	Vcc
End of Table 6.2	

Table 6.2: Connections between Bus pirate and I2C EEPROM

To read the contents of the EEPROM, a simple python script is used (See Appendix B). The script makes use of the different Bus pirate commands to interact with, and read from the EEPROM. The script also parses the output into a binary file. To view the contents, Hexdump is used.

6.3.4 Penetration testing methodology of the Glue WiFi Hub

The SPI flash and UART pinout connectors identified during the reconnaissance are selected as initial attack vectors to obtain the firmware. This is for the same reasons as the lock presented in section 6.3.3. Other attack vectors selected for the Glue WiFi Hub include JTAG and SWD.

SPI flash

To dump the contents from the SPI flash, the same method as for the lock is used. The chip is connected to using a SOIC clip and the Bus pirate. The connections are presented in table 6.1. The Bus pirate is connected to a computer running a serial terminal. The SOIC clip is used both with the SPI flash remaining on the PCB and removed from the PCB.

UART

To interact with the UART pinout connectors, each pin need to be identified. Tx and Rx are printed and well visible on the PCB. Vcc and GND however, are not marked. To identify these pins, a continuity test can be made [24]. With all pins identified, the pinouts are connected to using a Bus pirate with the connections presented in table 6.3.

Begin of Table 6.3	
Bus pirate	**UART**
MISO	Tx
MOSI	Rx
GND	GND
+ 5V	Vcc
End of Table 6.3	

Table 6.3: Connections between Bus pirate and UART

To interact with the device using UART, the Bus pirate's built in transparent bridge is used in the serial terminal. The correct baudrate is identified by simply checking that the output printed in the serial terminal is readable.

JTAG

In this case, the easiest way to interact with the device via JTAG is to connect the Bus pirate to the JTAG test points located on the PCB. This is done by simply soldering some jumper wires to each test point. The test points are all labeled and well visible on the PCB so no reverse engineering is needed. The connections to the Bus pirate can be seen in table 6.4.

Begin of Table 6.4	
Bus pirate	**JTAG**
MISO	JTDO
MOSI	JTDI
CLK	JTCK
CS	JTMS
AUX	JTRST (optional)
GND	JGND
+ 3.3 V	J3V3
End of Table 6.4	

Table 6.4: Connections between Bus pirate and JTAG

Having all connections setup and the Bus pirate connected to a computer, the open on-chip debugger Openocd [37] is used to connect to the device. When a connection is established, Openocd enables the user to connect to the device via telnet and to do remote debugging with gdb. This way the memory contents of the device can be extracted by simply running a command as long as the memory contents is not read protected.

SWD

To interact with the device using SWD, an approach similar to the one for JTAG can be used. Jumper wires are soldered to the SWD test points on the PCB and connected to the Bus pirate as seen in table 6.5.

Begin of Table 6.5	
Bus pirate	**JTAG**
MOSI	SWSDIO
CLK	SWSCK

Continuation of Table 6.5	
Bus pirate	**JTAG**
End of Table 6.5	

Table 6.5: Connections between Bus pirate and SWD

To add a supply voltage, the UART pinout connectors are used as they can provide voltage to the entire PCB. Since Openocd also supports SWD, the same tools and techniques as with JTAG are used to communicate with the device.

6.3.5 Results of the Glue Lock

SPI flash

The SPI flash is successfully identified running Flashrom. The memory dump however, shows the entire memory only contains 0xFF as shown in figure 6.3. In other words, dumping the firmware using this approach is not possible.

```
➜  ~ hexdump lockflash.bin
0000000 ffff ffff ffff ffff ffff ffff ffff ffff
*
0800000
```

Figure 6.3: Memory contents of SPI flash on the Glue Lock

I2C EEPROM

Extracting the contents of the I2C EEPROM, a few memory addresses of interest are identified running Hexdump.

```
00008000  67 33 49 42 6f 37 30 6f  31 46 37 31 35 39 78 77  |g3IBo70o1F7159xw|
00008010  7a 62 48 4a 36 6a 34 35  33 49 6e 33 56 6f 75 46  |zbHJ6j453In3VouF|
00008020  ba ff ff ff ff ff ff ff  77 4b 61 73 6c 78 68 32  |........wKaslxh2|
00008030  65 47 6d 64 42 4a 53 47  b7 ff ff ff ff ff ff ff  |eGmdBJSG........|
00008040  ff ff ff ff ff ff ff ff  ff ff ff 47 4c 30 34 41  |...........GL04A|
00008050  2e 41 4c 31 38 30 36 2e  30 30 30 31 b1 ff ff ff  |.AL1806.0001....|
00008060  00 3e 02 00 f4 00 ff ff  ff ff ff ff ff ff ff ff  |.>..............|
00008070  ff ff ff ff ff 26 ff ff  ff ff ff ff ff ff ff ff  |.....&..........|
00008080  ff ff ff ff ff 49 01 ff  ff ff ff ff ff ff ff ff  |.....I..........|
00008090  ff ff ff ff ff ff ff ff  ff ff ff ff ff ff ff ff  |................|
```

Figure 6.4: Memory contents of I2C EEPROM on the Glue Lock

The memory contents shown in figure 6.4 includes some interesting information. In the memory addresses ranging from 0x8000 to 0x8038 are two strings, one 16 byte string and one 32 byte string. In the memory address starting at 0x804b, the serial number of the lock is hard coded in plain text. Together with the serial number, these are the only values that will remain the same when running a memory dump after some action has been taken in the smartphone app, e.g. sharing or revoking a key, deleting the lock etc.

