EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP
STOCKHOLM, *SVERIGE 2019*

Finding Vulnerabilities in IoT Devices

Ethical Hacking of Electronic Locks

CHRISTOPHER ROBBERTS

JOACHIM TOFT

KTH
SKOLAN FÖR ELEKTROTEKNIK OCH DATAVETENSKAP

Finding Vulnerabilities in IoT Devices: Ethical Hacking of Electronic Locks

JOACHIM TOFT

CHRISTOPHER ROBBERTS

Bachelor in Computer Science
Date: June 20, 2019
Supervisor: Robert Lagerström
Examiner: Pontus Johnson
School of Electrical Engineering and Computer Science

Abstract

Internet of Things (IoT) devices are becoming more ubiquitous than ever before, and while security is not that important for every type of device, it is crucial for some. In this thesis, a widely available Bluetooth smart lock is examined through the lens of security. By using well-known attack methods, an attempt is made to exploit potential vulnerabilities in the system.

The researched lock was found to have design flaws that could be considered low-impact vulnerabilities, but using the system without these flaws in mind could lead to harmful outcomes for the lock owner.

Except for the design flaws, no real security problems were discovered, but the methods used in this thesis should be applicable for further IoT security research.

Keywords— ethical hacking; penetration testing; electronic locks; internet of things; iot; threat modeling; security

Sammanfattning

IoT-apparater blir allt mer vanliga i samhället. Det är inte ett krav för alla typer av apparater att ha stark säkerhet, men för vissa är det helt avgörande. I denna avhandling undersöks ett allmänt tillgängligt Bluetooth-smartlås utifrån ett säkerhetsperspektiv. Genom att använda välkända angreppsmetoder görs det ett försök att utnyttja potentiella sårbarheter i systemet.

Låset visade sig ha designfel som skulle kunna betraktas som sårbarheter med låg hotnivå, men att använda systemet utan dessa designfel i åtanke skulle kunna leda till farliga påföljder för låsägaren.

Förutom designfelen upptäcktes inga riktiga säkerhetsproblem, men metoderna som används i denna avhandling bör vara tillämpliga för ytterligare säkerhetsforskning inom IoT.

Nyckelord— etisk hacking; penetrationstestning; elektroniska lås; sakernas internet; iot; hotmodellering; säkerhet

Contents

Glossary

APK Android Package.

attack vector The means by which an malicious actor can perform an attack on a system.

black-hat hackers A term used to described a hacker with malicious intent.

BLE Bluetooth Low Energy.

GATT Generic Attribute Profile.

gattacker A command-line tool used to perform Bluetooth Low Energy man-in-the-middle attacks.

Ghidra A software reverse engineering suite of tools developed by the National Security Agency..

IoT Internet of Things.

MAC address A unique device identifier.

MITM man-in-the-middle.

Chapter 1

Introduction

IoT are physical devices that have the capability to in some shape or form communicate via the internet. According to Ericsson's Internet of Things forecast, around 18 billion IoT-related connected devices are forecast by 2022[1].

New types of connected devices are invented every day, and most of the large established actors in tech–in addition to a lot of new smaller actors–have recently entered the IoT market.

IoT devices are often equipped with microphones or cameras, and in addition to that, many are waiting continually for user input via sound or video. This fact makes security an essential part of ensuring the privacy of those that use them. If the security is lacking and an attacker can get access to the device, they could possibly eavesdrop on all communication around the device or gain remote access. News articles mentioning security breaches like this are common, and the impacted devices include everything from smart TVs[2] to drones[3], which underlines the necessity of security in the field. It is therefore of importance that the producers of IoT devices spend considerable effort to secure their products, and it is especially important when it comes to devices that ensure the safety of our private belongings, e.g., smart locks.

The thesis will make mention of the term penetration testing throughout the

[1] *Internet of Things forecast – Ericsson Mobility Report*. Nov. 2018. URL: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast.

