KTH ROYAL INSTITUTE
OF TECHNOLOGY

Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Ethical Hacking of a Ring Doorbell

ARNAR PÉTURSSON

Stockholm, Sweden, 2023

Ethical Hacking of a Ring Doorbell

ARNAR PÉTURSSON

Master's Programme, Computer Science, 120 credits
Date: January 26, 2023

Supervisor: Pontus Johnson
Examiner: Gerald Q. Maguire Jr.
School of Electrical Engineering and Computer Science
Swedish title: Etisk Hacking av Ring Dörrklockan

Abstract

Internet of Things (IoT) devices have entered the home, offices, and other parts of everyday life at an alarming rate in recent years. A large company in the IoT space is Ring. They mostly make smart home security devices, with the mission of making neighborhoods safer, and their flagship product is the Ring doorbell. However, an important question is: Does connecting such a device to your home network introduce new threats?

In this thesis, the security of the Ring doorbell and applications in its ecosystem are evaluated and presented. A structured approach is taken for the evaluation by constructing a threat model, evaluating threats, and further pursuing selected threats by penetration testing. One minor vulnerability was discovered but the overall conclusion was that this product followed best practices when it comes to securing IoT devices, reducing attack surfaces, and enforcing user security policies. The conclusion of this thesis is that adding a Ring doorbell to your home network does *not* introduce a new threat.

Keywords

Security, Ethical hacking, Penetration testing, Internet of things, Threat modeling, Ring Doorbell.

Sammanfattning

Sakernas internet (IoT) enheter har kommit in i hemmet, kontoret och andra delar av vardagen i en alarmerande takt under de senaste åren. Ett stort företag inom IoT området är Ring. De tillverkar mestadels smarta hem säkerhetsenheter, med uppdraget att göra stadsdelar säkrare, och deras flaggskeppsprodukt är Ring-dörrklockan. Men införs nya hot om du ansluter en sådan enhet till ditt hemnätverk?

I detta examensarbete utvärderas och presenteras säkerheten för Ring dörrklockan och applikationer i dess ekosystem. Ett strukturerat tillvägagångssätt används för utvärderingen genom att konstruera en hotmodell, utvärdera hot och fortsätta vidare utvalda hot genom penetrationstester. En mindre sårbarhet men den övergripande slutsatesen var att den här produkten följde bästa praxis när det gäller att säkra IoT enheter, minska attackytor och upprätthålla användarsäkerhetspolicyer. Slutsatsen av denna avhandling är att om du lägger till en Ring-dörrklocka i ditt hemnätverk *inte* introducerar ett nytt säkerhetshot.

Nyckelord

Säkerhet, Etiskt hackande, Penetrationstestning, Sakernas internet, hotmodellering, Ring-dörrklockan.

Contents

List of Figures

List of Tables

Listings

List of acronyms and abbreviations

adb	Android Debug Bridge
AP	Access Point
API	Application Programming Interface
APK	Android Package
ARP	Address Resolution Protocol
CBC	Cipher block chaining
CCS	Common Command Set
CSS	Cascading Style Sheets
DDoS	Distributed Denial-of-Service
DNS	Domain Name Service
DOM	Document Object Model
DoS	Denial-of-Service
DTD	document type definition
ECB	Electronic codebook
EFF	Electronic Frontier Foundation
GUI	graphical user interface
HD	high definition
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IoT	Internet of Things
JS	JavaScript
JSON	JavaScript Object Notation
LAN	local area network
MAC	Message Authentication Code

MAC address	media access control address
MIME	Multipurpose Internet Mail Extensions
MITM	Man-in-the-middle
NIST	National Institute of Standards and Technology
NTP	Network Time Protocol
OS	operating system
OSINT	Open Source Intelligence
OTA	Over-the-air
OWASP	Open Web Application Security Project
PDF	Portable Document Format
PKCS	Public Key Cryptography Standards
PoC	Proof of Concept
POODLE	Padding Oracle On Downgraded Legacy Encryption
RTP	Real-time Transport Protocol
SAML	Security Assertion Markup Language
SDK	Software development kit
SIP	Session Initiation Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSID	Service Set Identifier
SSL	Secure Sockets Layer
SSO	single sign on
SSRF	Server-side request forgery
TCP	Transport Control Protocol
TLS	Transport Layer Security
UAF	Universal Authentication Framework
UART	universal asynchronous receiver-transmitter
UDP	User Datagram Protocol
UI	User Interface
UID	Unique Identifier
URI	Uniform Resource Identifier

URL	Uniform Resource Locator
Wi-Fi	Wireless Fidelity
XML	extensible markup language
XSS	Cross-Site Scripting
XXE	XML External Entity

Chapter 1

Introduction

The market for Internet of Things (IoT) is enormous and growing fast. Cisco reports that the number of active connected IoT devices will go from 6.1 billion devices in 2018 to 14.7 billion devices in 2023 [1], while a newer report from IoT-analytics says that actively connected IoT devices where already 11.7 billion devices in 2020 and will exceed 30 billion devices in 2025 [2]. The growth and continuing growth predictions of the IoT can be attributed to many factors, such as great hardware advancements [3] and big data analytics becoming a popular research topic with big data analytic solutions addressing the vast amounts of data being collected by IoT devices [4, 5, 6], and networks have become faster and are expected to continue to improve with the introduction of 5G networks [7, 8]. With the IoT market growing at this incredible rate and showing no signs of slowing down, it is important to take a step back and consider the security risks involved with the introduction of these large numbers of devices.

IoT devices have gained a lot of popularity with consumers, mainly because they offer smart solutions to diverse problems and often automate mundane tasks. An example of this are internet-connected coffee makers that can start brewing when a user gives the command from their mobile application, saving them the extra trip to the kitchen and minimizing waiting time in the kitchen. Connecting the coffee maker to the internet and the extra features this provides might sound appealing to non-tech-savvy consumers, but consumers with more security awareness are likely to see this as a red flag and realize the risks involved might outweigh the benefits.

The IoT market is very competitive, with many companies developing low-cost devices to appeal to consumers as well as rushing devices to market to beat competing companies. The problem with rushed and low-cost devices is that they often come with the cost of neglected security. In the past, IoT devices with insufficient security have been a major problem, with coffee makers being exploited in a demand for ransom [9], home cameras hacked and footage leaked [10], and devices being recruited to botnets and then used to perform Distributed Denial-of-Service (DDoS) attacks [11].

The purpose of this thesis is to demonstrate how the security of IoT devices can be evaluated, using a specific device as an example. The selected device is the Ring Doorbell 3 Plus, which has 2-way audio and a 1080p high definition (HD) camera with night vision and motion detection. This device is compatible with other smart devices, such as Amazon Alexa and a variety of smart locks. Users can control the device from Apple iOS and Android mobile applications, as well as via a web application. A threat model was constructed of the system to identify and evaluate threats (see Chapter 3), and some of those threats were then selected for further penetration tests (see Chapter 4). This thesis should provide readers with the necessary methods and techniques to conduct their own security evaluation and help them become more security aware, as well as provide a thorough evaluation of the Ring device that has been examined in this thesis.

1.1 Ethical Hacking

Ethical hacking will be used to evaluate the security of the Ring Doorbell 3 Plus in this thesis. The role of ethical hackers is to attempt to exploit systems with the goal of discovering vulnerabilities and responsibly disclosing them. Ethical hacking is often used interchangeably with penetration testing [12], while in other instances ethical hacking is used as an umbrella term to define all attack vectors, hacking methods, techniques and tools while penetration testing is a subset of ethical hacking where only a specific part of a system is in the scope of the test [13, 14]. The latter definition will be used in this thesis so that ethical hacking refers to the *overall assessment of the device* and penetration tests will be performed *on specific parts of the system*.

Ethical hackers are often referred to as *white hat* hackers to differentiate them from *black hat* hackers. The major difference between the two is intent. While white hat hackers act with the goal of improving the security of a system, black hats act with malicious intent. White hat hackers mostly perform the same activities and use the same tools as black hat hackers, this is precisely

the main benefit of hiring a white hat hacker, to *simulate* cyber attacks with the goal of making the system more secure against real-world attacks.

Motives of ethical hackers can differ. Companies can hire ethical hackers to directly asses their security or they might hire them indirectly by offering a bug bounty that rewards ethical hackers based on the severity of the discovered vulnerability. There are also ethical hackers that do not receive any compensation for their work, they simply do it because they enjoy finding vulnerabilities in systems or they might have a passion for protecting consumers or companies from malicious actors. For the purpose of this thesis, the motivational difference of ethical hackers is important since ethical hacking is not always done with a company's best interest in mind. An ethical hacker might discover that a company is breaching consumer trust by sharing or selling personal information illegally or unethically and choose to share that information with consumers, authorities, or other stakeholders.

In this report, ethical hacking is used to evaluate the security of the Ring Doorbell 3 Plus. Ring does not have an active bug bounty program but according to their website*, vulnerabilities should be reported to *security@ring.com*. An important factor of ethical hacking is to responsibly disclose vulnerabilities. As a vulnerability made public before the responsible company has a chance to assess and patch this vulnerability can have dire consequences. This does not apply to privacy violations on Ring's behalf to its users. In that case, the emphasis is on notifying the users first and the first step to do would be to publish any such findings within this project.

1.2 Background & Problem Statement

The IoT concerns a network of physical components. These components have Unique Identifiers (UIDs) and are embedded with sensors, software, and other technologies to collect and communicate data over the internet without human intermediaries [15, 16].

IoT devices have gotten a bad reputation over the last decade for being insecure. With frequent reports about IoT devices being insecure and stories gaining global coverage, such as, attackers talking to children through home cameras [17], attackers recruiting hundreds of thousands of IoT devices to botnets [18], and cars being vulnerable to attacks [19], consumers are already concerned when buying IoT devices [20]. Security awareness regarding IoT devices is not only a concern for consumers, connecting IoT devices to

*https://shop.ring.com/pages/security | Visited: 2022-06-6

corporate networks enlarges their attack vector. In a report published by Gartner it was predicted that by 2020, 25% of all enterprise attacks would involve IoT [21]. These are alarming predictions for companies, especially when the Coronavirus pandemic is considered, where a lot of people were forced or opted to work from home, where attackers can possibly gain access to corporate networks via employees' home networks.

Increased awareness will ensure more regulations regarding IoT security and these will force companies to focus more of their efforts and resources on security. There have been some positive advancements, for example, California passed a law in 2018 requiring IoT devices to have reasonable security features [22] and the UK has proposed legislation that would require IoT devices to have unique passwords not re-settable to universal factory settings, along with having available public points of contact to report vulnerabilities [23]. These are only two of the first steps toward better security. This thesis should contribute to increasing security awareness for IoT devices and help consumers and companies realize that extra steps might need to be taken when connecting IoT devices to their networks.

1.3 Scope and Goal

The objective of this thesis was to evaluate the security of the Ring Doorbell 3 Plus, using ethical hacking methods and techniques. Part of the project was to gather information, construct a threat model, realize the attack vectors, and identify potential threats and penetration test identified (potential) threats. This thesis should provide a guide as to how to evaluate the security of other IoT devices and more specifically serve as a security report on the Ring Doorbell 3 Plus device.

The goal of the thesis was to answer the research question:

How secure is the Ring doorbell 3 Plus?

To give better insight into how the question was answered, this question can be broken down into more detailed questions:

1. Have previously discovered vulnerabilities in the Ring Doorbell been fixed?

2. What vulnerabilities can be found in the Ring Doorbell 3 Plus and can they be exploited?

3. What are the consequences of successful exploitation of the vulnerabilities found?

4. Is Ring breaching consumer trust by sharing the consumer's data without their consent?

1.4 Method

A deeper dive into the methods used for this thesis work can be found in Chapter 2. An overview of the methods breaks down the work into 6 parts:

1. A literature review of threat modeling, what it is, and how it can be used to evaluate the security of a system.

2. Research was done to uncover common vulnerabilities of IoT devices along with what previous security concerns have been raised regarding the Ring doorbell and what vulnerabilities have been discovered.

3. A threat model for the Ring doorbell was constructed.

4. With this threat model, a vulnerability analysis was performed to determine what threats should be pursued.

5. Selected threats were explored and attempts were made to exploit them.

6. Discovered vulnerabilities were reported to Ring.

1.5 Previous Work

As previously mentioned the IoT market is enormous and it can be expected that there will be many devices similar to the Ring doorbell, such as Nest Hello, Arlo's Video Doorbell, and August View. Additionally, there are many IoT home cameras from these same companies and other vendors that are present in the IoT market. Many of these home cameras could be placed in the same category as doorbells when considering their functionality and usage. Taking a deep dive into the history of discovered vulnerabilities and security issues of all of these devices is out of the scope of this thesis, instead, when constructing the threat model and assessing threats to further pursue, the Open Web Application Security Project (OWASP) top ten lists were used along with a deep dive into previously discovered vulnerabilities in the Ring doorbell.

Ring was founded in 2013 and was acquired by Amazon in 2018. Amazon is one of the biggest companies in the world and they have high standards when it comes to security. Therefore, it is safe to assume that the Ring doorbell has been rigorously tested and a lot of time and resources spent on securing it. Nevertheless, quite a few vulnerabilities have been discovered in the Ring doorbells, both before and after the company's acquisition by Amazon.

1.5.1 Leaked Wi-Fi credentials during setup

Many IoT devices, including the Ring doorbell, need to be connected to Wireless Fidelity (Wi-Fi) to function properly. However, connecting to a Wi-Fi network can be a challenge for many of these IoT devices, such as the Ring doorbell, when there is no User Interface (UI). Users somehow need to communicate to the device, which Wi-Fi it should connect to, and provide it with the necessary credentials. Ring uses a popular method to tackle this problem. During setup, the wireless module in the device will go into Access Point (AP) mode and the mobile device that is being used for the setup will connect to that AP. Hereafter, this AP will be called the Ring AP. During this configuration step, there are no credentials needed to connect to the Ring AP, so any Wi-Fi device in range can connect to the device while it is acting as an AP. To complete the setup, the user selects the target Wi-Fi network from a list of Service Set Identifiers (SSIDs) in the Ring mobile application and then enters the credentials for that Wi-Fi network. The SSID and password are then transferred to the Ring device and now the Ring device connects to the selected Wi-Fi network.