The I2C EEPROM contains some other information as well. On memory addresses starting at 0x0480 and ending at 0x0580, as well as addresses starting at 0x8900 and ending at 0x89e0, data is written depending on actions taken on the lock, e.g. sharing or revoking keys. The results show that the I2C EEPROM is used to store information, both statically and during run time. However, the firmware image is not stored on this component and cannot be extracted.

6.3.6 Results of the Glue WiFi Hub

SPI flash

Running Flashrom to dump the memory contents with the chip still on the PCB fails as Flashrom is not able to identify the component. Since the component was identified during the internal inspection phase of the reconnaissance, an attempt to force Flashrom to read the contents was made by setting a flag to specify the component manually. This however also failed. To verify the results, the built-in Bus pirate commands for SPI was used but without success.

With the SPI flash desoldered from the PCB, Flashrom was run successfully. However, the result shows the memory is empty, as seen in figure 6.5, meaning it is not possible to obtain a firmware image using this method.

```
→  ~ hexdump hub_flash_detached.bin
0000000 ffff ffff ffff ffff ffff ffff ffff ffff
*
0200000
```

Figure 6.5: Memory contents of SPI flash on the Glue WiFi Hub

UART

Connecting to the device via UART and a serial terminal reveals some information about the Glue WiFi Hub booting process and run time operations.

When connected, logs are printed to the serial terminal as show in figure 6.6, disclosing some information about the system.

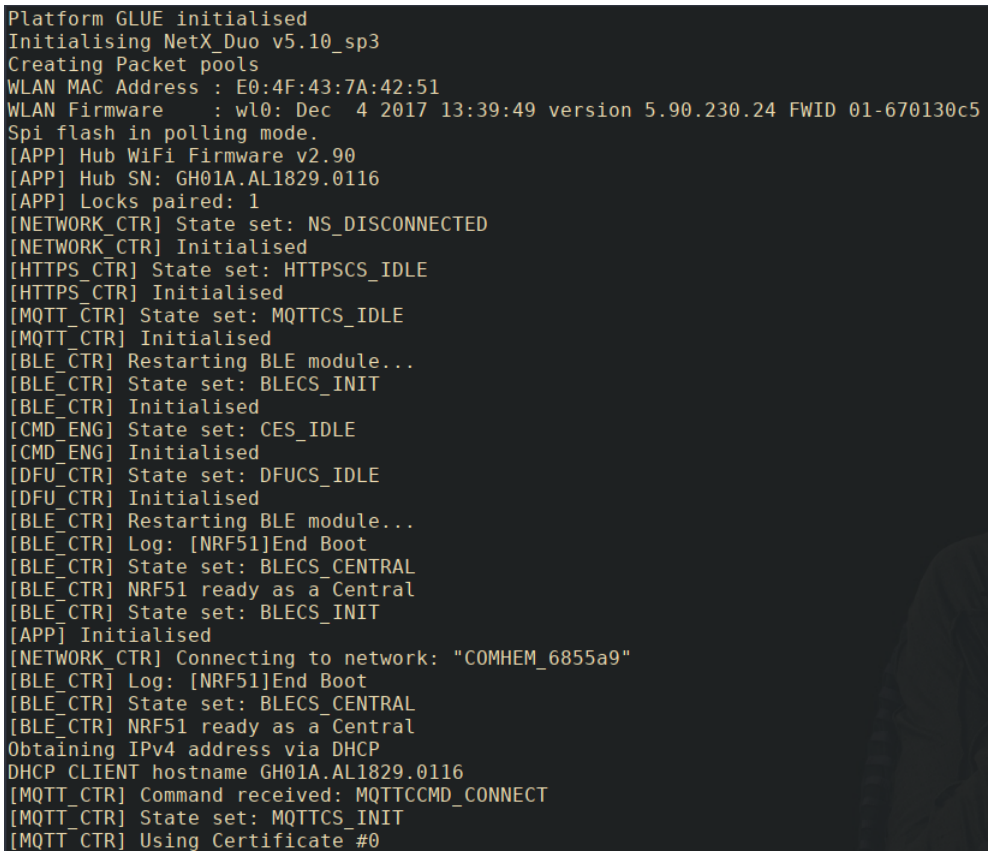


Figure 6.6: Logs printed to serial terminal via UART

While the logs provide some information about the system architecture, the connection never allows for any user input or returns a shell. This means no firmware image can be obtained using this method.

JTAG

Running Openocd auto probing [38], two test access ports (TAP) were identified. Auto probing discloses architecture and/or manufacturer. The TAPs were identified as a ST Microelectronics device, that through some reverse engineering of the memory mapping could be identified as an STM32F2X device with an ARM Cortex-M3 processor. This was later confirmed by removing the metal casing from the PCB that was previously remained untouched to prevent damaging the PCB.