[2] *SUPRA Smart TV Flaw Lets Attackers Hijack Screens With Any Video*. June 2019. URL: https://thehackernews.com/2019/06/supra-smart-tv-hack.html.

[3] Staff Report. *Teen's drone hack exposes poor security controls for Internet of Things devices*. May 2019. URL: https://gulfnews.com/uae/teens-drone-hack-exposes-poor-security-controls-for-internet-of-things-devices-1.63795510.

entirety of the thesis, which is "a controlled attack simulation that helps identify susceptibility to application, network, and operating system breaches."[4]

1.1 The security of smart locks

The increase of remotely managed locks in today's society can be found everywhere, ranging from vehicle locks to home locks. It is therefore vital that these devices are adequately tested for vulnerabilities for preventing malicious actors from gaining access to the private belonging of the smart lock owner. These locks are a respectable threat to the integrity of the owner if the lock is easily compromised.

Security involving our day to day life in this day and age of digitization should be at the highest level of priority when we depend on a large amount on said devices. Therefore a question arises: How do we avoid a tragic event, such as a robbery, due to inadequate security prioritization from the electronic lock creators? Also, will we ever be able to achieve a degree of certainty that our IoT devices we depend on will ever be wholly tested and secure? The answer to the second question is most likely no, but this does not mean we should not strive towards a safer and more trustworthy digital environment, making it as difficult and cumbersome as possible for black-hat hackers and criminals to compromise personal data and physical belongings.

1.2 The Lock

The vendors of the electronic lock in the focus of our study does also provide a mobile application that is used to control the lock via Bluetooth Low Energy (BLE).

The mobile application offers functionality regarding access to the lock. The owner can decide to gift a permanent key at any given time and revoke said key whenever. If the owner wishes to grant somebody a key for a limited amount of time, the owner could do this by setting a time restriction on the given key, releasing him/her from the responsibility of revoking lock access when needed. The mobile application also provides user log documentation to view lock activity.

[4] *Penetration Testing*. July 2018. URL: https://www.doi.gov/ocio/customers/penetration-testing.

1.3 Problem

Does the lock contain security vulnerabilities, and if it does, what are these, and what are the potential consequences?

1.4 Goals

The main goal of this project is to produce a collection of possible threats and vulnerabilities to the lock in addition to provide suggestions in order to improve the security of the product.

If a vulnerability is discovered and proves to be exploitable, the study aims to provide suitable countermeasures to smart lock vendors and owners of the lock. If the producer of the lock does not plan to fix any issues reported, the vulnerabilities will be disclosed publicly as discussed in section 1.6.

Furthermore, this study aims to inspire and deliver insight for readers to pursue further researching based on the information provided by this thesis.

1.5 Purpose

The purpose of this project is to provide information that can improve the state of security in IoT devices in general, and in smart locks specifically. The hope is that a more thorough investigation of a single product, in this case the lock, can lead to insights that can be applied more generally within the IoT industry.

1.6 Ethical considerations

Penetration testing is at its core testing and needs to be done with the consent of the client, which commonly is the company who's product is being targeted. Regarding the consumer as the client gets around many of the usual ethical dilemmas facing controlled penetration testing. Pierce, Jones, and Warren [5] presents 5 themes of ethical concern regarding penetration testing–Serve and Protect the Customer, Uphold the Security Profession, Avoid Conflicts of Interest, Avoid False Positives and False Negatives and Binding Ethics Legally.

If we regard the consumer as our customer, most of these concerns are not applicable or not a problem. The most important consideration is avoiding false positives and false negatives, which we take the utmost care to avoid in this paper. Any potential vulnerabilities will be reported as matter-of-fact as possible, describing the exact steps taken to reproduce them. We will not make

any judgments of the product as a whole, only on the exact things we find. We will also be careful to point out that a failure to find a vulnerability does not necessarily mean that the product is secure, it only means that taking the steps that we have taken to exploit the product does not lead to a vulnerability.