Setup should only be required once and it is unrealistic for attackers to assume that they can be present or in range of the AP when that happens. Still, attackers can find a way around this problem, for example by forcing the Ring device off the network it is connected to by sending de-authentication messages that are spoofed, *i.e.,* seem to be coming from the router that the device is connected to. This will kick the device off the network and force the owner of the device to go through the setup steps again.

In 2016, Pen Test Partners noticed that the Gainspan wireless unit used by Ring, by default, has its admin graphical user interface (GUI) exposed and this had not been disabled by Ring developers [24]. As a result, anyone connected to the AP could visit `192.168.240.1` in their browser to get the settings GUI for the AP, and then they could navigate to */gainspan/system/config/network* and obtain the wireless network configurations, including the Wi-Fi SSID and password.

In 2019, Bitdefender discovered that when users are sharing their Wi-Fi credentials from their mobile application to the Ring device during setup mode, the data was being shared via HTTP; hence, it was unencrypted [25]. As connecting to the AP does *not* require credentials, attackers could easily listen in on the communications and read the credentials in plaintext.

1.5.2 Ring website vulnerable to XSS

In January 2017, an ethical hacker by the name *k0t* disclosed a Cross-Site Scripting (XSS) vulnerability on the Ring webpage to *openbugbounty.org* [26]. Ethical hackers do not always want to contact companies directly to disclose vulnerabilities and instead opt for platforms such as openbugbounty.org that help connect ethical hackers to website owners so that vulnerabilities can be disclosed in a transparent and mutually beneficial manner.

1.5.3 Outdated user sessions and credential stuffing

Not all Ring hacks can be traced back to bad implementations in their software, hardware, or data transfers. In some cases, users were not following best security practices or the recommended usage of the Ring products. In cases such as this, it can be hard to blame the insecurity of the product (or rather the lack of security), but such issues expose areas of a system that need tighter security.

The Ring doorbell is a doorbell, so household users might need to have access to it. Ring recommends that all users that need access to a device create their own Ring account with strong and unique passwords and the owner of the device can then share access to the device with the necessary accounts. Ring also recommends that users use 2-factor authentication when logging in. Contrary to the recommendations of Ring, many users chose to share the credentials of their accounts with other users, meaning that many users used the same accounts. Additionally, many users were guilty of reusing passwords from other services (*i.e.,* those not connected to Ring), as well as either choosing not to or not having the required knowledge to turn on 2-factor authentication for their accounts.

In 2018, users noticed that when they changed their passwords, sessions on that account were not ended [27]. This meant that users that were logged in could stay logged in and view the video feed from the camera and download videos even though the password for the account they were logged into had been changed.

In December 2019, reports came out that account information (such as email addresses, passwords, camera locations, and camera names) had been leaked online for more than 3000 Ring accounts [28]. Ring investigated and concluded that they had not been breached, rather than the attackers got and used credentials from another non-Ring service [29]. This type of attack is referred to as credential stuffing which occurs when attackers get a hold of compromised user credentials and use bots to attempt to log in to other services with those credentials. Since many users reuse their email addresses and passwords, these hacked users had gone against Ring recommendations regarding account sharing, password reuse, and 2-factor authentication. This and the lack of security on Ring's behalf resulted in a large number of compromised Ring accounts.

1.5.4 Insecure data transfers

The Ring doorbell handles sensitive data that needs to be securely transferred. The feed that is captured by the Ring doorbell will of course depend on where the device is placed, but in most cases, the device is placed by the front door of households; hence, it will capture the comings and goings of the residents and their guests. This information can be very sensitive and private for several reasons. Additionally, burglars looking for an empty home might use this information to target specific households. This data needs to be transferred to the owner of the doorbell for the doorbell to function properly; therefore, the integrity of that data transfer is an important security factor. If an attacker can spoof a video that is sent to a Ring user so that it looks as if someone that is not in front of the device is there, this could have serious consequences. An example might be a Ring user viewing the video feed coming from their doorbell and thinking that it is a friend or a family member and therefore choosing to open the door when in reality, the user just watched a video clip sent by an attacker and thus lets the attacker into the house.

In 2019, a vulnerability in the encryption of data being transferred to Ring applications was discovered [30, 31]. If an attacker was connected to the same network as a Ring user, the attacker could obtain the unencrypted video and audio stream being transferred to the Ring application. As a result, the attacker could spoof the video so that the user would receive a video from the attacker rather than from the Ring doorbell. This vulnerability compromised the confidentiality and integrity of the data and as a result, could have serious consequences when the data is as sensitive as the Ring doorbell data. Fortunately for Ring and its users, the vulnerability was discovered by

ethical hackers at the cyber security company BullGuard, so the vulnerability was reported and patched before the discovery of it was released.

1.5.5 Third-party trackers discovery

As previously discussed, consumers are not only concerned with the security of their devices relating to malicious actors but also with the handling of their private data by companies and services [32]. Consumer reports reveal that consumers are concerned with how companies use and share their data and consumers want more transparency about how data is handled and used [33, 34].

Considering these consumer concerns, an interesting discovery was made by the Electronic Frontier Foundation (EFF). During an investigation of the Ring Doorbell, EFF revealed that the Ring Doorbell was sending personal information (such as email addresses, full names, device information, mobile carrier, and more) to four different companies [35]. The companies receiving information were the deep linking solution company Branch, business analytics company Mixpanel, mobile marketing analytics company AppsFlyer, and social media giant Facebook. According to EFF, only MixPanel was mentioned briefly in Ring's list of third-party services, but to what extent MixPanel received data was not mentioned and the other three companies were *not* mentioned at all.

These third-party trackers were discovered when a Hypertext Transfer Protocol Secure (HTTPS) proxy server was set up to intercept the traffic coming from the Ring application. To allow active interception of secure traffic, a root certificate needs to be installed in the mobile operating system (OS) in use so that the proxy server is trusted. The researchers discovered that the certificate was not being trusted and concluded that the Ring application was using app-level certificate pinning which means that the Ring server only trusted certificates that were stored within the Ring application and not the ones installed in the mobile OS. This is done to secure the data being transferred, but a side effect was that consumers and security researchers could not view the data being transferred nor know to which destinations it is being sent. To circumvent this, the researchers injected code at runtime into the Ring application to ensure that the certificate installed on the mobile's OS would be accepted. This resulted in them seeing the data being transferred. This, of course, was not a vulnerability discovery but nevertheless, it indicates a breach of trust by Ring with regard to its consumers, especially as these third-party trackers were not mentioned or barely mentioned by Ring.

1.6 Structure of the thesis

- Chapter 2 describes methods used for information gathering and penetration testing.
- Chapter 3 outlines the information gathered and the constructed threat model.
- Chapter 4 describes penetration tests performed on the system, methods used, results and discussion.
- Chapter 5 gives a discussion of the result.
- Chapter 6 present the conclusions that were drawn, discusses limitations of this work, suggests ideas for future work, and gives some reflections on the project.

Chapter 2

Method

This chapter introduces and explains the methods that were used. The method used in this thesis is based on Weidman's seven steps to penetration testing as documented in her book, *Penetration testing: a hands-on introduction to hacking* [36]. Each of the steps are examined and the methods, techniques, and tools are introduced that will help achieve the goals of each step. Weidman uses the terms ethical hacking and penetration testing interchangeably, but according to the definition used in this thesis, what she is referring to are *the seven steps of ethical hacking*, rather than penetration testing.

2.1 Ethical Hacking

Ethical hacking is the practice of evaluating the security of a system. It involves discovering vulnerabilities as well as exploiting those vulnerabilities to assess an attacker's potential gain and provide a Proof of Concept (PoC) [36, 12]. Part of penetration testing is simulating real attacks. This simulation of attacks is done to secure the system against future attacks as well as provide a better understanding of the risks associated with successful attacks.

Ethical hacking can be broken down to smaller steps which can vary based on the objective, the system to test, or the tester's preferences. Engebretson defines four steps [12]: reconnaissance, scanning, exploitation, and maintaining access. He encourages testers to visualize the steps as an inverted triangle since the results from the first steps are broad but the steps become narrower with each step taken. The metaphor might not be directly applicable to all variations of ethical hacking methods, but the fundamental idea of it is. The tester will gather information, probe the system with the discovered information, and then drill down to specific parts of the system.

Another widely used method is Weidman's seven steps [36]:

1. **Pre-engagement:** involves coming to an agreement with the client regarding the goals and scope of the security assessment.
2. **Information Gathering:** The tester gatherers publicly available information about the system.
3. **Threat modeling:** Information is used to construct a threat model. Now the information be evaluated and the impact of potential attacks assessed.
4. **Vulnerability analysis:** With the information gathered and the constructed threat model, the tester tries to discover vulnerabilities in the system that could potentially be exploited.
5. **Exploitation:** In this step, attacks are attempted by exploiting the discovered vulnerabilities.
6. **Post Exploitation:** Successful exploits are used in an attempt to gather additional information, gain unauthorized access to the system, or retrieve sensitive data that has been exposed.
7. **Reporting:** The tester sums up his discoveries and distributes this information to appropriate parties.

For this thesis, Weidman's method will be used. However, the first step can be overlooked since the initiator of the penetration test is the thesis researcher and the test was not requested by Ring.

The remainder of this chapter will focus on each of the steps described by Weidman and introduce the necessary methods, techniques, and tools to achieve the goals of each step.

2.2 Information Gathering

The first step in conducting the security assessment was information gathering. This step lays the foundation for the following steps and can be an important factor in the eventual findings. An important aspect of information gathering is to differentiate valuable information from useless information. If too much information is gathered, then the data might become too noisy, making it hard to separate beneficial data from noise.

Information that might be valuable can be gathered from all directions, such as employees oversharing on social media, revealing employment ads, or public records. This information is publicly available and the process of gathering it is often referred to as Open Source Intelligence (OSINT) gathering. For this project, the official Ring website provided valuable information as well as the Ring doorbell 3 Plus manual. Following OSINT gathering, the device's ports were scanned.

2.2.1 Port Scanning

Valuable information can be gathered by scanning the ports of the device, such as, port states, active services, operating systems, and communication protocols. One of the widely used scanning tools is Nmap [37]. Nmap can be used to discover active hosts on a network and further probe that host for information on what ports are open, what services are running on which ports along with their versions, OSs, and the presence of a firewall.

Nmap offers different types of port scanning, varying in speed, obtrusiveness, and accuracy. Nmap divides ports into six different states [38] where the three most states common are:

Open The application running actively listens for and accepts Transport Control Protocol (TCP) connections or User Datagram Protocol (UDP) datagrams on the port. Open ports expand the attack vector and expose the system to possible attacks. System administrators try to keep ports closed or protect them with firewalls while avoiding hindering legitimate users, while hackers try to find open ports to exploit.

Closed There is no application listening on the port, but it is still accessible and will receive and respond to Nmap packets. Closed ports can be of use to attackers since they respond to probes and they might reveal that a host is running during host discovery and provide some information during OS detection.

Filtered Packet filtering, such as firewall or router rules, prevent Nmap packets from reaching the port; therefore, Nmap cannot determine whether the port is open or closed. Sometimes the filter responds with Internet Control Message Protocol (ICMP) error messages but more commonly the filter just drops the packages. This gives attackers and security researchers little information gained and

> further frustrates them because Nmap does not know whether the packets are being dropped by a filter or if the packet loss is due to network congestion. As a result, Nmap will re-send the packets several times which slows down the scan.

Port scans supported by Nmap include the TCP SYN scan, which is one of the more commonly used scan options. It works by sending a raw TCP SYN packet to ports on the target. The status of ports is determined by the port's response or lack of response. TCP SYN packets are sent to start a connection but Nmap never completes the connection process, this both improves the scan speed, but also makes the connection less likely to be flagged or logged [39].

When a Nmap user does not have the necessary privileges to send raw packets, they can opt for the TCP connect scan. Nmap then asks the OS it is running on to establish a connection instead of sending raw packets. In this type of scan the connection is completed which affects speed, since more packets need to be sent to retrieve the same information as in the TCP SYN scan but this also increases the likelihood that the target machine logs the connection, making detection more likely [39].

Nmap offers more types of TCP scans, but importantly offers other types of scans as well. Those include UDP scans, which in most cases send empty packets to target ports, except for common ports where protocol-specific payloads are sent, and then evaluates the response or lack of response. UDP scans can be a bit more challenging and time-consuming than TCP scans, due to lack of responses from open and filtered ports and closed ports responding with ICMP port unreachable errors. ICMP port unreachable errors are sometimes rate limited by hosts and to avoid flooding hosts, Nmap slows down, resulting in an even slower scan. This can be somewhat counteracted by scanning multiple hosts in parallel or instructing Nmap to skip slow hosts [39].

OS detection scan is another valuable scan that allows the tester to identify the OS on the target host. Nmap uses TCP/IP stack fingerprinting to detect the OS. A round of TCP and UDP packets are sent to the host and the responses are thoroughly examined. Nmap then compares the resulting fingerprint to its database of over 2600 recognized OS fingerprints and returns OS details about the closest match [40].

When devices are protected by a firewall and ports are reported as filtered, Nmap offers many optional techniques for bypassing the firewall to gain additional information. Most of these techniques have a low probability of succeeding and are mainly effective in combating firewalls that have been poorly configured. Nmap recommends trying as many techniques as possible to increase the likelihood of success [41, 42]. Techniques include sending

fragmented packets, packets with bad checksums, and using scripts such as firewalk. Hosts and ports can also be configured to only accepts requests from specific IP addresses and Nmap provides options to spoof such information.