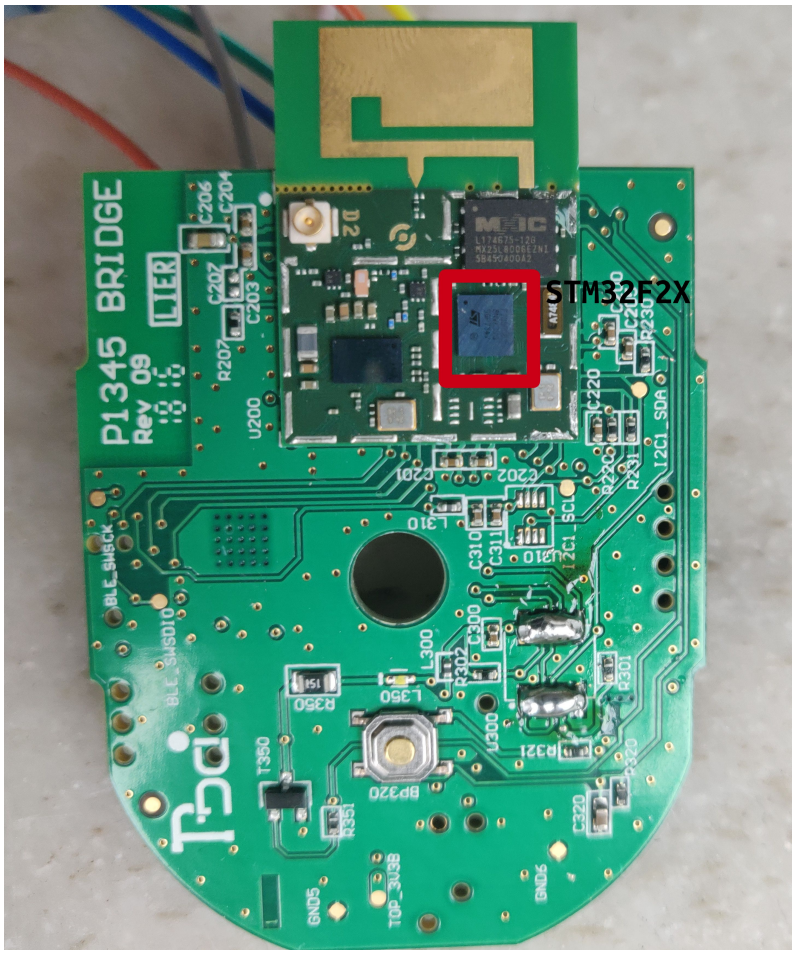


Figure 6.7: PCB of Glue WiFi Hub metal casing removed

Dumping the memory contents, both from flash and RAM, resulted in a binary file that could be analyzed using tools such as Binwalk [5], Hexdump [25], Xxd [54] and Strings [49]. The results of running Binwalk on the dumped memory contents can be seen in figure 6.8.