A central part of hacking "ethically" despite the lack of written consent is responsible disclosure. Responsible disclosure means that the producer that has a vulnerable product gets a period of time to fix the vulnerability before it is disclosed to the general public. The authors of this thesis will take into consideration the guidelines[5] from OWASP (Open Web Application Security Project) if a vulnerability is found, and any public disclosure will be in collaboration with the thesis supervisor.

[5]Owasp. *OWASP/CheatSheetSeries*. URL: https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Vulnerability_Disclosure_Cheat_Sheet.md.

Chapter 2

Threat modeling

According to Xiong and Lagerström [7], "the definitions of threat modeling are numerous, and used in many different and perhaps also incompatible ways." With that said, most definitions include the identification of possible threats in addition to some kind of analysis, classification or rating of these threats. The threat modeling methodology used in this paper is based on the steps outlined by Guzman and Gupta [8].

Threat modeling is generally the first step in any penetration testing project. It includes gathering information about the system, modeling this information into a comprehensive model and analyzing possible threats existing therein. Threat modeling is a key part of penetration testing, as it guides the approach for the rest of the project. If one were to skip this stage, the exploitation attempted would be almost arbitrary as it is not guided by any deep insight into how the system works, and no analysis of which parts of the system are likely to contain vulnerabilities.

2.1 Identifying and rating threats

The model of threats used in this thesis is STRIDE, which stands for Spoofing of user identity, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege [9]. Looking at each category of threat, an analysis of the device with regard to that category is performed and documented.

After the threats has been documented, each threat is assigned a risk rating, following the DREAD rating system. The DREAD rating considers the following categories [10]:

- Damage potential

Table 2.1: Threat 1: Unauthorized Lock Access

Threat description	Attacker could unlock the lock without permission from the owner
Threat target	Lock customers, Lock Bluetooth, Lock applications
Attack techniques	Attacker could perform a man-in-the-middle attack and record and later replay the data sent (replay attack). An attacker could also follow the user, pretend to be the lock, and when the customer's application connects forward Bluetooth traffic to another attacker that is nearby the lock. An attacker could circumvent the time limits granted by the lock owner to gain access.
Countermeasures	All communication is encrypted with a secure encryption algorithm.
Risk rating (DREAD)	
Damage potential	3
Reproducibility	2
Exploitability	2
Affected users	2
Discoverability	2
Total	11
Risk rating score: Medium	

- Reproducibility
- Exploitability
- Affected users
- Discoverability

The main threats with their associated risk rating can be seen in Table 2.1-2.3. In addition, a risk rating comparison can be seen in Table 2.4.

Table 2.2: Threat 2: Avoidance of Logging

Threat description	Attacker could unlock the lock (with permission) without the action being logged (recorded)
Threat target	Lock customers, Lock Bluetooth, Lock applications
Attack techniques	Attacker could use a custom application that does not send logs to the lock services. Attacker could turn off internet access for the application while unlocking the lock.
Countermeasures	All locks/unlocks are saved in the memory of the lock itself, and sent to the owner the next time they connect to the lock via Bluetooth.
Risk rating (DREAD)	
Damage potential	2
Reproducibility	3
Exploitability	1
Affected users	2
Discoverability	3
Total	9
Risk rating score: Medium	

Table 2.3: Threat 3: Denial of Service

Threat description	Attacker could deny any access to the lock
Threat target	Lock customers, Lock Bluetooth, Lock applications
Attack techniques	Attacker could connect to the lock via BLE and keep the lock busy indefinitely.
Countermeasures	Lock automatically disconnects from a connected device after a while.
Risk rating (DREAD)	
Damage potential	1
Reproducibility	1
Exploitability	3
Affected users	3
Discoverability	3
Total	11
Risk rating score: Medium	

Table 2.4: Threat risk rating comparison

	Unauthorized Lock Access	Avoidance of Logging	DoS
Damage potential	3	2	1
Reproducibility	2	3	1
Exploitability	2	1	3
Affected users	2	2	3
Discoverability	2	3	3
Total	11	9	11

2.2 Conclusion

By analyzing the potential threats posed to the lock and their respective risk rating, as portrayed in the tables above, the study gained a satisfying overview of the problems to research and prioritize. This laid path for finding and utilizing applicable threat exploitation methods.