2.2.2 Capturing APK files

Somewhat obfuscated source can be captured by reverse engineering Android applications, such as the Ring app [43].

The Android mobile phone used for testing during this project work was the *Samsung Galaxy S7 Edge*. This model does not have root privileges enabled by default. Root privileges are required to pull files from Android devices, so a new rooted boot image was downloaded and flashed to the Samsung device and root access enabled.

Android Debug Bridge (adb) is a command-line tool that enabled communications from a computer to a Android device [44]. With a rooted device and adb tools setup, a rooted shell can be opened up on the device and the its files captured.

By unzipping the captured Android Package (APK) file the *.dex* files that form it are revealed. They can be converted to *.jar* files by using *dex2jar* [45]. Stored within the *.jar* files are Java *.class* files which can be viewed by using a decompiler such as *JD-GUI* [46].

2.2.3 Bypassing Android application certificate pinning

As discussed in Section 1.5 researchers from EFF, while discovering third-party trackers, noticed that the Ring mobile application uses app-level certificate pinning. This technique can prevent Man-in-the-middle (MITM) attacks since the application only trusts Secure Sockets Layer (SSL) certificates stored within itself and drops connections if it is not communicating directly with a server that has a certificate that it trusts. A byproduct of this is that researchers cannot read unencrypted traffic between the application and the server.

A few attempts were made to bypass the application certificate pinning which included, rooting an Android device to gain admin privileges, injecting code to the running Ring application at run-time, capturing the application's APK files, and updating them but without success. Finally the website *apkpure* [47] was used which contains a database of applications' APK files. There the Ring application APK file was downloaded and it was used with the

apk-mitm tool [48], which decodes the file, updates configurations and source code to remove certificate pinning functionality, encodes the updated file and signs it so that it can be installed on an Android device. After the updated device was installed, unencrypted traffic could be captured between the Ring application and servers.

2.3 Threat Modeling

The third step in Weidman's seven-step method in penetration testing is threat modeling, but what is threat modeling? The results from the literature review of threat modeling done by Xiong and Lagerström [49] revealed that there are many definitions of threat modeling being used. Many of these definitions are not directly applicable to this thesis due to the perspective from which the threat model is constructed and the desired outcome and purpose of the threat model. Uzunov and Fernandez's definition is widely applicable with them stating "threat modeling is a process that can be used to analyze potential attacks or threats, and can also be supported by threat libraries or attack taxonomies" [50]. OWASP also provides an applicable threat modeling definition, stating "A threat model is essentially a structured representation of all the information that affects the security of an application" [51].

Threat models are helpful to identify attack vectors and threats to systems. Threat models can be used by developers when they build a system. Similarly, threat models can be used by ethical hackers when evaluating a system. Depending on the situation, threat modeling can be done once or be an iterative process. If threat modeling is being done during the development of a system, the process will likely be iterative since features are constantly being added and the system's architecture might change. With added features and changes to the architecture, new attack vectors or threats might be introduced which need to be accounted for in the threat model. Threat modeling by ethical hackers might also be iterative if new information is learned or threats need to be re-evaluated.

2.3.1 Perspective

The perspective from which a threat model is being constructed needs to be considered. This perspective helps the constructor of the model better understand what information is available as well as what parts of the system can be expected to be tested.

In **white-box** testing, the tester has knowledge about the source code and internal mechanisms of the system in question. The tests are expected to cover internal flows of the system, such as the control flow or data flow [52].

In **black-box** testing, the tester has no knowledge about the source code or internal mechanisms of the system in question. Nidhra and Dendeti stated "The code is purely considered to be a "big black box" to the tester who can't see inside the box. The software tester knows only that information can be input into the black box, and the black box will send something back out" [52]. For black-box testing, the focus is on confirming that valid and invalid inputs to the system are handled correctly.

Testing can also be performed from a mixed perspective, which is called **grey-box** testing. From this perspective, the tester has knowledge of major aspects of the system but knowledge of the source code and internal mechanisms is limited [53].

For this thesis, the penetration testing is not done in collaboration with Ring, so information regarding internal workings and source code of the Ring doorbell is not available and the perspective from which the test is conducted is a **black-box** perspective.

There are some disadvantages to only doing black-box testing. Black-box testing is mainly based on assumptions about a device or system and testing these assumptions is an expensive task [54]. Since information regarding internal workings and source code is lacking, tests can only be completed with a trial and error method [53]. Another disadvantage is the limited coverage that the perspective provides [53], hence, to get a better tested and more secure device, *both* black-box and white-box testing should be done [52].

2.3.2 Threat modeling method

As previously discussed, threat modeling can be done from different perspectives, during different stages of development or post-development independently of the development team and with varying scopes and goals. With that in mind, it can be concluded that not all methods of threat modeling are applicable to this thesis project and some methods might need to be modified.

Adam Shostack proposes the idea of viewing threat modeling as a process that is built on steps to help the completion of subgoals, rather than seeing it as a single activity [54]. He introduces the four-step framework, with the following essential questions to complete the subtasks:

1. What are you building?

2. What can go wrong with it once it is built?
3. What should you do about those things that can go wrong?
4. Did you do a decent job of analysis?

Shostack explains that the building blocks of the framework are:

1. Model System,
2. Find Threats,
3. Address Threats, and
4. Validate.

These steps are not directly applicable to the work being done in this thesis; hence they need to be modified. This thesis project is not in any way involved in building, deploying, or changing the system under consideration and all work is done from a black-box perspective. Therefore, threat modeling will be addressed differently than in the method suggested by Shostack. Specifically, instead of attempting to mitigate the threats, they need to be *identified and documented* in order to be further explored and possibly exploited. The main purpose of threat modeling for this project is to identify and document threats so that further penetration testing can continue, with the next step being vulnerability analysis.

With the selected perspective and limited access to information, it can be a troublesome task to build a model of the system. Threatmodeler, an automated threat modeling solution [55], suggests identifying assets and analyzing the attack surface as the first two steps in their six-step method to threat modeling. Additionally, OWASP suggests creating architecture diagrams, dataflow transitions, and data classifications [56] to better understand the application being built, or as in the case of this thesis, being examined.

With the combined and modified methods, the steps to complete for threat modeling for this thesis were:

1. Model System
    - Asset Identification (Section 2.3.2.1)
    - Attack Surface Analysis (Section 2.3.2.2)
2. Find Threats (Section 2.3.2.3)
    - Analysis/Prioritization
3. Document Threats (Section 2.3.2.4)

2.3.2.1 Asset Identification

The first step of modeling the system is to identify its assets. The assets are device specific, such as the video and audio streams to and from the Ring doorbell. Assets can also be material items, such as items in the house the doorbell is connected to, since the doorbell can be used in combination with a smart lock and then has the capability of opening the door it is connected to.

Assets were identified and listed to understand what the system is trying to protect and better realize the consequences of each considered attack. Identified assets are listed in Section 3.5.1.

2.3.2.2 Attack Surface Analysis

There are plenty of methods to help security researchers better identify attack surfaces. Listing use cases is sometimes used to understand devices' features and functionality. Another method is listing the technologies the devices use. The quality of this list will depend on the information gathered in the information-gathering stage.

To identify attack vectors, two models of the system were constructed. One describes the system during setup and the other describes it post-setup. Entry points to the system were also identified and listed. System models and the listed entry points are shown in Section 3.5.2.

2.3.2.3 Finding Threats

Microsoft's STRIDE is a model of threats to help testers reason about and identify potential threats. STRIDE is a mnemonic where each letter represents a threat category. To identify threats, the system model is iterated, and in each step, the tester tries to identify threats from each of the STRIDE categories [54, 57].

The threat categories that STRIDE addresses are:

Spoofing is when an attacker masks his or her identity and pretends to be another user or part of the system.

A desired property of a system is authenticity which spoofing violates.

Tampering is when an attacker modifies a part of the system he or she is not authorized to modify. This could include persistent data, wired/wireless data in transit or data in memory for example.

A desired property of a system is integrity which tampering violates.

Repudiation is when an attacker performs an action, then denies having performed that action and the system has no means of proving the attacker did this action. An example might be a user performing an action and then altering or deleting logs, or the system not having any logs in the first place.

A desired property of a system is non-repudiability which repudiation is in violation of.

Information Disclosure is when information is available to individuals that do not have authorization to view it. An example is when a user has the ability to view a file he or she should not have access to, or an attacker reads wired/wireless data in transit.

A desired property of a system is confidentiality which information disclosure is in violation of.

Denial-of-Service (DoS) is when an attacker performs an attack that denies other legitimate users service. DoS attacks can vary from completely crashing a system or by making the system unusable by slowing it down or filling all the storage.

A desired property of a system is availability which DoS in violation of.

Elevation of Privilege is when an unprivileged user gains privileged access, providing him or her with an opportunity of compromising or sabotaging the system.

A desired property of a system is authorization which elevation of privilege is in violation of.

2.3.2.4 Documenting Threats

After threats have been identified and prioritized, they will be documented for the next step, vulnerability analysis. Threats identified using STRIDE and assessed threats are found in Section 3.5.4.

2.3.2.5 Attack References

Open Web Application Security Project (OWASP) is a foundation with the goal of improving software security. OWASP is the source of many different projects such as development guides, cheat-sheets, and testing tools. One of their projects is called OWASP Top Ten, a top ten list composed of the most common security threats to various systems and applications. OWASP Top Ten lists for IoT devices and web applications will be used as references when identifying threats to the Ring system.

OWASP Top Ten Web [58]

W1: Broken Access Control
W2: Cryptographic Failures
W3: Injection
W4: Insecure Design
W5: Security Misconfiguration
W6: Vulnerable and Outdated Components
W7: Identification and Authentication Failures
W8: Software and Data Integrity Failures
W9: Security Logging and Monitoring Failures
W10: Server-Side Request Forgery (SSRF)

OWASP Top Ten IoT [59]

I1: Weak, Guessable, or Hardcoded Passwords
I2: Insecure Network Services
I3: Insecure Ecosystem Interfaces
I4: Lack of Secure Update Mechanism
I5: Use of Insecure or Outdated Components
I6: Insufficient Privacy Protection
I7: Insecure Data Transfer and Storage
I8: Lack of Device Management
I9: Insecure Default Settings
I10: Lack of Physical Hardening

Additionally, previously discovered vulnerabilities will be used as attack references. Below are the previously identified threats and vulnerabilities discussed in Section 1.5.

P1 Admin Software development kit (SDK) pages left open on AP during setup.
P2 Wi-Fi credentials transferred insecurely during setup.

P3 XSS vulnerability in web application.
P4 User sessions not ended on password change.
P5 Traffic from servers to web and mobile apps not encrypted, leading to information disclosure and spoofed video streams.
P6 Third-party trackers discovered, sharing private information.

2.4 Vulnerability Analysis

Once a threat model has been constructed and threats documented and prioritized, the threats can be analysed and vulnerabilities discovered. Vulnerabilities can be discovered manually by interacting with the system, such as open ports, but various tools can also be used to discover vulnerabilities.

The aforementioned Nmap offers users a means to broaden its usability by using Nmap scripts. A core feature of Nmap is detecting versions of scanned services, this feature can be combined with running scripts, for example, vulscan [60], which will look up vulnerabilities associated with the detected service version. Another widely used method for detecting vulnerabilities is the Metasploit Framework [61]. It can be used to develop and launch exploit code and scan for vulnerabilities.

A variety of methods and scans were used on the system to discover vulnerabilities (see Chapter 3). Those are discussed in more detail in the methods sections for each of the tested threats in Chapter 4.

2.5 Exploitation

The exploitation methods used depend heavily on the threat being evaluated at each given moment. The exploitation methods used are discussed in more detail method sections for each of the tested threats in Chapter 4.

2.6 Post Exploitation

If an attack is successful, the position a tester is left in needs to be evaluated. For example, by answering the following questions [54].

- Does the tester have access to sensitive data?
- Can the position be utilized to access other parts of the system?
- Can additional information be gathered?

2.7 Reporting

If an attack is successful, it needs to be documented with all relevant information about vulnerabilities, attack method, and consequences of the attack and disclosed to the device developer.

Chapter 3

Information Gathering and Threat Modeling Results

In this chapter, the results from the initial information-gathering stage are presented. Additionally, a system model will be presented along with enumerated threats.

3.1 System under consideration

The Ring Doorbell 3 Plus is a video doorbell that users can interact with via web, Android, and iOS applications. It was released in mid-2020.

(a) Front

(b) Back

(c) Front cover removed

Figure 3.1: Ring Doorbell 3 Plus

The device is battery-powered, with the battery designed to last up to six months. To prevent downtime when the battery is drained, users are encouraged to purchase an extra battery or hardwire the device to an existing doorbell system for continuous charging.

The device communicates via Wi-Fi and works in both 2.4 GHz and 5 GHz bands [62]. The mobile application also asks permission to connect through Bluetooth, only to enable users to continue using headphones when using the Ring application.

3.1.1 Features

The Ring device offers a variety of functionality which is listed in Table 3.1.

3.1.2 Setup

Setting up the device is done from the Ring application. The user needs to scan the QR code on the back of the device or enter the five-digit code located beneath the QR code. Next, the device needs to go into "set up mode", this is done by removing the shield on the front of the device and pressing the orange button in the top right corner of the lower half of the front of the device (see Figure 3.2). The light surrounding the button on the front of the device then starts spinning in white to indicate that set-up mode has been activated. The device will then act as an AP that the mobile device can associate with. There is no authentication needed to associate with the AP. The user can then select the Wi-Fi network the device should subsequently associate with and enter the credentials to be used to access this Wi-Fi network. The device will then finish the setup by connecting to the selected Wi-Fi network.