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
9540          0x2544          CRC32 polynomial table, little endian
264956        0x40AFC         Copyright string: "Copyright (c) Glue Home Ltd, 2019. All rights reserved"
270293        0x41FD5         PEM certificate
271554        0x424C2         PEM certificate
282421        0x44F35         PEM certificate
283759        0x4546F         PEM certificate
296864        0x487A0         CRC32 polynomial table, little endian
301209        0x49899         Certificate in DER format (x509 v3), header length: 4, sequence length: 177
415841        0x65861         Boot section Start 0x42423242 End 0x0
415845        0x65865         Boot section Start 0x0 End 0x0
416015        0x6590F         Boot section Start 0x3C42 End 0x0
416019        0x65913         Boot section Start 0x0 End 0x0
416502        0x65AF6         Boot section Start 0x14 End 0x30000
522446        0x7F8CE         PEM RSA private key
522692        0x7F9C4         SHA256 hash constants, little endian
523017        0x7FB09         PEM certificate
527168        0x80B40         Copyright string: "Copyright (c) 1996-2017 Express Logic Inc. * NetX Duo Cortex-M3/GNU Version G5.10.5.2 SN: 4712-108-0515 *"
```

Figure 6.8: Running Binwalk on the dumped flash memory contents

The output of running Binwalk on the binary file show some interesting headers including certificates, RSA private keys etc. Looking closer at these memory locations however, shows that no sensible information is stored on these locations. Also, no compressed data or bootloader information was shown, suggesting the firmware is not stored on this component.

SWD

Using the same approach for SWD as with JTAG it was not possible to connect to the device using Openocd.

6.3.7 Discussion

Using the techniques described above, dumping the firmware directly from the devices was not successful. Following the penetration testing methodology as described in section 3.2 however, the validity and reliability of the work can be assured as the most likely attack vectors have been thoroughly tested using several approaches.

While the firmware image could not be obtained, the memory contents of the investigated components and the UART logs gave some valuable information about the system architecture and where the firmware is stored.

Extracting the memory contents of the SPI flash chips on both the Glue Lock and the Glue WiFi Hub gave the same results, showing they were empty. Looking at the UART logs, it is disclosed that the SPI flash is set in polling mode. While it does not explicitly say which flash chip it is referring to, this strongly suggest that the SPI flash chips should be removed from the PCB in order to read the memory contents. The results however were consistent and showed the memory was empty even with the components removed from the PCB.

The memory contents on the STM32F2X on the Glue WiFi Hub, as well as the I2C EEPROM on the Glue Lock provided further information about the system. What is also shows is that the firmware image is stored on some other location.

The results strongly implies that the firmware image is stored on the internal flash memory of the BLE SoC, both on the Glue Lock and the Glue WiFi Hub, as this is the only memory components that could not be read.

What should not be seen as a definite result is the penetration tests using SWD as attack vector. Since JTAG was only enabled to debug the STM32F2X component, this strongly suggests that in order to read the internal flash mem-

ory contents of the BLE SoC, SWD must be used. Due to time limitations of this thesis, the SWD debug interface could not be investigated sufficiently.

The results of these penetration tests suggest further research on the ToE's SWD debug interface as a way of obtaining the firmware image. Another suggestion would be to reverse engineer the STM32F2X machine code via JTAG to see if it can be used to abuse the BLE SoC as a way of obtaining the firmware image.

6.4 Analyzing the APK

As presented in section 3.2, analyzing the APK of the smartphone app is another approach to obtain the firmware.

6.4.1 Introduction

Firmware related information can sometimes be found in the code of the web or smartphone app that is used to control the device. By having access to the APK, an adversary can analyze the app and look for vulnerabilities to obtain the firmware [24].

6.4.2 Background

Developing applications with complex functionality can be hard. In the case of a smart lock system there is sensitive data that has to be stored somewhere, such as credentials, ids, secrets and code that handle certain functionality. Sometimes developers makes mistakes or opt for an easy solution by hard coding sensitive information into the application. The information can then be extracted and exploited using open source technologies for decompiling and analyzing the APK file [24, 14, 13].

6.4.3 Method

To analyze the APK, the Glue User app is downloaded from Google Play Store using an Oneplus 6. The APK file is then extracted from the smartphone to a computer using Apktool [2]. Using the open source tool MobSF [35] which automates static and dynamic analysis of smartphone apps, the APK file is then analyzed.

MobSF is used to look for hard coded sensitive information regarding firmware such as firmware update URL's. It can also be used to look for func-

tionality related to how the firmware of the Glue Lock and Glue WiFi Hub is updated.

6.4.4 Results

Running MobSF creates a web interface for analyzing any APK file. By uploading the APK of the Glue user app an overview was created as shown in figure 6.9.

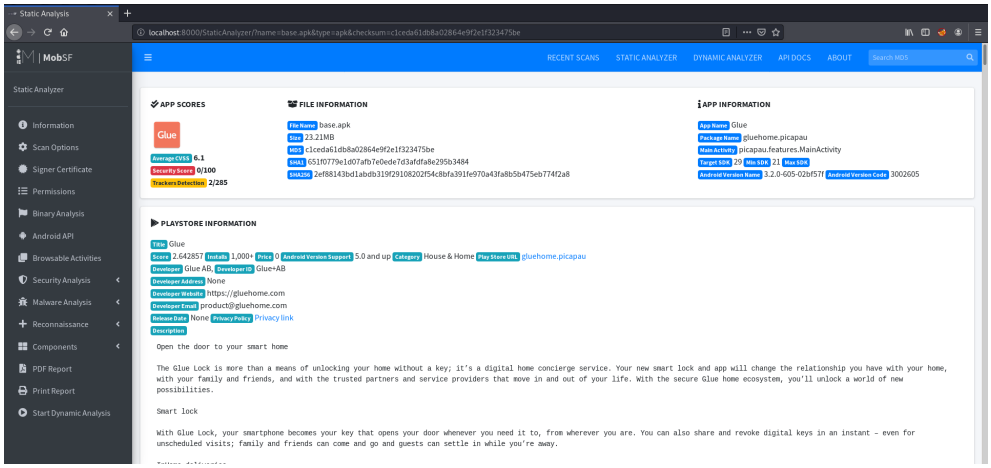


Figure 6.9: MobSF web interface

MobSF analyzes the APK automatically and searches for potential vulnerabilities. Looking at the section "code analysis" in the web interface, potential vulnerabilities are listed with "hardcoded sensitive information" being one of them. Looking through the files, no vulnerabilities could be found and no sensitive information was hard coded.

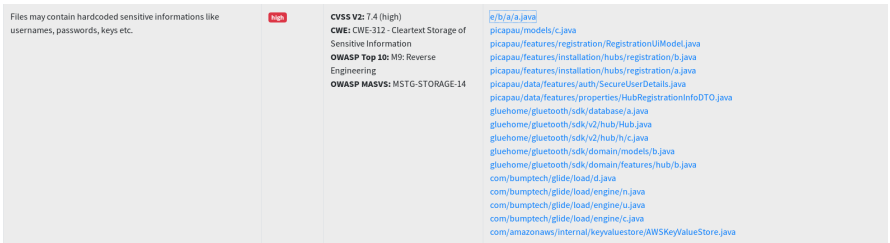


Figure 6.10: MobSF sensitive information

MobSF supports searching for strings in all of the included files. Searching for "firmware" resulted in 20 files containing at least one occurrence of the

string. Looking through the files, most of them contained code for handling API requests and responses. One file called "GlueLockV2Firmware.java" contained functions that could be of interest for obtaining firmware but the code was obfuscated, making it hard to reverse engineer, this is presented in figure 6.11.

```
static final class c<V> implements Callable<io.reactivex.e> {
    final /* synthetic */ GlueLockV2Firmware u;

    c(GlueLockV2Firmware glueLockV2Firmware) {
        this.u = glueLockV2Firmware;
    }