Chapter 3

Threat exploitation methods

Using the information acquired from the threat modeling, a number of attack techniques were chosen.

3.1 Bluetooth man-in-the-middle attack

Man-in-the-middle (MITM) attacks have two major forms: eavesdropping and manipulation. [11] Eavesdropping provides communication data that can be analyzed in order to better understand the encryption methods used, while manipulation enables replay attacks and data fuzzing.

To connect via Bluetooth to the lock the mobile app does not scan for a specific MAC address–even after pairing once–it instead scans for a device which advertising data corresponds to the lock serial number, i.e., the lock advertises its serial number. When connected, all outgoing communication is done through writing to one specific characteristic in the lock Bluetooth Generic Attribute Profile (GATT) server; all incoming data is received through characteristic notifications.

To perform a MITM attack, we first do a Bluetooth scan using gattacker, saving the MAC address-addresses of all devices we find. Next, we examine the advertising data to find the correct device. When we have the correct MAC address-address for the lock, we can perform a discovery process of all of the locks services and characteristics, which is also saved for later. With both a copy of the device's advertising data and GATT services, we can mimic the lock on a Bluetooth capable device, while connecting to the lock with another.

With this setup, we can trick the target device to connect to our mimicked version ("peripheral") instead of the original lock, and forward all data to the lock via our "central" device. This allows us to read all data between the appli-

cation and the lock, although this is already possible by passive eavesdropping. The real strength is being able to fuzz the data being sent, allowing us to try different fuzzing to see if we can find any vulnerabilities. This setup also allows one to save the data sent from the application, which can be used to perform a replay attack.

The prerequisites for executing the attack with gattacker was access to two Bluetooth-capable Unix-based machines (where one machine is acting as the peripheral device and the other as the central device).

The MITM attack will aim to answer:

- Is the Bluetooth communication properly encrypted?
- Is it possible to replay communication in order to repeat previous actions, e.g. lock or unlock?
- Can the data somehow be fuzzed, to produce unexpectedly or insecure behavior?

3.2 Design flaws

Since the mobile application is responsible for receiving and sending access keys and reporting lock access to the vendor server, internet connectivity is a must for these functions to work. Because of this, restricting internet access in key moments of the interaction with the lock could lead to undesired outcomes (vulnerabilities).

A few test scenarios were devised to answer the following questions:

- Can time limits be circumvented?
- Can the recording of access logs be prevented?
- Can "remove access" be used when the target is not connected to the internet?

All of the scenarios below have been performed using two variations, one where Alice is within communication distance of the lock, and one where she is not. This is done because the results could be dependent on whether Alice can send updates for the lock to process, or whether this communication is impossible due to distance.

Scenario 1

1. Alice gives Chuck a time-restricted digital key to her lock
2. Chuck opens his application which now has access to the lock
3. Chuck turns of his internet connectivity
4. Chuck tries to open the lock after the access time has expired

This scenario aims to answer the first question about time limits. If the application, in combination with the vendor server, is solely responsible for enforcing time limits, something as simple as turning off the internet on the device running the app could prove enough to circumvent these limits.

Scenario 2

1. Alice gives Chuck a digital key to her lock
2. Chuck opens his application which now has access to the lock
3. Chuck turns of his internet connectivity
4. Chuck opens the lock and waits 5 minutes
5. Chuck turns on his internet connectivity

The second scenario aims to answer the second question about logging. Because the logging is dependent on the vendor server and the lock has no internet connectivity itself, one would expect that no logging would be made if Chuck's app has no connectivity. Despite this, one could possibly expect the lock to keep track of the access patterns and transfer these the next time Alice communicates with the lock, in order to not lose any history. It would also be possible for Chuck's app to queue the access logs transfer for internet connectivity comes back, and try to send them later.