Table 3.1: Ring features

Feature	**Description**
Motion detection	Motion detection, with adjustable motion zones, including ”near motion zones” and people only motion to only be notified when motion was caused by a human. Notification settings are also flexible and users can control entirely what they get notifications for [62].
Pre-Roll	Device will record video prior to motion detection in lower quality so that users can get a better idea of what caused an alert.
Live View	Ring Doorbell 3 Plus has a 1080p HD camera and 2-way audio. Users can access the video and audio stream and communicate through the device at any time. The camera also has night vision, so features work during the day and night.
Privacy Zones	If the device’s field-of-view includes areas that are private or the user wants to exclude for any reason, users can create a “Privacy zone”. In that case, the privacy zone will appear as a black rectangle in the video feed. Privacy zones are dead zones, so nothing that happens in them can be viewed in the live view nor in saved recordings [63].
Share Device	Users can share their devices with other Ring users. By simply entering another user’s email account into the Ring app, that user can be allowed shared access to the device. The owner of the device can control the privileges other users have on the device or revoke all access to the device at any time [64].
Storage	Users can subscribe to Ring Protect. Videos are then stored and users can access and share at any time. Ring Protect is included for the first 30 days after Ring Doorbell 3 Plus is purchased.
Alexa and Smart lock integration	It can be set up to work with Alexa and smart locks.

Figure 3.2: Ring Doorbell 3 Plus setup mode button

3.1.3 Ports and Protocols

According to Ring Support [65] their devices use a variety of ports and protocols. The well-known ports that are used are shown in Table 3.2. Additionally, their devices use UDP, TCP, Session Initiation Protocol (SIP) and Real-time Transport Protocol (RTP) as shown in Table 3.3. Furthermore, it is mentioned that the Ring mobile applications listens for connections from the Ring servers on ports 7076/7077 and 9078/9079.

Table 3.2: Well-known ports and protocols

Port number	**Transport protocol**	**purpose**
80	TCP	Hypertext Transfer Protocol (HTTP)
443	TCP	HTTPS
53	UDP	Domain Name Service (DNS)
123	UDP	Network Time Protocol (NTP)

Table 3.3: Additional ports and protocols

Port number	**Transport protocol**	**purpose**
9998,9999	TCP	Maintain communication paths
15063,15064	TCP/UDP	SIP
16500..65000 (Range)	UDP	RTP

3.2 Scan Results

After the device has been setup and it has been verified that it is connected to the Wi-Fi network, Nmap was used to discover this device on the local area network (LAN) and further scan the device.

The device was discovered on the network with lightweight scans, such as the list scan and host discovery scan. The device was discovered as *Ring-13b395.lan* where the six characters following the dash, *i.e.,* 13b395, represents the last six hexadecimal digits in the device's media access control address (MAC address).

Following the discovery of the device and having learned its local IP address (in this case 192.168.1.132) the device could be scanned further. Multiple port scans were attempted aimed at all 65535 ports (for both UDP and TCP), but for each of these scans the results were the same, Nmap reported

that the host was up but that all ports were filtered. The results of one of the port scans can be seen in Figure 3.3.

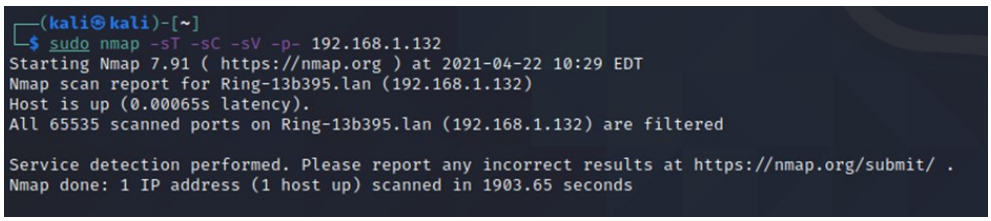

Figure 3.3: Port Scanning Result

As previously mentioned in Section 2.2.1, Nmap reports ports to be filtered when it cannot determine if a port is open or closed. This is usually caused by a firewall or filter. It was mentioned in Section 3.1.3 that several ports are indeed open on the device, but those ports could not be accessed due to the firewall or filter. Therefore, no further information was gained on open ports, services, or versions running on the Ring device by performing scans with Nmap.

Multiple types of Nmap scans were run to attempt to bypass the firewall or gain further information about the status of ports or services running on them, but no further information was gained. This included using information gained when traffic was captured, further discussed in Section 4.1.7, to spoof IP addresses and source ports.

3.3 Certificate-Pinning Bypass Results

After having successfully captured the Ring application's APK file and removed certificate-pinning, the app was re-installed on a *rooted* Samsung device. AFWall+ [66] was installed on the device to be able to limit network traffic in the mobile device. A proxy server was set up with mitmproxy [67] and the mobile devices network traffic routed through it. The UI within *mitmproxy* was then used to analyze the unencrypted network traffic.

It can be noted that Ring now mentions that data is potentially shared with third-party service providers in their terms of service [68] but also that users can opt-out of sharing their information [69].

In the analyzed traffic, a number of requests to heapanalytics.com was noted. All requests were GET requests and user and device information was being sent in with each request. Information included the page the user was

visiting, scroll depth on the page, screen dimensions and orientation. A full list of query strings sent in with requests can be seen in Figure 3.4.

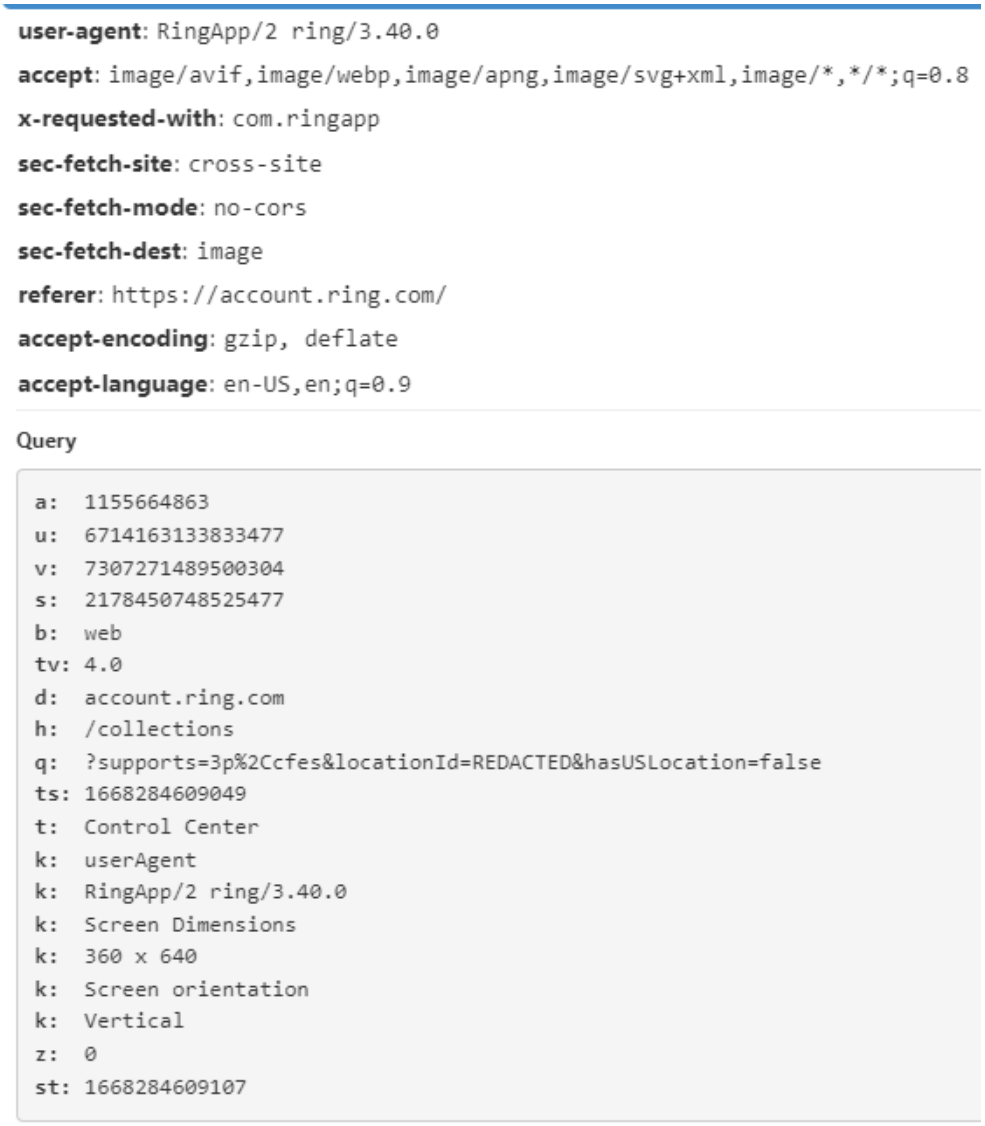

Figure 3.4: heapanalytics.com anayltics request

It was further noted that as soon as the user's preferences were updated to opt-out of sharing information with third-party providers, no further requests to *heapanalytics* or any other third-party service providers were noted.

3.4 Source Code of Android Application Capture Results

The steps in Section 2.2.2 were followed and the source code of the Ring Android application was captured. The code was somewhat obfuscated but could be used as a reference when following stack traces or searching for specific values, such as in Section 4.3.1.

3.5 Threat modeling

After taking a closer look at the system under consideration and gathering information, the threat model can be constructed. As was outlined in Section 2.3.2, this includes identifying the system's assets, analyzing its attack surface, identifying threats, and assessing those threats. All of these steps will be taken in the following section with the end result being a constructed threat model of the Ring Doorbell system.

3.5.1 Asset Identification

To gain a better overview of the system and which assets it holds and tries to protect, assets need to be identified. The identified assets are listed in Table 3.4.

Table 3.4: Identified assets for Ring doorbell

ID	Asset
1	Video Stream
2	Stored Videos
3	Firmware
4	Certificates
5	Encryption Keys
6	Wi-Fi credentials
7	Ring account credentials and information
8	Event logs
9	Physical safety of people and objects inside house
10	Physical security of device

3.5.2 Attack Surface Analysis

In the following section, the Ring doorbell's overall system is modeled to better realize how parts of the system communicate and possible entry points to the system identified. This will provide an overview of the system and its attack surface.

3.5.2.1 System model

The Ring doorbell has a fairly simple structure, leaving it with a small attack vector. After the Ring doorbell has been set up, all communications go through Ring's cloud servers. Communications from the web or mobile applications with the Ring doorbell and vice versa will first go through the Ring servers, even if all devices are on the same LAN. This is shown in Figure 3.5.

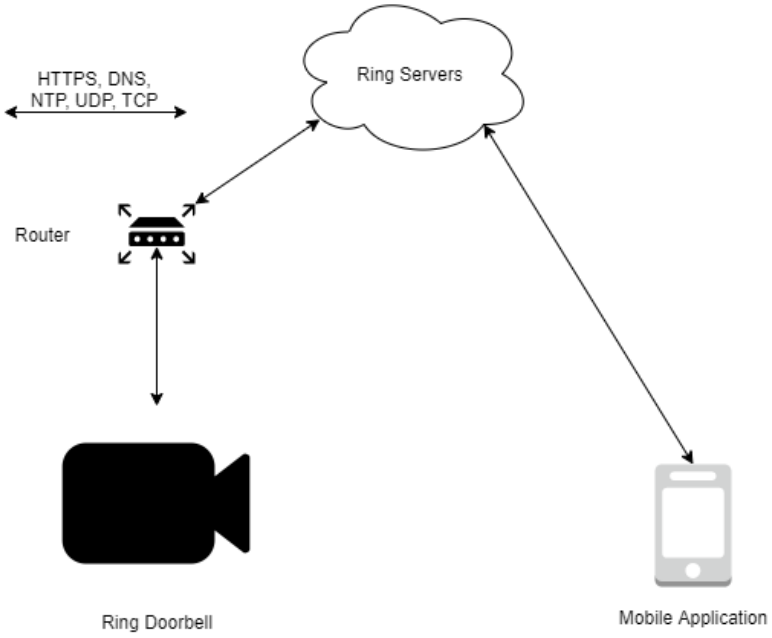

Figure 3.5: Ring doorbell system post setup

As previously discussed, the Ring doorbell acts as an AP during setup. The mobile device used for the setup will then connect to this AP and the mobile device will be used to provide the credentials needed to connect to a local Wi-Fi network. The Ring doorbell then connects to the Wi-Fi network using these credentials and exits setup mode. This is shown in Figure 3.6.

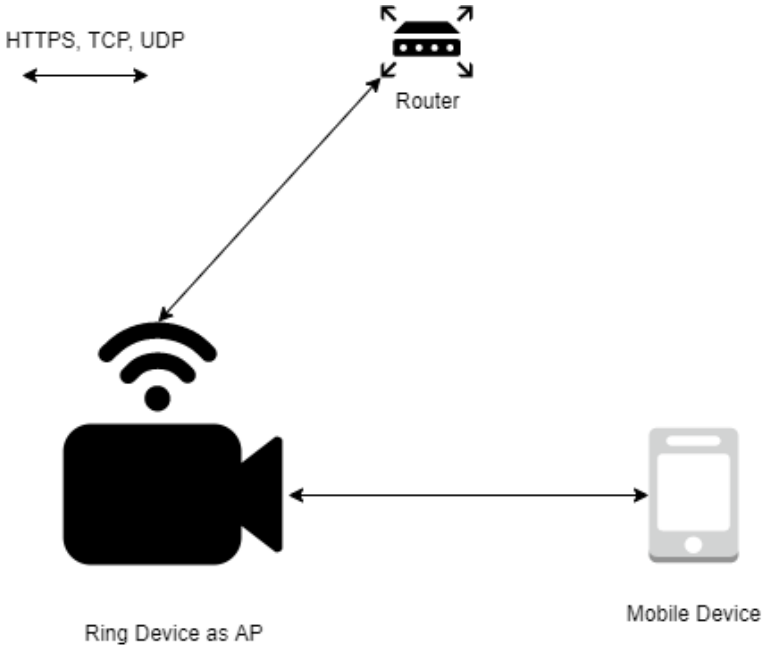

Figure 3.6: Ring doorbell system during setup

3.5.2.2 Entry Points

To gain a better overview of the system's entry points and better realize the attack surfaces, entry points were enumerated and are listed in Table 3.5.