    /* JADX WARNING: type inference failed for: r2v0, types: [gluehome.gluetooth.sdk.v2.b] */
    /* JADX WARNING: Multi-variable type inference failed */
    public final io.reactivex.a call() {
        this.u.p = CalibrationStep.UNLOCK;
        GlueLockV2Firmware glueLockV2Firmware = this.u;
        io.reactivex.a a = glueLockV2Firmware.a(glueLockV2Firmware.n.d());
        kotlin.jvm.b.l a2 = this.u.z();
        if (a2 != null) {
            a2 = new b(a2);
        }
        return a.a((io.reactivex.g0.g<? super Throwable>) (io.reactivex.g0.g) a2);
    }
}
```

Figure 6.11: MobSF obfuscated code

6.4.5 Discussion

Since reverse engineering technologies are so well known, it is important not only to avoid storing sensitive information but also to prevent the code from being reverse engineered. This is easily achieved by obfuscating it. With unlimited amount of time, more exhaustive reverse engineering can be done to understand the application logic and data flows in detail. For this thesis however, exhaustive reverse engineering of the APK is out of scope due to time constraints.

Chapter 7

Ethics and sustainability

The following chapter covers ethcis and sustainability aspects of this thesis. Considering ethics, this chapter will also cover how the law is navigated and the principles of responsible disclosure.

7.1 Ethics and the law

Penetration testing and hacking is often spoken of in terms of White hat, Gray hat and Black hat hacking [55]. These different approaches follow different sets of ethical frameworks. This thesis project follows a white hat approach with a responsible disclosure policy as described in [28]. This means that if any vulnerabilities are found, they will be reported to Glue with a 90 day time frame to patch the vulnerabilities before the paper is published.

Another ethical aspect of this thesis is to ensure that all conducted penetration tests have been performed within the law as stated by the Swedish government. For this thesis, there are mainly four Swedish laws that need to be taken into consideration.

The Swedish Criminal Code chapter 4 9c § [44] states that anyone who unlawfully tries to gain access to a resource intended for automated processing or unlawfully modifies, obliterates, blocks or registers such a resource is sentenced for data breach of fines or imprisonment for a maximum of two years. The same applies to those who unlawfully interfere with any other similar action or hinder the use of such a resource. It also states that if the act has caused serious damage or intended for a large number of tasks or has otherwise been of a particularly dangerous nature, the sentence will be imprisonment for a minimum of six months and a maximum of six years.

The Swedish Criminal Code chapter 4 8 § also states that a person who

unlawfully obtains access to a communication being conveyed by a postal or tele-communications company as an item of post or in an electronic communications network is guilty of breach of postal or tele-communications secrecy and is sentenced to a fine or imprisonment for at most two years.

The Act on Trade Secrets [46] contains two paragraphs of certain interest, 2 § and 4 §. 2 § contains regulations on indemnity, prohibition of penalty, and penalties for unauthorized attacks on business secrets. § 4 however states that the law applies only to unauthorized attacks on business secrets. As an unauthorized attack, it is never considered that anyone is attacking a business secret to disclose or reveal to an authority or other appropriate body something which is

- reasonably suspected to be a crime with a prison sentence, or
- can be considered to be any other anomaly where the disclosure is to protect the public interest.

The Law on Copyright for Literary and Artistic Works [45] states that subject to the limitations prescribed hereinafter, copyright shall include the exclusive right to exploit the work by making copies of it and by making it available to the public, be it in the original or an altered manner, in translation or adaptation, in another literary or artistic form, or in another technical manner. However, it also states that anyone who has the right to use a computer program is entitled to observe, study or test the function of the program in order to as certain the ideas and principles which lie behind the various details of the program. This applies provided that the act is performed in connection with such loading, display on a screen, processing, transmission or storing of the program that he is entitled to make.

7.2 Sustainability

By conducting security assessment of consumer IoT products, the life span of these products can potentially be increased. If the product is of high quality in terms of security, especially for a smart lock, the incentive to keep the product is strengthened. This reduces the electronic waste. Another aspect is that if vulnerabilities are found and disclosed responsibly, the product manufacturers can save time and money on debugging and potential damage caused by security breaches.

Chapter 8

Results

A summary of the results of the conducted penetration tests are presented as a threat traceability matrix given in table 8.1. All results are related to the threats described in table 5.1. The main assets considered for all attacks are the firmware images of the Glue Lock or Glue WiFi Hub. All of the attacks were conducted with the goal of obtaining the firmware image of either of these devices. The indirect impacts of the conducted attacks for obtaining the firmware are summarized in table 5.2. It was not possible to obtain a firmware image using any of the conducted attacks.

Begin of Table 8.1				
Attack	**Attack surface**	**Adversary**	**Sufficiently tested**	**Control (Mitigation)**
Firmware publicly available	-	-	Yes Section 6.1.2	Firmware not publicly available
Dump firmware by gaining admin access to the device via network service	Network services	Remote software attacker, Network attacker	Yes Section 6.1.2	No network services available
Spoof Glue WiFi Hub in the communication with the cloud API to receive firmware related information	Communication between Glue WiFi Hub and cloud API	Network attacker	No Section 6.4.4 Discussion 6.4.5	TLS encrypted communication

Continuation of Table 8.1

Attack	**Attack surface**	**Adversary**	**Sufficiently tested**	**Control (Mitigation)**
Trigger OTA firmware update by modifying request/response to cloud API and eavesdrop traffic	Communication between smartphone app and cloud API	Network attacker	Yes Section 6.2.4	Server side initiated firmware updates or firmware image sent via Glue WiFi Hub
Eavesdrop on firmware related information on the local network	Communication between smartphone app or Glue WiFi Hub and cloud API	Network attacker	Yes Section 6.2.4	No sensitive information disclosed or TLS encrypted communication