Scenario 3

1. Alice gives Chuck a digital key to her lock
2. Chuck opens his application which now has access to the lock
3. Chuck turns of his internet connectivity
4. Alice revokes Chuck's access

5. Chuck tries to open the lock

This scenario aims to answer questions about the ambiguous "remove access" option available in the app. How, if at all, does the lock know that Chuck's access has been revoked when it can only communicate via Bluetooth, and there is no guarantee that Alice–who did the revoking–will interact with the lock before Chuck does. Does the lock even start to block access to Chuck after its subsequent interaction with Alice, or does it see Chuck's key as valid forever?

3.3 Mobile application analysis/tampering

To get a greater understanding of the attack techniques possible against the mobile application, it was decided that a reverse engineering analysis should be performed. Due to the difficulty of analyzing iOS applications, the lock Android app was chosen as the target. Ghidra, apktool[1] and jadx[2] was chosen as prospective tools to perform the analysis.

The analysis will aim to answer:

- Can the app be modified in order to introduce vulnerabilities?
- Does the source reveal any security flaws or information that could be used by an attacker?

[1] https://github.com/iBotPeaches/Apktool
[2] https://github.com/skylot/jadx

Chapter 4

Results

4.1 Bluetooth man-in-the-middle attack

gattacker was used successfully to intercept Bluetooth communication, perform a replay attack, and to fuzz data. Below follows the results.

- Is the Bluetooth communication properly encrypted?

Viewing the traffic between the lock and the application did not reveal any information in plain text. However, studying a byte representation of the data closer, a general message structure revealed itself; as seen in the excerpt in Figure 4.1 every message starts with *c0 00 f1* and ends with *c1*. The serial number of the lock used in the study (*d3 dc 64 38 44 55*) can also be seen in every message in the same place.

About 50 messages were communicated per each lock interaction, where about half were sent from the application to lock, and the rest received from the lock via BLE notifications. Albeit most of the messages were the same length, a few of the messages sent in each interaction were about 2-3 times longer.

```
c0 00 f1 d3 dc 64 38 44 55 9c cb 43 7f 01 00 00 2b e6 16 1c c1
c0 00 f1 d3 dc 64 38 44 55 9c cb 43 7f 01 00 00 2b f6 27 0e c1
c0 00 f1 d3 dc 64 38 44 55 9c cb 43 7f 01 00 00 2b 06 38 e1 c1
c0 00 f1 d3 dc 64 38 44 55 9c cb 43 7f 01 00 00 2b 16 09 f3 c1
```

Figure 4.1: An excerpt of the Bluetooth data sent between the mobile application and the lock during a lock operation. Each hexadecimal pair represents one byte of data.

Although we found a general pattern for each message, it did not reveal any more in-depth insights into the encryption methods used.

- Is it possible to replay communication in order to repeat previous actions, e.g. lock or unlock?

By referencing the saved data from a previous exchange between the application and the lock, where both unlock and lock operation were performed, a replay attack was conducted. This produced no discernible effects on the lock.

- Can the data somehow be fuzzed, to produce unexpectedly or insecure behavior?

Using a gattacker hook function, which allows changing write values to specific GATT characteristics, different bytes of the Bluetooth messages were changed. This made the lock unresponsive–the fuzzing did not provide any visible results except for that.

4.2 Design flaws

To recap, each of the below scenarios was performed in two variations, one where Alice it within communication distance of the lock, and one where she is not.

Scenario 1

1. Alice gives Chuck a time-restricted digital key to her lock
2. Chuck opens his application which now has access to the lock
3. Chuck turns of his internet connectivity
4. Chuck tries to open the lock after the access time has expired

The tests of the scenario revealed that Chuck is not able to open the lock in any of the two variations, suggesting that the time limits are enforced adequately despite the lack of internet connectivity.