Table 3.5: Identified entry points for the system

ID	Entry point	Description
1	Mobile Application	The mobile application contains all user information and critical device information. It provides an interface to view and change that information, as well as interact with the system.
2	Web Application	The web application provides similar functionality to the mobile application. User and device information can be accessed and changed, but interaction with the device is slightly limited compared to the mobile application.
3	Access Point	During setup, devices in range can connect to the Ring device as an AP. One of these devices will inform the Ring devices of the user's selection of Wi-Fi network and provide the credentials to access it.
4	Wireless communications	All communications between the Ring applications and the Ring doorbell are via wireless links.
5	Firmware	Firmware is stored on the device and includes the instructions and data for the device to operate.
6	Hardware	The Ring hardware can be accessed. Encryption keys, Wi-Fi credentials, and certificates are stored in the device. By targeting the hardware, attackers can in some cases set up a backdoor or access the firmware and disassemble this code to produce source code.
7	Ring Doorbell	The Ring doorbell can be found on the local network, but all ports are reported as filtered and attempts to bypass the firewall did not work.
8	Ring Servers	There is no official Application Programming Interface (API) for the Ring servers, but some endpoints can be identified by analyzing requests made via the web application and traffic captured after certificate-pinning bypass in the mobile application. Finally, there are also some unofficial APIs available online*.

*For example, an unofficial API is documented in https://github.com/dgreif/ring

3.5.3 Identifying Threats

In Section 2.3.2.5, three OWASP Top Ten lists were used as attack references as well as previously identified threats. In combination with these attack references, STRIDE was used to brainstorm as many threats as possible. All of these threats are described in Section 3.5.3.1.

3.5.3.1 Identified Threats

Identified threats were categorized into the six threat categories represented in the STRIDE mnemonic and listed in Table 3.6, where each threat has been given an ID. Some threats do not necessarily belong to a single STRIDE category. All consequences are considered for each threat but threats will only be placed in one category.

Table 3.6: Identified Threats

Category	ID	Threat
Spoofing	S1	Spoof user-agent when logging in to Ring account to bypass 2-factor authentication.
Tampering	S2	Obtain firmware & code injection via firmware updates.
	S3	Tamper with QR code on the back of Doorbell so that malicious code is scanned during setup.
Repudiation		None
Information Disclosure	S4	Services exposed during setup.
	S5	Perform downgrade attack on connections being made in the system.
DoS	S6	Spam Doorbell with requests to empty battery or disrupt regular functionality
Elevation of Privileges	S7	Ring accounts are vulnerable to credential stuffing attacks.

3.5.4 Assessing Threats

Now that threats have been identified they can be further assessed. A few factors need to be considered when assessing threats, including the overall scope of the project with regard to time, resources, and perspective, the likelihood of discovering a vulnerability related to a threat, and the potential impact of discovering a vulnerability related to a threat. In the following

section, identified threats will be assessed with the aforementioned factors in mind, outlining which threats will be excluded and which will be included in this project.

3.5.4.1 Hardware

For this thesis project, hardware threats are not considered. The hardware was not considered a viable entry point to the system for several reasons. For an attacker to exploit the hardware he or she would first need to physically steal the doorbell, disassemble it, exploit it, and then possibly return it. The physical security of the doorbell will vary, but in many cases, it might be quite easy to steal it. On the other hand, the Ring Doorbell is a security camera and will detect an attacker removing it from its location and alerting its owner. Attackers could mask their faces or disguise their identity by pointing a laser or light into the camera during the theft, but the owner will nevertheless be notified.

Hardware in the previous versions of the Ring doorbell has not been found to be insecure. Although a universal asynchronous receiver-transmitter (UART) connection was available, but none of the available commands were found to contain vulnerabilities [70]. The hardware of other Ring products has also been tested and proved to be secure [71].

Lastly, the time constraint of the project was considered; hence, not all threats could be further assessed within the given time frame for this thesis project, and therefore, hardware threats were omitted.

3.5.4.2 Repudiation/Logging

Testing logs require access to logs for most tests. Logs can also be tested to some extent by performing actions repeatedly, those actions should be logged and might warrant a response from the system at some point. Many of these tests are quite intrusive and cannot be performed due to legal issues. Some logging was noted when certificate-pinning was bypassed but no logs were observed on the Ring Doorbell in the web application. Therefore, tests and observations about logs are limited, but some remarks were made and conclusions drawn with the limited tests that could be performed and the data captured.

3.5.4.3 Server Threats

For penetration tests done from a black-box perspective and not in collaboration with the client, testing for DoS vulnerabilities cannot be done due to legal issues. Most threats regarding the Ring servers can be deemed to be intrusive and thus cannot be done due to legal reasons.

3.5.4.4 Communications with doorbell

It is good to note that no *direct* means to communicate with the doorbell have been discovered. All ports are filtered and attempts to bypass the firewall were not successful. Additionally, no physical connections can be made to the doorbell, except for a USB port on the battery, used for charging. Moreover, the battery needs to be removed to enable someone to plugin a cable to this USB connector.

3.5.4.5 Selected Threats

Seven STRIDE threats were identified, but all fall within an OWASP Top Ten category. Therefore, each of the OWASP Top Ten threats for IoT and Web will be considered and discussed in the next chapter. The STRIDE threats discussed were applicable, except for the QR-code injection threat which is discussed in a section of its own.

Chapter 4

Exploits

In this chapter, threats selected for further assessment are examined. For each of the addressed threats background, method, result, and discussion is provided. The contents of the chapter are grouped into the OWASP Top Ten - IoT (see Section 4.1), OWASP Top Ten - Web (see Section 4.2), and STRIDE threats (see Section 4.3).

4.1 OWASP Top Ten - IoT

An OWASP top ten list for IoT was initially released in 2014 but later updated in 2018 [59]. In the following section threats from the updated 2018 top ten list will be considered. The following threats are versatile and the depth of assessment for each threat depends heavily on how much information could be gathered relating to it as well as how thoroughly it could be tested within the frame of the law. Additionally, some threats or aspects of threats fall outside the scope of this thesis project as discussed in Section 3.5.4.

4.1.1 I1: Weak, guessable, or hardcoded passwords

According to OWASP, the number one threat to IoT devices are weak, guessable, or hardcoded passwords. Many IoT devices have default passwords set by manufacturers that can be used to authenticate into some part of the device's ecosystem. This threat can refer to (i) passwords that are hardcoded, (ii) weak or guessable passwords that manufacturers expect users to change after purchase or setup of device, and (iii) passwords that can be easily reverted back to factory defaults that are known, weak, or guessable.

Having hardcoded passwords is generally not a good security practice since if these passwords become known to an attacker, the user has no way of changing them and is left with a permanently vulnerable device.

Expecting users to change passwords is also not a good security practice, since users do not always behave as expected. There can be many reasons why users do not change the default password, such as being unaware that a password requires changing, forgetting to change the password or simply not caring enough to change the password, and being unaware of the risks associated with leaving the weak password set as a default.

Lastly, some devices allow users to easily change their password back to a weak, known, or guessable password which can serve as a backdoor for attackers even when users have already changed their password from the default one.

4.1.1.1 Method

Examine the system and identify parts of it that require authentication. Where authentication is required, find out where passwords are set, the strength of those passwords, and how easily they can be changed.

4.1.1.2 Result

As discussed in Section 2.2, scanning of the Ring Device did not reveal any accessible services. Additional scans performed during setup, discussed in Section 4.1.2, revealed the same results.

Ring accounts are required for users to be able to use the web or mobile interfaces to communicate with the Ring doorbell. These Ring accounts are not created by default when a Ring product is bought and users are required to create their own accounts on the Ring website, providing their email address and creating their own password. Ring enforces strong passwords for Ring user accounts [72], which are composed according to seven rules:

1. Must be at least eight characters long.
2. Must include both upper- and lowercase letters.
3. Must include at least one number.
4. Must include at least one symbol.
5. Must not be easily guessable (*e.g.,* ”password”)
6. Must not include the first or last name of the user nor the user’s email address.
7. If this is not a new Ring account password, then the password cannot be the same as the previous password.

4.1.1.3 Discussion

The password creation requirements by Ring force users to use strong passwords. These requirements are in line with recommendations from National Institute of Standards and Technology (NIST) [73] and do not allow weak easily guessable passwords and the required minimum length of 8-characters makes brute-forcing passwords unfeasible.

4.1.2 I2: Insecure Network Services

This threat relates to network services running within IoT device's ecosystems and being exposed to the internet. Such network services, when not properly configured or insecure, can lead to attackers gaining remote access to the system or unauthorized access to data.

As discussed in Section 2.2, there were no (accessible) services discovered running on the Ring device, with all ports closed or filtered. The only way to communicate with the doorbell is through web and mobile interfaces, where all communications go through the Ring servers. It was also noted that during setup, the Ring device acts as an AP that users connect to using the mobile application and Wi-Fi network configuration information is passed to the device. Ideally, the setup only needs to be completed once, but attackers could force users to perform the setup on multiple occasions, for example by kicking the Doorbell off the Wi-Fi network using a deauthorization attack. Since the setup phase is a small part of the system, security considerations might be more likely to be overlooked, and in the past, this has been the case, with traffic being sent insecurely and SDK pages being left open on the AP.

4.1.2.1 Method

Scanning the AP during setup was done using Nmap. Nmap scans will reveal running services and open SDK pages. Metasploit was used for additional scans and Telnet and OpenSSL for manually connecting to and testing ports [74].

4.1.2.2 Result

The Nmap scans did not reveal any SDK pages left open on the AP. Scans revealed two open ports and the device's MAC address, while the rest of the ports were closed or filtered. Results for an Nmap scan with the -A flag set, which enables OS detection, version detection, script scanning, and traceroute can be seen in Figure 4.1.

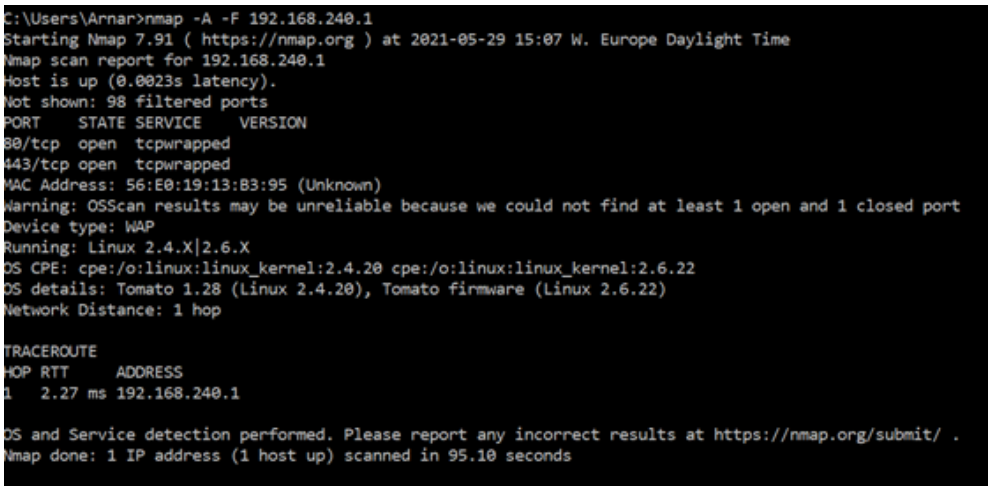

Figure 4.1: Nmap scan results on AP

The ports are reported as *tcpwrapped* which indicates that there is an access control program running, protecting these open ports. Further scans were run, including one against the two open tcpwrapped ports using all SSL scripts included in Nmap. The results of this scan can be seen in Figure 4.2.

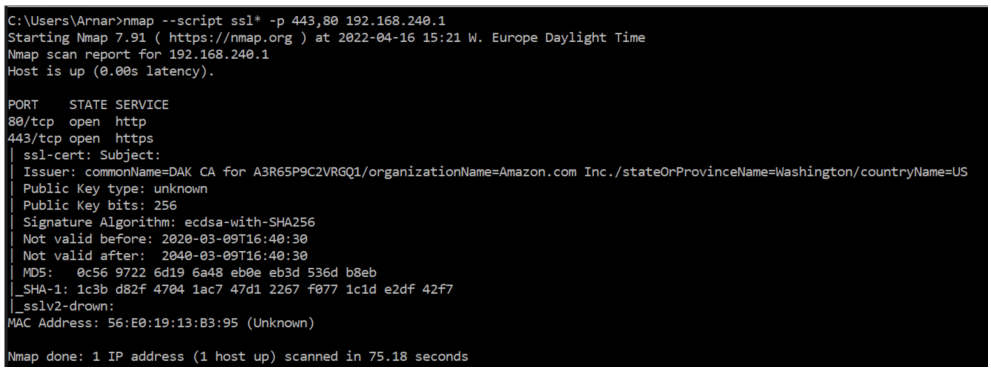

Figure 4.2: Nmap scan with SSL scripts

Multiple scans were performed to gather more information and to attempt to bypass the tcpwrapper. Including scans with HTTP- and SSL-specific scripts, spoofed information such as IP and Message Authentication Code (MAC) addresses of the mobile device being used to perform the setup as well as Ring server IP addresses and source ports noted when traffic was captured in Section 4.1.7. None of the scans resulted in any further information gained.