Dump firmware image from external SPI flash	Hardware storage	Simple hardware attacker	Yes Section 6.3.5, 6.3.6	Firmware image not stored on external SPI flash
Dump firmware image from external I2C EEPROM	Hardware storage	Simple hardware attacker	Yes Section 6.3.5	Firmware image not stored on external I2C EEPROM
Dump firmware by tampering with debug interface JTAG	Hardware debug interface	Simple hardware attacker	Yes Section 6.3.6	Store firmware on internal memory of SoC
Dump firmware by tampering with debug interface UART	Hardware debug interface	Simple hardware attacker	Yes Section 6.3.6	Store firmware on internal memory of SoC

Continuation of Table 8.1				
Attack	**Attack surface**	**Adversary**	**Sufficiently tested**	**Control (Mitigation)**
Dump firmware by tampering with debug interface SWD	Hardware debug interface	Simple hardware attacker	No Section 6.3.6 Discussion 6.3.7	Store firmware on internal memory of SoC
Firmware related information disclosed in debug interface developer console	Hardware debug interface	Simple hardware attacker	Yes Section 6.3.6	No sensitive information disclosed
Firmware related information stored in smartphone app	Smartphone app APK	Remote software attacker	No Section 6.4.4 Discussion 6.4.5	No sensitive information stored or obfuscated code
End of Table 8.1				

Table 8.1: Threat traceability matrix

Chapter 9

Discussion

This thesis aimed to investigate how the firmware of the embedded components of the Glue Smart Lock was secured. All threat modelling and penetration testing have been done according to well known methodologies. Attack vectors were carefully selected, based on the identified threats. Penetration tests have been conducted according to a recipe developed by IoT security researchers with additional sources. Due to time limitation however, some attack vectors could not be investigated sufficiently. As is presented in the threat traceability matrix in table 8.1, some attack vectors need to be further tested in order to have a definite answer to the research question. These attack vectors include the certificate validation process of the Glue WiFi Hub, reverse engineering of the STM32F2X component on the Glue WiFi Hub PCB to abuse the N51822 BLE SoC, SWD hardware debug interfaces of the Glue Lock and the Glue WiFi Hub, more exhaustive reverse engineering of the Glue User app APK as well as the vector BLE communication between the devices and the smartphone app. Even though some attack vectors remains to be further investigated, it is fair to say that the firmware of the Glue Lock and Glue WiFi Hub as the main assets are secure to the extent that they withstood the attacks conducted in this thesis. What should also be mentioned is that a successful extraction of the firmware images would not necessarily mean that the system is insecure. If a firmware image was obtainable there could possibly be other protection mechanisms that prevent it from being analyzed and reverse engineered e.g. encryption. There is also the possibility, even though less likely, that there are no vulnerabilities with the firmware and the implementation of the functionality is perfect. Since no firmware image could be extracted, the second part of the research question could not be answered in this thesis.

Chapter 10

Conclusions

In this thesis, obtaining the firmware of the Glue Lock and the Glue WiFi Hub was not possible. However, the results of the penetration tests give valuable information about the system architecture and where the firmware is most likely to be stored on both devices, on the BLE SoC. The results suggest that further research of the non-explored attack vectors is needed in order to confidently state that the firmware is not obtainable. The penetration tests conducted in this thesis covers many of the most common attack vectors for obtaining the firmware of an IoT device. With Glue Smart Lock withstanding all of them, the potential attack surface of the system in general has been reduced. The results of the thesis combined with the result of the security assessment conducted by Viderberg [51], what can be said about Glue Smart Lock is that the level of security is high. In general, Glue Smart Lock constitutes a good example within consumer IoT products for how to handle firmware, both considering updating mechanisms and storage.

Bibliography

[1] Cloud Secutiry Alliance. *Recommendations for IoT Firmware Update Processes*. 2018. URL: https://downloads.cloudsecurityalliance.org/assets/research/internet-of-things/recommendations-for-iot-firmware-update-processes.pdf (visited on 05/19/2020).

[2] *Apktool - A tool for reverse engineering 3rd party, closed, binary Android apps.* URL: https://ibotpeaches.github.io/Apktool/ (visited on 05/24/2020).

[3] *Attify Badge - UART, JTAG, SPI, I2C (pre-soldered headers)*. URL: https://www.attify-store.com/products/attify-badge-uart-jtag-spi-i2c-pre-soldered-headers (visited on 05/13/2020).

[4] Meriem Bettayeb, Qassim Nasir, and Manar Abu Talib. "Firmware Update Attacks and Security for IoT Devices: Survey". en. In: *Proceedings of the ArabWIC 6th Annual International Conference Research Track on - ArabWIC 2019*. Rabat, Morocco: ACM Press, 2019, pp. 1–6. ISBN: 978-1-4503-6089-0. DOI: 10.1145/3333165.3333169. URL: http://dl.acm.org/citation.cfm?doid=3333165.3333169 (visited on 05/05/2020).

[5] *Binwalk | Firmware Extraction*. en-US. URL: https://www.refirmlabs.com/binwalk/ (visited on 05/23/2020).

[6] *Bus Pirate - DP*. URL: http://dangerousprototypes.com/docs/Bus_Pirate (visited on 05/13/2020).

[7] Brian Clinton and Suresh Marisetty. *Arm - Webinar recording - How threat modelling can secure your next IoT product in 5 steps*. URL: https://pages.arm.com/threat-models-webinar-recording-typ.html (visited on 04/15/2020).

[8] Elvis Collado. *DEF CON 24 Internet of Things Village - Reversing and Exploiting Embedded Devices - YouTube*. URL: https://www.youtube.com/watch?v=r4XntiyXMnA (visited on 05/05/2020).

[9] *Datasheet - EEPROM Serial 512-Kb I2C*. en. URL: https://www.onsemi.com/pub/Collateral/CAT24C512-D.PDF (visited on 04/17/2020).

[10] *Datasheet - S25FL116K/S25FL132K/S25FL164K, 16-Mbit (2 Mbyte)/32-Mbit (4 Mbyte)/64-Mbit (8 Mbyte), 3.0 V, SPI Flash Memory*. en. URL: https://www.cypress.com/file/196886/download (visited on 04/17/2020).

[11] *DEF CON 24 - Hardware Hacking Village - Matt DuHarte - Basic Firmware Extraction - YouTube*. URL: https://www.youtube.com/watch?v=Kxvpbu9STU4 (visited on 05/05/2020).

[12] *DEF CON 24 - Jmaxxz - Backdooring the Frontdoor - YouTube*. URL: https://www.youtube.com/watch?v=MMB1CkZi6t4 (visited on 03/27/2020).