Scenario 2

1. Alice gives Chuck a digital key to her lock
2. Chuck opens his application which now has access to the lock
3. Chuck turns of his internet connectivity
4. Chuck opens the lock and waits 5 minutes
5. Chuck turns on his internet connectivity

The result of this scenario was that no logs were sent in either variation, suggesting that the lock does not keep any internal history of the commands given to it, nor does the lock app keep track of logs that could not be transferred due to connectivity in order to resend later.

Scenario 3

1. Alice gives Chuck a digital key to her lock
2. Chuck opens his application which now has access to the lock
3. Chuck turns of his internet connectivity
4. Alice revokes Chuck's access
5. Chuck tries to open the lock

This scenario proved to be successful. After Chuck removes his internet connection and Alice revokes Chuck's lock permissions while disconnected, Chuck can lock and unlock the lock as long as he is disconnected from the internet.

4.3 Mobile application

- Can the app be modified in order to introduce vulnerabilities?

Android apps are packaged in a format called Android Package (APK), which is a zip file that contains the files needed to run the app. The files within the APK are compressed, but can easily be decompressed. In this project, apktool was used for this task, which allows decompression and recompression of APK files. apktool produces a list of smali files corresponding to each Java

class that the program was compiled from. These files can easily be decompiled using e.g. the tool smali2java.

Using the Java files as a reference, changes can be made to the smali-files and then be recompiled using apktool.

Although no direct modifications were found that could compromise the security of the application, turning on various debug logging messages was possible. In addition to that, adding custom logging for Bluetooth communication also proved to be viable. This suggests this approach can be effectively used to aid in understanding the IoT device one is investigating.

- Does the source reveal any security flaws or information that could be used by an attacker?

Reverse engineering the Android application and its native libraries did not reveal any code indicating a vulnerability flaw. The only finding the study came across was a string constant that seemed to be the encryption key for a database password encryption; this was not further pursued due to time constraints.

Chapter 5

Discussion

5.1 Bluetooth man-in-the-middle attack

The vendors of the lock promised that the lock uses an advanced encryption method for transferring data (AES 256). Even if this is the case, it is important for vendors to know that no matter how well encrypted the data is, it is of little significance if the same encrypted messages could be replayed back to the IoT device.

To confirm the truthfulness regarding the vendor's encryption statement and its efficiency, the study aimed at attempting to reverse the mobile application binary using suitable reverse engineering tools to find potential encryption algorithms. In retrospect, this could have been done in a more structured manner since it was very time-consuming. The study would have benefited greatly in regards to the time limit if this section had not been tinged by unstable structure, due to insufficient knowledge surrounding the topic. This time period of the study could be viewed as a trade-off between time and acquiring competence surrounding reverse engineering, which could be seen as either a negative or a positive fact although only unclear results where attained form the research.

When viewing the transferred data between the lock and the mobile application, it was clear that the data was encrypted although many interesting patterns were noted in the encrypted messages. This led the study in the direction of trying to understand the encryption methods used, which proved to be very time-consuming. The messages being sent to the lock were almost identical, differing with three "random" consecutive bytes.

When performing the replay attack, it proved to be unsuccessful as stated in the result section. This led to the suspicion that the lock expected to receive

different initialization data within every distinct handshake, perhaps the current time of day embedded in the encryption; or merely the fact that messages may be dependant on unique data delivered back from the lock, rendering the replay attack useless. This proves to be a secure and efficient strategy for circumventing replay attacks in this case.

5.2 Design flaws

In conclusion, the results show that the "remove access" option in the product, in addition to the access logs, cannot be trusted.

Regarding the logging, the effects of this attack are limited, as it does not allow an attacker without the proper permission from the owner to access the locks. Despite this, it can be problematic in cases where access is used as proof that someone has accessed the lock. One example is hospitality services, e.g., Airbnb, where smart locks are often used to provide easy access to lodging. If the access logs are seen as a source of truth, an attacker could enter the lodging preventing the access from being logged, and then claiming they were not ever there.