Banner grabbing using Telnet and Metasploit was attempted but proved unsuccessful.

A connection could be established to the open ports using Telnet, but any interaction resulted in the connection being closed with the message "Connection closed by foreign host", and the device exiting setup mode. This was additionally tried using *OpenSSL* which could not complete a connection due to an error stating that a self-signed SSL certificate was detected.

A scan of the Ring Doorbell during setup was re-attempted after an automatic firmware update was performed. It was noted that more restrictions had been applied and mainly that port 80 had been closed as well and the only port reported to be open was 443. Results from this scan can be seen in Figure 4.3

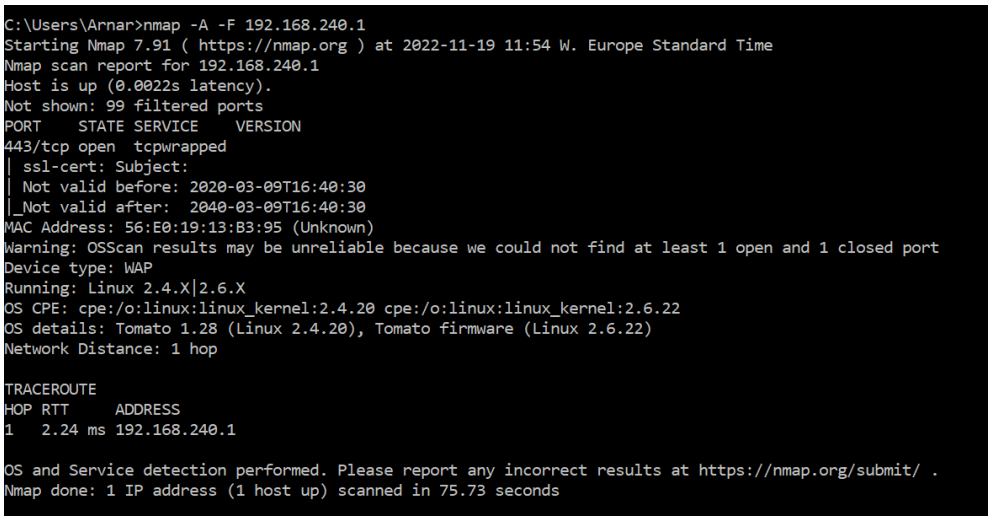

Figure 4.3: Nmap scan results on AP - Post Firmware update

4.1.2.3 Discussion

When Nmap reports ports to be tcpwrapped it refers to the Linux and Unix network access control program tcpwrapper. As can be observed when connecting manually with the open ports using Telnet, the connection is completed successfully but the connection is then closed without any further data being received from the host. Since tcpwrapper is used to protect programs and not ports, it would indicate that there is an actual program listening on the ports, as opposed to the scans run on the device post setup where all ports are reported filtered and there is no way of telling which ports actually have running services. Therefore, the tcpwrapped result indicates that there are services running on the ports, but only specific hosts are allowed to communicate with them [75].

Traffic capturing, including during setup is further discussed in Section 4.1.7, but it can be noted here the only connections made were from port 443. Port 443 is used for HTTPS services and in this case, has a tcpwrapper in place to protect those services (as did port 80 before being closed). Results in Figure 4.2 indicate that an SSL-certificate is needed to connect to the ports. As discussed in Section 2.2.3, the Ring mobile app was patched and reinstalled with certificate pinning implementations removed, but the certificates themselves are not removed and they might be needed to connect to port 443. Additionally, in this section and Section 3.2 it was noted that source IP addresses and ports were spoofed and it can therefore also be concluded that they are not needed to communicate with ports of the Ring doorbell.

As reported by Nmap and seen in Figure 4.1, OS scans may be unreliable due to a lack of information gathered during scans. Nevertheless, the reported OS, firmware, and Linux version (*i.e.,* Linux 2.6.22) do not have any known vulnerabilities.

Information gathering revealed that the Ring doorbell has no exposed network services after it has been set up and in this section, it has been confirmed that no services are exposed during setup.

4.1.3 I3: Insecure Ecosystem Interfaces

It is not always the IoT device itself that contains vulnerabilities, a big part of evaluating the security of a device is assessing the full ecosystem, which includes interfaces such as web- and/or mobile applications. The Ring doorbell has two interfaces, web- and mobile. This single OWASP threat does not do those interfaces justice when it comes to assessing their security. As was discussed in Section 3.5.3, quite a few threats were identified and pursued specifically for the Ring interfaces. Therefore, this threat will not be discussed further, and the security of the interfaces will be evaluated incrementally by looking at individual threats specific to the web and mobile interfaces.

4.1.4 I4: Lack of Secure Update Mechanisms

Firmware is the software on a device that is used to control the device's hardware. The complexity of firmware can vary depending on the device. In some cases it has multiple functionalities, controlling and monitoring the device, *i.e.,* acting as an OS for the whole device and in others, it only provides services for high-level software (such as the OS).

A Microsoft-commissioned report published in March 2021 found that firmware attacks are becoming more common [76]. Inspecting firmware can expose vulnerabilities in the system, backdoors, or stored credentials. Devices can also be exploited by tampering with firmware updates in transit or by injecting malicious firmware updates.

This threat relates to the Ring's device's ability to securely update its firmware. Additionally, further steps were taken into acquiring the firmware.

4.1.4.1 Method

There are a few methods that allow attackers to obtain firmware, such as traffic sniffing, OSINT, or by analyzing web and/or mobile applications [77, 78, 79].

Even if the firmware is unobtainable, a device can be exploited by tampering with firmware updates in transit or by injecting malicious code as a firmware update to the device.

OWASP provides a checklist that can be used when validating a device's update mechanism [59]:

- Firmware validation when installed or updated.
- Firmware updates are securely transferred.
- Anti-rollback mechanism to prevent attackers from downgrading firmware to an older, less secure, version.
- Notifications of security changes triggered when system is updated.

4.1.4.2 Result

- The firmware or information regarding it could not be obtained
- Firmware updates in transit could not be tampered with.
- Malicious firmware updates could not be sent to the Ring device.
- Firmware validation, presence of an anti-rollback mechanism, or notifications of security changes could not be validated.
- Firmware updates are transferred securely.

4.1.4.3 Discussion

Neither the firmware itself nor any information regarding it could be gathered through OSINT. Further inspections of web and mobile applications did not reveal any information regarding firmware, neither did inspecting the Android mobile apps APK files. It can be noted that it is unlikely to find information regarding firmware in the web and mobile apps since they do not communicate

directly with the device. Hardware hacking to obtain firmware was not attempted since it is outside the scope of this project.

Firmware updates are securely transferred, likewise, all traffic in the Ring ecosystem is secure and firmware could therefore not be obtained through traffic sniffing and firmware updates in transit could not be tampered with. Security of traffic is further discussed in Section 4.1.7 .

As discussed previously, no means of direct communications with the Ring device were discovered; therefore, a malicious firmware update could not be sent to the device.

Firmware validation and the presence of an anti-rollback mechanism could not be verified from a black-box perspective. In both the web and mobile applications, users can see the status of firmware on the Ring device. Specific versions of firmware cannot be seen, but users see the status of firmware as either "Up to date" or a message indicating that the firmware will update. Firmware updates are done automatically and transferred securely directly from the Ring servers. If the firmware needs an update, the update will be triggered on the doorbell's next event, such as motion detection or doorbell ring. No notifications are received when the device needs an update or when an update has finished, but it could not be verified that the system sends notifications regarding specific security changes when they occur. This update strategy is referred to as Over-the-air (OTA) firmware updates, or more specifically an automatic-OTA update since the updates are pushed from the Ring servers and the doorbell updates its firmware without any user feedback [80]. In a report were a previous version of the Ring doorbells hardware and firmware was examined and published on *exploitee.rs* [70], it was noted that there are two firmware files present on the device. One file refers to the microcontroller that is used to realize the doorbell's functionality, while the other refers to the Wi-Fi module for further firmware updates and general Wi-Fi functionality.

4.1.5 I5: Use of Insecure or Outdated Components

As mentioned in the previous section, the firmware could not be obtained. Additionally, no information was gathered regarding services running on the Ring device pre-setup or post-setup as discussed in Sections 2.2 and 4.1.2. Therefore, no information was gathered on the components of the Ring camera. Considering the information gathered and that the system is being tested from a black-box perspective, it cannot be verified whether the Ring Doorbell uses insecure or outdated components.

This threat only relates to the usage of insecure or outdated components on the IoT device itself and not interfaces within the ecosystem. Usage of outdated or vulnerable components on the Ring web application is discussed in Section 4.2.6.

4.1.6 I6: Insufficient Privacy Protection

This threat relates to the transfer and storage of data, especially sensitive data, such as the video captured on the Ring doorbell and user information linked to Ring accounts. As was discussed in Section 3.5, all communications between interfaces and the device go through the Ring servers. All user information accessible on the web- and mobile interfaces are fetched from the Ring servers. Additionally, live streams from the doorbell to the interfaces go through the Ring servers. The current assumption is that all sensitive data is stored on the Ring servers and that only certificates, encryption keys, and Wi-Fi credentials might be stored in the Ring device itself, though that cannot be verified. Additionally, it cannot be verified whether all sensitive data is stored in encrypted form on the Ring servers nor was it possible to tell how securely it is stored.

All data transfers relating to the Ring device and within the Ring ecosystem are discussed in Section 4.1.7. Additional information regarding the cryptographic safety of data transfers is discussed in Section 4.2.2 and usage of security headers in the web application is discussed in Section 4.2.5.

4.1.7 I7: Insecure Data Transfer and Storage

Previous penetration tests on older versions of the Ring doorbell revealed that traffic was being transferred unencrypted during setup and during communications between web/mobile interfaces and servers. This meant that attackers that captured the traffic could read sensitive information in plaintext such as Wi-Fi credentials or account information, and even capture the doorbell's video stream and send spoofed videos to the Ring applications.

All traffic within the system has similar security features, so for this section, the traffic to and from the Ring device is inspected along with all of the traffic within the Ring system.

4.1.7.1 Method

To capture traffic between mobile applications and Ring servers, Address Resolution Protocol (ARP) poisoning with *Ettercap* [81] was performed. A

computer that will intercept the traffic is connected to the same network as the mobile device running the Ring application. False ARP reply messages are sent to the network's default gateway and the mobile on the network, poisoning their ARP caches. This means that on the default gateway, the mobile's IP address is associated with the MAC address of the sniffing computer and on the mobile, the IP address of the default gateway is associated with the sniffing computer. Once the ARP poisoning has been performed, all traffic to and from the mobile applications will go through the listening computer before reaching its destination.

Wireshark [82] was used to capture traffic from the web application to the Ring servers. Wireshark can capture traffic on the device it is running on by performing traffic capturing in promiscuous mode, and can therefore capture the traffic when the Ring website is visited on the device. The sniffing device was a Dell computer running the Windows 10 OS.

ARP poisoning was not effective when capturing traffic between the Ring doorbell and the Ring servers. Instead, a Windows computer was set up as an AP and the Ring Doorbell connected to it. Wireshark was set up on this computer to capture the traffic from the Doorbell to the servers.

To capture traffic on the Ring AP during setup, *i.e.,* between the Ring doorbell and the Ring servers, a Mac computer with a wireless network card that supports monitor mode was utilized. Wireshark was then run in monitor mode which allows it to capture traffic on the network it is connected to, *i.e.,* it could then capture traffic to and from the AP without the traffic flowing directly through the computer that was performing traffic sniffing.

4.1.7.2 Result

The following traffic was captured:

- Setup: Mobile connects to and sends data to AP.
- Setup: Doorbell connects to Wi-Fi network.
- Doorbell communicates with the Ring servers.
- Mobile applications communicate with the Ring servers.
- Web application communicates with the Ring servers.

All captured traffic was encrypted. TLSv1.2 was noted on each of the connections being made and no older versions of TLS were observed.

4.1.7.3 Discussion

TLSv1.2 is considered secure [83, 84] but it is not without issues. The main issue is that there are a lot of options when it comes to cipher suites, key exchange schemes, signature formats, *etc*. Therefore, the security of TLSv1.2 will depend upon its actual configuration. The newest TLS version, v1.3, is considered to be more secure, both due to which options it supports and also due to offering fewer options, minimizing the threat of misconfiguration. Threats due to incorrect configurations of the web application are discussed further in Section 4.2.5.

With traffic securely encrypted there is no information leak of data in transit and communications cannot be tampered with.

4.1.8 I8: Lack of Device Management

As discussed in Section 3.5, there is only one device, the Ring video doorbell and it only communicates directly with the Ring servers, with seemingly no data stored on the device itself, except for possibly Wi-Fi credentials, certificates, and encryption keys. The device's communications can also be considered secure as discussed in the previous section, *i.e.,* Section 4.1.7. The update process of the device was also discussed in Section 4.1.4 and was deemed secure. Moreover, the update mechanism allows Ring to rollout security updates quickly and easily.

Users can manage their devices from either the web- or mobile interface. There, they can see device information, such as battery and Wi-Fi status. Via those interfaces, users can also interact with the device, such as getting a live feed from it, and can easily manage the device's settings (such as notification settings, blackspot settings, and capture settings). Lastly, users can easily manage access to their device from the various interfaces by allowing other Ring users access and controlling the extent of their access.

With the information gathered and discussed in previous sections, the Ring doorbell is thought to be securely managed and no more tests could be performed to verify the management of the device.

4.1.9 I9: Insecure Default Settings

All security settings discussed so far in this thesis have been found to be secure, *i.e.,* access and communications to the device can only happen via Ring servers and there are no open ports on the device that enable direct access to the device. During setup, SDK pages have been disabled for users and open ports are

protected with tcpwrapper as discussed in Section 4.1.2. Users have control over their devices, while Ring can swiftly and securely update those devices if vulnerabilities are discovered.