[13] *DEF CON 24 - Stephan Huber and Siegfried Rasthofer - Smartphone Antivirus and Apps*. URL: https://www.youtube.com/watch?v=_aLhKmIT-Fo&feature=youtu.be (visited on 05/24/2020).

[14] *DEF CON 26 - Dr Rasthofer and Panel - Worrisome Security Issues in Tracker Apps - YouTube*. URL: https://www.youtube.com/watch?v=e9PdX-NmCSg (visited on 02/26/2020).

[15] Elena Dubrova. *Physical Unclonable Functions*. Lecture. Stockholm, Feb. 2020.

[16] *Firmware Engineer job at Glue Home, London May 2020*. Library Catalog: workinstartups.com. URL: https://workinstartups.com/job-board/job/84707/firmware-engineer-at-glue-home/ (visited on 05/07/2020).

[17] *flashrom*. URL: https://flashrom.org/Flashrom (visited on 05/14/2020).

[18] Megan Fuller, Madeline Jenkins, and Katrine Tjølsen. "Security Analysis of the August Smart Lock". en. In: (), p. 17.

[19] *Glue Smart Lock SE Elektroniskt dörrlås - Smarta lås*. SV. URL: https://www.kjell.com/se/produkter/smarta-hem/smarta-las/glue-smart-lock-se-elektroniskt-dorrlas-p51230 (visited on 02/27/2020).

[20] Dharmik Gohel. *IoT Pen testing*. en. Jan. 2020. URL: https://medium.com/@xesey/iot-pen-testing-1c4849327499 (visited on 06/10/2020).

[21] *Google Home*. SV. URL: https://store.google.com/product/google_home (visited on 03/19/2020).

[22] Garrett M. Graff. "The Mirai Botnet Was Part of a College Student Minecraft Scheme". en. In: *Wired* (). ISSN: 1059-1028. URL: https://www.wired.com/story/mirai-botnet-minecraft-scam-brought-down-the-internet/ (visited on 04/24/2020).

[23] Aditya Gupta. *The IoT Hacker's Handbook A Practical Guide to Hacking the Internet of Things*. eng. 1st ed. 2019.. 2019. ISBN: 1-4842-4300-5.

[24] Aaron Guzman. *IoT Penetration Testing Cookbook*. eng. 1st ed.. 2017. ISBN: 9781787285170.

[25] *hexdump(1)*. URL: https://www.freebsd.org/cgi/man.cgi?query=hexdump&sektion=1 (visited on 06/11/2020).

[26] Grant Ho et al. "Smart Locks: Lessons for Securing Commodity Internet of Things Devices". In: *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*. ASIA CCS '16. Xi'an, China: Association for Computing Machinery, May 2016, pp. 461–472. ISBN: 978-1-4503-4233-9. DOI: 10.1145/2897845.2897886. URL: http://doi.org/10.1145/2897845.2897886 (visited on 03/30/2020).

[27] *HomePod*. en-US. URL: https://www.apple.com/homepod/ (visited on 03/19/2020).

[28] Allen D Householder et al. "The CERT Guide to Coordinated Vulnerability Disclosure". en. In: (), p. 121.

[29] ReFirm Labs. *ReFirm White Paper*. Tech. rep. 2019. URL: https://www.refirmlabs.com/wp-content/uploads/2019/04/ReFirm-White-Paper-Why-Firmware-Vulnerabilities-Matter.pdf (visited on 03/24/2020).

[30] Arm Ltd. *Architectures | Introduction to security*. en. Library Catalog: developer.arm.com. URL: https://developer.arm.com/architectures/learn-the-architecture/introduction-to-security/single-page (visited on 04/23/2020).

[31] Arm Ltd. *Layered Security for the Next 1 Trillion Devices*. en. Library Catalog: www.arm.com. URL: https://www.arm.com/solutions/security (visited on 04/24/2020).

[32] *Man-in-the-middle attack | OWASP*. en. URL: https://owasp.org/www-community/attacks/Man-in-the-middle_attack (visited on 05/05/2020).

[33] H. Mansor et al. "Don't Brick Your Car: Firmware Confidentiality and Rollback for Vehicles". In: *2015 10th International Conference on Availability, Reliability and Security*. 2015, pp. 139–148.

[34] *mitmproxy - an interactive HTTPS proxy*. URL: https://mitmproxy.org/ (visited on 05/12/2020).

[35] *MobSF/Mobile-Security-Framework-MobSF*. original-date: 2015-01-31T04:36:01Z. May 2020. URL: https://github.com/MobSF/Mobile-Security-Framework-MobSF (visited on 05/24/2020).

[36] *Nmap: the Network Mapper - Free Security Scanner*. URL: https://nmap.org/ (visited on 05/07/2020).

[37] *Open On-Chip Debugger*. en-US. URL: http://openocd.org/ (visited on 05/23/2020).

[38] *OpenOCD User's Guide: TAP Declaration*. URL: http://openocd.org/doc/html/TAP-Declaration.html (visited on 05/26/2020).

[39] OWASP. *OWASP-IoT-Top-10-2018-final.pdf*. URL: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf (visited on 04/29/2020).

[40] *Product Brief - NRF51822*. URL: https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF51822-product-brief.pdf?la=en&hash=A4B5A9AA6675A58F7B779AF81C860CD69EB867FD (visited on 04/17/2020).

[41] *Product Brief - NRF52832*. URL: https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52832-product-brief.pdf?la=en&hash=2F9D995F754BA2F2EA944A2C4351E682AB7CB0B9 (visited on 04/17/2020).

[42] Ameco Research. *Global Smart Lock Market Size With Exponential Cagr Forecast Till 2024*. URL: http://www.amecoresearch.com/market-report/global-smart-lock-market-emr-12281 (visited on 02/18/2020).

[43] Jakob Rieck. "Attacks on Fitness Trackers Revisited: A Case-Study of Unfit Firmware Security". In: (2016).

[44] Riksdagsförvaltningen. *Brottsbalk (1962:700) Svensk författningssamling 1962:1962:700 t.o.m. SFS 2020:173 - Riksdagen*. SV. URL: https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/brottsbalk-1962700_sfs-1962-700 (visited on 05/24/2020).

[45] Riksdagsförvaltningen. *Lag (1960:729) om upphovsrätt till litterära och konstnärliga verk Svensk författningssamling 1960:1960:729 t.o.m. SFS 2018:1099 - Riksdagen*. SV. URL: https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/lag-1960729-om-upphovsratt-till-litterara-och_sfs-1960-729 (visited on 05/24/2020).

[46] Riksdagsförvaltningen. *Lag (2018:558) om företagshemligheter Svensk författningssamling 2018:2018:558 - Riksdagen*. SV. URL: https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/lag-2018558-om-foretagshemligheter_sfs-2018-558 (visited on 05/24/2020).

[47] Adam Shostack. *Threat Modeling, Designing For Security*. Indianapolis, Indiana: John Wiley & Sons, Inc., 2014. ISBN: 978-1-118-80999-9.

[48] *Smarta hem - Produkter*. SV. URL: https://www.kjell.com/se/produkter/smarta-hem (visited on 03/19/2020).

[49] *strings(1) - Linux man page*. URL: https://linux.die.net/man/1/strings (visited on 06/11/2020).

[50] *The Shikra*. en. URL: https://int3.cc/products/the-shikra (visited on 05/13/2020).

[51] Arvid Viderberg. "Security evaluation of smart door locks". sv. In: (), p. 77. URL: http://www.diva-portal.org/smash/get/diva2:1336796/FULLTEXT01.pdf.

[52] *Watch*. en-US. URL: https://www.apple.com/watch/ (visited on 03/19/2020).

[53] *Wireshark · Go Deep.* URL: https://www.wireshark.org/ (visited on 05/07/2020).

[54] *xxd(1): make hexdump/do reverse - Linux man page*. URL: https://linux.die.net/man/1/xxd (visited on 06/11/2020).

[55] Kim Zetter. "Hacker Lexicon: What Are White Hat, Gray Hat, and Black Hat Hackers?" In: *Wired* (Apr. 2016). ISSN: 1059-1028. URL: https://www.wired.com/2016/04/hacker-lexicon-white-hat-gray-hat-black-hat-hackers/ (visited on 05/24/2020).

Appendix A

Script for MITM