Because the logs are unreliable, it is questionable if they provide any utility at all. On the contrary, because this fact is not communicated in the application or the manual, the logs could provide the owner with a false sense of trust and lead to confusion, in addition to the potential of causing real damage.

The "remove access" option in the application is outright misleading, as the only enforcement of this revocation is through a web request from the app of the key holder being revoked. Because the revocation is not enforced at a lock level at all, no guarantee can ever be made that the access has been revoked. An attacker could turn off their internet access on their device, modify the application, or even record the request from the vendor server to save the access key to the lock.

From the lock's perspective, a permanent key is always permanent, completely disregarding the "remove access" option in the app.

Lastly, the tests to circumvent time restrictions were ineffective, which suggests that the time restrictions are encoded in the access key, and enforced on a lock level.

5.3 Mobile application analysis/tampering

The application analysis did as mentioned in the results section, not give any concrete results. Despite this, it was an invaluable tool to get a better understanding of the IoT system as a whole.

5.4 Problems and constraints

At the beginning of the hacking phase of the project much focus was laid on the reverse engineering of the Android application; this was because the Bluetooth encryption was vulnerable, that would make the lock profoundly insecure. In the end, nothing important was found relating to the BLE communication, and the investigation of the application proved to be larger in scope than first thought. At this stage, one could argue that too much time was spent trying to understand the encryption process instead of focusing on other possible attack vectors.

The choice of the lock–with no open source code, and as later revealed, encrypted firmware–as the testing subject could also be criticized, as understanding a closed-source code base is a massively time-consuming task. With the limited time constraints of the project, a more thorough test could have been performed if another product were chosen.

Another thing is the choice of attack vectors. If only one attack vector were to be chosen, e.g., Bluetooth or the Android application, the research needed to be undertaken to exploit the system properly could have been reduced, leading to that more practical tests could have been performed.

Chapter 6

Conclusions

In conclusion, the lock seems to be well protected against the attack vectors the study chose to examine. The vendors seems to have done a good work regarding Bluetooth communications, making it difficult for an attacker to understand the communication being sent by the lock with the mobile application and vice versa. When it comes to the mobile application, it proved to be very easy to attain the source code for the application with original function names; reverse engineering could be made more difficult by obfuscating the code using a product such as ProGuard[1].

Given a longer time period for the study, more attack vectors could have been examined, and the ones covered could have been analyzed further and reiterated. Having said this, the study does not indicate that the lock is an entirely secure device by any means, only the fact that the chosen attack vectors were covered using the methods described in this thesis.

[1] *ProGuard*. May 2019. URL: https://www.guardsquare.com/en/products/proguard.

Bibliography

[5] Justin Pierce, Ashley Jones, and Matthew Warren. "Penetration Testing Professional Ethics: a conceptual model and taxonomy". In: *Australasian Journal of Information Systems* 13.2 (2006). ISSN: 1449-8618. DOI: 10.3127/ajis.v13i2.52. URL: https://journal.acs.org.au/index.php/ajis/article/view/52.

[7] Wenjun Xiong and Robert Lagerström. "Threat modeling – A systematic literature review". In: *Computers Security* 84 (2019), pp. 53–69. DOI: 10.1016/j.cose.2019.03.010.

[8] Aaron Guzman and Aditya Gupta. *IoT penetration testing cookbook: identify vulnerabilities and secure your smart devices*. Packt, 2017.

[9] Zhimin Yang and Zengguang Zhang. "The Study on Resolutions of STRIDE Threat Model". In: *2007 First IEEE International Symposium on Information Technologies and Applications in Education* (2007). DOI: 10.1109/isitae.2007.4409285.

[10] Sonia, Archana Singhal, and Hema Banati. *Fuzzy Logic Approach for Threat Prioritization in Agile Security Framework using DREAD Model*. 2013. arXiv: 1312.6836 [cs.SE].

[11] Bruce Potter and Bob Fleck. *802.11 security*. OReilly, 2003.

TRITA-EECS-EX-2019:311