No weak, guessable, or hardcoded passwords are used on the Ring device and users need to create Ring accounts with secure passwords to be able to access interfaces that can communicate with the Ring device (as described in Section 4.1.1). Additionally, 2-factor authentication is enabled by default for all Ring user accounts and cannot be opted out of.

No further tests could be done to validate the default security settings of the device. Default security settings noted and discussed so far, indicate that the device is securely set up by default.

4.1.10 I10: Lack of Physical Hardening

As discussed in Section 3.5, hardware threats are considered to be outside the scope of this thesis. Therefore, this threat was not inspected any further and no assessments were made of the physical security of the device.

4.2 OWASP Top Ten - Web

The latest OWASP top ten list relating to web applications was released in 2021 [58]. It has some similarities to its predecessor that came out in 2017 with some re-ordering of the list, some threats being merged together but also three new threats being introduced. New threats introduced were (i) insecure design (ii) software and data integrity failures (iii) Server-side request forgery (SSRF). In the following section the most recent, 2021, OWASP top ten list for web will be considered in regards to the Ring Doorbell system.

4.2.1 W1: Broken Access Control

This section does not contain tests that directly assessed if access control on the web application was working properly. This is due to the fact that most tests and tools used to perform those tests are too intrusive and cannot be performed legally. Additionally, since the Ring application is being tested from a black-box perspective, it can be hard to verify that access control is as secure as it should be. Therefore, this threat cannot be fully verified, although some aspects related to this threat have been tested. Penetration tests performed in this thesis that relate to this threat are force browsing, parameter tampering, id or cookie tampering, black-box view of the logged access control failures

and rate limiting, and evaluation of session management. These tests are performed and discussed in Sections 4.2.5 and 4.2.7.

4.2.2 W2: Cryptographic Failures

When a connection is being made between two parties such as client and server, a handshake is performed. A handshake is when the two parties exchange information regarding the communication protocols. In Section 4.1.7 a Transport Layer Security (TLS) handshake was noted, where the two parties agreed upon a TLS version and cipher suite and the server was authenticated with a public key and SSL certificate, lastly session keys were generated to encrypt and decrypt traffic between the two parties. When an attacker sits in the middle and performs a MITM attack, they can often retrieve valuable information or tamper with the traffic. ARP poisoning is a form of MITM attack and was used in Section 4.1.7 to capture traffic. When the traffic is securely encrypted it makes it a lot harder for attackers to retrieve information or tamper with traffic. In some cases, the attacker can trick the two communicating parties into using a less secure protocol or encryption algorithm, which can allow the attacker to decrypt traffic, and retrieve and possibly tamper with data in transit, this is called a *downgrade attack*.

Block ciphers are a common method used to encrypt traffic between parties. Block ciphers work by splitting the data to be encrypted into fixed-length blocks, for example, 128 or 256 bits, and encrypting one block at a time using a key of a predetermined length. Block ciphers are often used in combination with Cipher block chaining (CBC), where each block of encrypted data relies on the previous block of encrypted data [85]. In block ciphers, each block needs to be of the same length, so *padding* is used to ensure that the final block will be of the agreed-upon length. Different types of padding schemes can be used for block ciphers, for example, Public Key Cryptography Standards (PKCS) padding, where each byte of added padding has its value set to the length of the padding. The final byte of the final block should always represent the padding length so the decryption algorithm can determine where the data ends and the padding starts. Since all the padding bytes have the same value, during decryption it can be asserted that all bytes in the padding range are actually padding. Since the last byte needs to represent the padding length, if the data is divisible with the block size, a whole extra block of padding needs to be added so that the final byte of the data is not interpreted as the padding length [86]. To ensure the integrity of the encrypted data, a MAC is used. The MAC for an encrypted message is calculated using

the block cipher, secret key, and the blocks of the message. An incorrect MAC indicates that the message might have been tampered with [87].

Padding oracle attacks are attacks mostly seen against block ciphers used in combination with CBC mode. Padding oracle attacks are when the attacker can observe the validation of cryptographic messages and derive information from the validation response. There are different types of Padding oracle attacks, depending on the information that can be derived from validation responses [88, 89]. For example, information can be derived when validation responds with padding length errors, padding validity errors, or MAC validity errors. Padding oracle attacks are often the final step of an attack, where the first step is a downgrade attack to trick a victim's client into using encryption that is possibly vulnerable to padding oracle attacks. An example of this is the Padding Oracle On Downgraded Legacy Encryption (POODLE) attack, where either TLS connections are downgraded to SSLv3 where the ciphers used are vulnerable to padding oracles, or the selected encryption algorithm used in TLS is downgraded to an algorithm vulnerable to padding oracles. Similar attacks include: GOLDENDOODLE, Zombie POODLE, Sleeping POODLE, and OpenSSL 0-Length attacks. The difference between them being what information can be gathered from validation responses [90, 91, 92]. An important note is that computational time for decrypting, validating MAC, and padding may all differ but the validation of cryptographic messages should always have the same or similar response time to prevent side-channel attacks, so that attackers cannot gain any information, for example learning the padding length by evaluating the response time, as this would have the same consequences as too informative error messages [93]. Downgrade and padding oracle attacks can depend on the protocols used, their versions, cipher suites, implementations, and configurations. In most cases, downgrade attack vulnerabilities occur when systems strive to be backward compatible and therefore support less secure protocols and cipher suites.

4.2.2.1 Method

As mentioned in Section 4.1.7 network traffic was captured within the Ring system which could then be further analysed for this section. Additionally, many types of scanners exist on the web that scan for individual vulnerabilities, such as the POODLE [94] scanner but for a more comprehensive result, the Qualys SSL server test was used [95].

The following questions from the OWASP Sensitive Data Exposure cheat-sheet [96] were also considered:

- Is any data transmitted in clear text?
- Are any old or weak cryptographic algorithms used either by default or in older code?
- Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing?
- Is encryption not enforced?
- Does the user agent not verify if the received server certificate is valid?
- Are initialization vectors ignored, reused, or not generated sufficiently secure for the cryptographic mode of operation? Is an insecure mode of operation such as Electronic codebook (ECB) in use? Is encryption used when authenticated encryption is more appropriate?
- Are passwords being used as cryptographic keys in absence of a password base key derivation function?
- Is randomness used for a cryptographic purpose that was not designed to meet cryptographic requirements?
- Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when cryptographic hash functions are needed?
- Are deprecated cryptographic padding methods such as PKCS number 1 v1.5 in use?
- Are cryptographic error messages or side channel information exploitable, for example in the form of padding oracle attacks?

4.2.2.2 Result

The Qualys SSL server scan resulted in an overall grade of B on the scale of A to E. The grade was capped at B, due to the server's backward compatibility. This occurs because although the Ring Doorbell uses TLSv1.2, it also supports versions 1.1 and 1.0. The complete cipher suites supported in versions 1.1 and 1.0 are considered weak by Qualys and most of the encryption algorithms supported in TSLv1.2 are also considered weak by Qualys due to them being CBC algorithms which can be considered vulnerable if the protocol is configured incorrectly [97].

The Qualys SSL server scan checks for protocol versions, configurations, and server responses to see if the device being tested is vulnerable to known attacks. The Ring servers were considered secure against all attacks:

- POODLE(SSLv3) [98]
- POODLE(TLS) [99]
- Zombie POODLE, GOLDENPOODLE, Sleeping POODLE and OpenSSL 0-Length [100, 101]
- SSL/TLS compression (Insecure configuration) [102]
- RC4 (Insecure cipher) [103]
- Heartbeat (Insecure configuration)
- Heartbleed [104]
- Ticketbleed [105]
- Open SSL Common Command Set (CCS) vulnerability [106]
- Open SSL Padding Oracle vulnerability [107]
- Robot [108]

The Ring servers also have downgrade attack prevention, with TLS_FALLBACK_SCSV supported.

It was also noted that when connecting to the Ring servers that TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 is the preferred cipher suite which is considered secure and has no known vulnerabilities [109].

4.2.2.3 Discussion

As discussed in Section 4.1.7, TLSv1.2 is generally considered secure. However, the Ring servers are backwards compatible and support TLS versions 1.1 and 1.0, but SSLv3 has been disabled. All cipher suites supported by TLS 1.1 and 1.0 are considered weak as well as a considerable number of the cipher suites supported by version 1.2. They are mainly considered weak due to the fact that they are block ciphers in CBC mode which opens the door for padding oracle attacks as discussed earlier, but as previously mentioned, how vulnerable they are relies heavily on the implementation of the ciphers. It was noted in the captured traffic and the results from the Qualys SSL test, that the Ring servers were not deemed vulnerable to any of the known padding oracle attacks and their implementation seems secure. Ideally, block ciphers in CBC mode should be completely disabled [110] but in practice that is not an option since backward compatibility becomes an issue and Ring would not be able to support many of their clients.

What can be done when less secure protocols or cipher suites need to be supported? First, the server needs to support the more secure or preferable

protocols and cipher suites. Not only do they need to be supported, but also preferred, so the order needs to be considered [111, 112]. This is already done by Ring, by supporting and defaulting to TLSv1.2 with the TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 cyber suite. Next the server should have TLS_FALLBACK_SCSV enabled. This option ensures that clients connect to servers with their most secure protocol and cipher suite, in other words, it prevents downgrade attacks. Incorrectly configured servers sometimes crash during protocol negotiation or attackers performing MITM attacks sometimes spoof server responses, responding with a server crash during negotiation, which leads to clients attempting to re-connect with less secure protocols or cipher suites. The TLS_FALLBACK_SCSV option prevents this behaviour and causes connections to drop if the most secure protocol is not used for the connection. As was noted in the result section, the TLS_FALLBACK_SCSV option is enabled on the Ring servers. Third, the implementation of the cipher suites needs to be closely considered and responses need to be carefully considered so that there is no unnecessary information leak that attackers can potentially use when padding, MAC or 0-length records are incorrectly sent to servers [113].

In regards to the Qualys SSL Lab test, it does not actually perform specific or thorough tests for all the vulnerabilities it is deemed to be secure from. For example, when testing for padding oracle attacks, the test only checks for which protocols are supported, which cipher suites are in use, and if the server protects against downgrade attacks. For example, it deems Ring to be secure against POODLE(SSLv3) since SSLv3 is not supported and POODLE(TLS) since the Ring servers default to TLSv1.2 with a secure cipher suite. This test does not consider all the cipher suites supported, all types of payloads that could be sent to the server to gain information, nor if the timing of responses leak information. This test is, strictly speaking, not a vulnerability test, and if they were, using it against the Ring servers might be considered illegal. The author of this thesis contacted Ring with a request to test the servers more thoroughly but got no response, so further tests could not be performed.

It is also worth mentioning that even if the server is deemed vulnerable to padding oracle attacks, the conditions for the attack are extremely difficult. First, the attacker would need to be able to run scripts within the victim's browser to send requests to the Ring website. Second, the attacker would need to be in the middle of the victim's and server communications (MITM attack) and be able to modify the victim's requests (*i.e.,* an active MITM attack scenario). Since the Ring servers already prevent downgrade attacks, the victim would also need to be in a browser or on a client that only supports

the less secure protocols (TLS v1.1 or v1.0) or less secure cyber suites within TLSv1.2. Due to these hard conditions, it is noted in the tool's description by the team that wrote "Scalable scanning and automatic classification of TLS padding oracle vulnerabilities" [114], that there have not been any reports of these types of attacks in the wild [97].

Regarding the OWASP cheat-sheet:

- As discussed in Section 4.1.7, all traffic in the Ring ecosystem is encrypted using TLS.
- Some of the identified cryptographic algorithms are considered weak but that depends on them being configured incorrectly. Results from vulnerability scans performed did not reveal any vulnerabilities in the Ring system. Those scans include vulnerabilities associated with use of weak cryptographic algorithms. Therefore, it can be concluded that they are configured correctly and are not a security threat for now.
- 2048-bit RSA keys are used and are rotated in each session.
- All traffic is encrypted and all scans indicated that encryption is enforced. No means of bypassing enforced encryption was found.
- Browsers handle certificate validation for the web application. Additionally, the mobile application has in app certificate pinning.
- Initialization vectors are not reused.
- Only RSA 2048-bit cryptographic keys are used.
- No reoccurring or predictable behavior could be observed that indicates insufficient randomness. However, this cannot be further verified from a black-box perspective.
- No deprecated hash functions were observed, only the SHA256 hash function was used.
- No deprecated padding methods observed or vulnerabilities found relating to deprecated padding methods.
- No revealing cryptographic error messages observed or vulnerabilities found relating to side channel or padding oracle attacks.

In conclusion, the Ring servers are not vulnerable to any known vulnerabilities regarding traffic negotiation and encryption. They are backward compatible to TLS versions 1 and 1.1 but can be considered secure when configured correctly, which the scan results indicate. Inspection of traffic and encryption used indicates that Ring does not have any vulnerabilities due to cryptographic failures, and it can be concluded that it is not exposing sensitive data in any way.

4.2.3 W3: Injection

This OWASP threat has moved down from first place in the previous top ten list down to third. Additionally, the previous OWASP Top Ten list contained XML External Entity (XXE) threats and XSS threats specifically, and both of these will be considered in this section.

Injection attacks can vary quite a bit, depending on frameworks, libraries, languages, and context, and the consequences can subsequently be very different as well. What enables most injection attacks is when user-supplied data is trusted, which it should never be. All User-supplied data should be validated, filtered, or sanitized by applications before being used any further.