```
import json

def response(flow):
    if flow.request.url == "https://mobile.gluehome
       .com/api/v1/properties/":
        json_from_response = json.loads(flow.
           response.content)
        json_from_response['doors'][0]['locks'][0]['
           firmwareVersion'] = "-1000\u0000"
        flow.response.content = json.dumps(
           json_from_response).encode()

def request(flow):
        if flow.request.method == "PATCH":
            json_request = json.loads(flow.request.
               content)
            json_request['firmwareVersion'] =
               "-1555\u0000"
            flow.request.content = json.dumps(
               json_request).encode()
```

Appendix B

Script for reading EEPROM

```
"""python3 control of buspirate"""

import serial
import sys

BUSPIRATE_PORT = '/dev/ttyUSB0' #customize this!
   Find it in device manager.

def send(ser,cmd):
    """send the command and listen to the response
       ."""
    ser.write(str(cmd+'\n').encode('ascii')) # send
        our command
    for line in ser.readlines(): # while there's a
       response
        print(line.decode('utf-8').strip()) # show
           it

def readeep(ser, filename):
    """read eeprom addr 0xa from 0x00 to 65535"""
    newFile = open(filename, "wb")
    ser.write(str('[0xa0 0x00 [0xa1 r:65535]'+'\n')
       .encode('ascii')) # send read eeprom command
    for line in ser.readlines(): # while there's a
       response
        line = line.decode('utf-8').strip()
```

```
        if line.startswith("READ:"):
            line=line.split(" ",1)[1].replace(" ACK
               ", '')
            hexbytes = line.split(" ")
            length = len(hexbytes)
            toutf = []
            for i in range(length):
                toutf.append(int(hexbytes[i], 16))
            newFileByteArray = bytearray(toutf)
            newFile.write(newFileByteArray)

if len(sys.argv) != 2:
    print("Usage: python(3) i2creadlock.py [
       filename output]")
    sys.exit()
ser=serial.Serial(BUSPIRATE_PORT, 115200, timeout
   =1) # is com free?
assert ser.isOpen() #throw an exception if we aren'
   t connected
send(ser,'#') # reset bus pirate (slow, maybe not
   needed)
send(ser,'m') # change mode (goal is to get away
   from HiZ)
send(ser,'4') # mode 4 is i2c
send(ser,'4') # 400KHz
send(ser,'W') # turn power supply to ON. Lowercase
   w for OFF.
send(ser,'P') # pullup resistor ON
readeep(ser, sys.argv[1]) # read eeprom
print("Reading done. Disconnecting...")
ser.close() # disconnect so we can access it from
   another app
```

TRITA-EECS-EX-2020:259