XSS attacks are a form of injection attacks. They can be classified into three categories; (i) *reflected XSS* (non-persistent XSS), (ii) *stored XSS* (persistent XSS), and (iii) *Document Object Model (DOM) XSS*. Reflected XSS vulnerabilities exist when unvalidated or unescaped user input is used as a part of the final HTML. As an example, an attacker might send a malicious link that contains an injection in the link's query string, and the malicious code is run when the victim loads the page in their browser. Stored XSS is when an attacker can post malicious code that is not properly sanitized and stored server-side and later loaded by the victim when the user visits a page. An example might be that an attacker posts a comment that contains malicious code, which is then run when a victim visits the page that contains the comment. DOM XSS is when user-controlled data is dynamically included in a web page. An example might be that an attacker injects code on a page that is then dynamically loaded without sanitation by a React component. Previously discovered XSS vulnerability in the Ring web application, discussed in Section 1.5, was a reflected XSS vulnerability.

XXE vulnerabilities can be found when users are allowed to upload XML or when XML content includes user-controlled data. In that case, attackers can possibly exploit vulnerable XML processors that do not properly escape malicious inputs which can cause sensitive data exposure or privilege

escalation.

4.2.3.1 Method

OWASP provides cheat-sheets to validate web applications against XSS vulnerabilities and XXE vulnerabilities.

OWASP Testing for XSS vulnerabilities from a black-box perspective [115]:

1. **Detect Input Vectors**: For each page in the web application, the tester needs to identify all user defined variables and how they are input.

2. **Analyze Input Vectors**: In some cases, malicious code can be injected with little to no modifications. In other instances, the code needs to be specially crafted. An example is symbols that can be used to escape DOM elements, such as '<', are blacklisted but could then be replaced with '%3c', which is the same character from the UTF-8 library, just unencoded. The injected code is usually simple and harmless, and it can be easily verified that it gets executed. A simple example might be <script>alert('XXS!')</script>, where the vulnerability is confirmed if an alert pops up when the page is visited.

3. **Check Impact**: In this step, the tester verifies, for each of the tested inputs, if there is a vulnerability, what resources can be accessed, and what impact it can have on the application. The tester should then identify which special characters are not handled correctly.

OWASP Cheat-sheet for XXE testing [116]:

- The application accepts extensible markup language (XML) directly or XML uploads, especially from untrusted sources, or inserts untrusted data into XML documents, which is then parsed by an XML processor.

- Any of the XML processors in the application or Simple Object Access Protocol (SOAP) based web services has document type definitions (DTDs) enabled.

- If the application uses Security Assertion Markup Language (SAML) for identity processing within federated security or single sign on (SSO) purposes. SAML uses XML for identity assertions and may be vulnerable.

- If the application uses SOAP prior to version 1.2, it is likely susceptible to XXE attacks – if XML entities are being passed to the SOAP framework.

To further validate XSS test results, XSS-Radar [117] was used. XSS-Radar is a Google Chrome extension that performs non-intrusive tests to find reflected-XSS vulnerabilities.

Many web application have good client-side protection against user-inputs. To test server-side protection, *Tamper Dev* [118] was used. Tamper Dev is a Google Chrome extension which allows users to catch and edit HTTP/HTTPS requests without the use of a proxy server. This allows users to bypass client-side protection.

4.2.3.2 Result

All user-defined variables and input vectors were identified. This includes both inputs to the web application as well as all pages containing Uniform Resource Locators (URLs) with parameters. All user-inputs on the web application had some sort of client-side filtering or sanitation. The client-side protection did depend on the requested input, for example, inputs for changing the user's first or last name could only contain alphabetic characters, and no symbols or digits were allowed. When client-side protection was bypassed and a symbol added to the payload sent to the Ring server, a HTTP 422 status code error was returned with the message "Invalid Input".

Input vectors were analyzed, with all inputs treated as displayable text, read from JavaScript Object Notation (JSON) files received from the Ring servers and never dynamically rendered directly from user input on the page. Therefore, there were no reflected XSS vulnerabilities discovered which were further supported by the results from XSS-radar, and no DOM XSS vulnerabilities were discovered.

No user-controlled XML nor XML, including user inputs, was identified. Usage of the SOAP was not identified on the web page nor the usage of SAML for sign-in. Therefore, there were no XXE vulnerabilities discovered.

4.2.3.3 Discussion

Client-side protection offers good initial protection for web applications. Although they can be easily bypassed, the Ring servers validate and sanitize the user inputs as well. Only a few non-malicious inputs were tested on the Ring web application to assert that user inputs are validated and sanitized.

Further inputs were not tested for legal reasons, so injection threats cannot be completely ruled out.

Reflected and DOM XSS threats could be thoroughly tested as well as XXE threats and no vulnerabilities were discovered. The web application can be considered safe against these specific threats, although injection threats might exist.

Injection via QR-code is further discussed in Section 4.3.1

4.2.4 W4: Insecure Design

Secure design is one of the foundations of a good system. It is important to differentiate secure design from secure implementation. If it is implemented incorrectly, a secure design can still lead to an insecure system. An insecure design, on the other hand, can be implemented perfectly but still be insecure. To ensure a secure design requires constant consideration during the development cycle.

Most aspects of a secure design cannot be verified from a black-box perspective. User flows can be verified, and the only user flow with identified gain tied to it from an attacker's perspective is the password recovery flow.

Ring accounts do not specifically have password recovery, they have a "Forgotten password" flow which allows users to update their passwords in the case of a forgotten password. Users then need to enter their email address associated with their Ring account. They then receive an email with a link to the Ring web application where they can reset their password. Users are required to create a new password and re-enter it. When the password has been updated, users get an email notification that the password for their account has been updated.

In the URL when resetting the password, there is a "reset_password_token" parameter that contains a 60-character token. This token is only accessible to users by following the link sent to their email and can be assumed to be randomly generated since, for each password change request, the token is different. In this flow, users request a password change, so they cannot authenticate themselves using their password; instead, a token is generated for the password-changing session. When the user sends in their new password, this token is also sent with the request to determine if the password change is authorized. Typically, tokens like this are generated and stored in a database, where additional information is also stored. This information might typically include the token's expiration (how long it is valid and/or has it already been used) and to which user account it is associated.

4.2.4.1 Method

Two types of tests were performed:

- Visits were made to the password-update pages with altered URLs where the token parameter had been set to previously acquired tokens (*i.e.,* tokens that were expired), random tokens, and valid tokens. This is done to see if the response status types were different. If they are, attackers could make a bot that tries random tokens and gathers valid tokens based on responses.

- Entering a new password in the password-update page and submitting previously used tokens, random tokens, and valid tokens. This can be done by catching and editing requests made in the browser with the use of *Tamper Dev* [118].

4.2.4.2 Result

Update-password pages can be visited with expired, random, and valid tokens. The Ring servers do not send different types of responses based on the token.

Entering a new password and sending in an expired token or a random token results in the same error. The server responds with a HTTP 200 status code with the body: errors: [”Invalid Token”], and the page shows the error message “Your reset password token has expired. Please submit a new request below” and asks the user to re-enter their email address and gives a new link to the password reset flow.

4.2.4.3 Discussion

The password reset page can be visited with the correct token, invalid token, and expired token, but there is no way to automate an attack to request pages with random tokens and filter by status code, and see which tokens are valid.

In theory, it can be assumed that an attacker can guess or obtain a token and update the password linked to that token/account. However, this is not a major security concern since the attacker would need to correctly guess a 60-character token, which is infeasible. Additionally, if the attacker guesses the token correctly, the victim would get an error indicating that the token has expired and go through the flow again. The attacker would also not know for which account the password was updated, and even if the attacker did know, the attacker would still need to bypass the 2-factor authentication (2-factor authentication for Ring accounts is discussed further in Section 4.2.7).

Considering the discussion above, the password update flow for Ring accounts is considered secure.

4.2.5 W5: Security Misconfigurations

This threat has moved up in the OWASP top ten list and is now number five on the most recent version of the list. This might not come as a surprise to some people since many companies and developers use configurable software, libraries, frameworks, or platforms instead of developing everything from scratch themselves. This opens the door for more configuration threats.

Similar to many of the threats listed on the OWASP Top Ten list, this threat can be hard to verify from a black-box perspective. Nevertheless, the consequences of security misconfigurations can be serious and lead to glaring vulnerabilities in the system.

The usage of security headers and their configurations can be validated. OWASP provides a list of security headers and what values to set them to for best practises [119]. The following is OWASP's list of headers and values with a short explanation of the header:

1. Strict-Transport-Security : max-age=31536000 ; includeSubDomains

   Forces user-agent (browser) to use HTTPS instead of HTTP.

2. X-Frame-Options : Deny

   Tells browser if the page is allowed to be rendered in a frame (frame, iframe, embed, object)

3. X-Content-Type-Options : nosniff

   Tells browser that Multipurpose Internet Mail Extensions (MIME) types are deliberately set and should not be sniffed/changed.

4. Content-Security-Policy : default-src 'self'; object-src 'none'; child-src 'self'; frame-ancestors 'none'; upgrade-insecure-requests; block-all-mixed-content

   Added security layer that helps developers control content sources on pages. This header is designed to help mitigate and detect attacks such as XSS.

5. X-Permitted-Cross-Domain-Policies : none

   Set rather cross-domain requests originated from documents such as Portable Document Format (PDF) files are permitted.

6. Referrer-Policy : no-referrer

   How much information regarding referrer should be sent with requests in referrer-header.

7. Clear-Site-Data : “cache”,”cookies”,”storage”

   Clears browsing data of requesting website of values selected.

8. Cross-Origin-Embedder-Policy : require-corp

   Prevents documents from loading resources that are not from the same origin.

9. Cross-Origin-Opener-Policy : same-origin

   Prevents documents from sharing a browsing context with documents from another origin.

10. Cross-Origin-Resource-Policy : same-origin

    Blocks resources such as scripts from other origins from being read.

11. Permissions-Policy : accelerometer=(),autoplay=(),camera=(),display-capture=(),document-domain=(),encrypted-media=(),fullscreen=(), …

    Tells the browser which features within it can be used on page.

12. Cache-Control : No-store, max-age=0

    Gives the browser caching instructions.

4.2.5.1 Method

OWASP provides more description of the vulnerability and where possible misconfigurations can be found:

1. Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud servers.

2. Unnecessary features are enabled or installed (*e.g.,* unnecessary ports, services, pages, accounts, or privileges).

3. Default accounts and their passwords are still enabled and unchanged.

4. Error handling reveal stack traces or other overly informative error messages to users.

5. For upgraded systems, the latest security features are disabled or not configured securely.

6. The security settings in the application servers, application frameworks, libraries, databases, *etc.*, are not set to secure values.

7. The server does not send security headers or directives, or they are not set to secure values.

8. The software is out of date or vulnerable.

On this list, items number 1, 5, and 6 cannot be verified from a black-box perspective. Additionally, item number 8 has its own place on the OWASP Top Ten list as "Vulnerable or Outdated Components" and that threat is examined in Section 4.2.6.

Manual testing and scanner tools were used to look for unnecessary features. ScreamingFrog [120] is used to scrape the Ring web application. ScreamingFrog crawls a web application links, images, and files and then returns a list of all the web application's pages. URLs with user specific parameters were identified and attempts to change those parameters to access other users' content, pages, or data were done.

Testing accounts with bots or other tools is too intrusive and hence cannot be run on the Ring web application. Manual tests of the most common accounts and password combinations were performed. Rapid7 [121] has a list of the top ten most common usernames and passwords. Tests were performed with 6 of the most common usernames and all the most common passwords were tried for each, with the following usernames:

- administrator
- Administrator
- user1
- admin
- demo
- Admin

And the following passwords:

- X
- Zz
- St@rt123
- 1
- P@ssw0rd

- bl4ck4ndwhite
- admin
- alex
- .....
- administrator

Error handling was verified manually. Error messages were noted in research for Sections 4.2.2, 4.2.3, and 4.2.4. Additional error handling was noted when manual tests of web applications pages was performed.

Traffic was captured, in Section 4.1.7 the security headers can be seen. Additionally, *Security Headers* scanner [122] was used to scan the web application to further highlight the security headers used in the web application, verify results from traffic sniffing, get an assessment of headers, and check if some information was missed during traffic sniffing.

4.2.5.2 Result

ScreamingFrog scrape of the Ring web application did not reveal any open pages that should require authentication. Manual tests to escalate privileges or gain unauthorized access by tampering with URLs and session ids were also unsuccessful.

Manual testing of the most common username and password combinations did not reveal insecure or default accounts left open.

No revealing error messages were found on web application.

Security headers are sent and they were set to secure values. Additionally, the web application receives a grade of A from the security headers scan on the scale of A+ to F.

4.2.5.3 Discussion

Regarding unnecessary features being enabled, as discussed in the information gathering section, no ports are open on the device itself and as per Section 4.1.2 no ports were accessible during setup of the device. ScreamingFrog scraping revealed 151 pages in the Ring web application. None of which should require authentication. More intrusive scans, that try to gain unauthorized access or spam the web application by guessing open URLs were not tried due to legal reasons. Manual attempts to access unauthorized pages were tried with URL and access token fuzzing, all of which ended up in redirects or generic error handling by Ring.

Error messages noted during manual testing, and during testing in Sections 4.2.2 and 4.2.3 did not reveal any stack traces or potentially useful information.

The absence of default accounts left open or insecure default accounts cannot be fully verified, but manual tests with the most common username and password combinations did not reveal any such accounts.

Security headers were noted to have the following values in captured traffic and verified with securityheaders.com scanners. Additionally, ScreamingFrog further confirmed that these headers were not missing from any of the discovered pages.

- X-Frame-Options : DENY
- Content-Security-Policy : block-all-mixed-content; frame-ancestors 'none'; upgrade-insecure-requests;
- X-Content-Type-Options : nosniff
- Strict-Transport-Security : max-age=7889238

Additionally noted headers from securityheaders.com scan where:

- X-XSS-Protection: 1; mode=block
  XSS filtering is enabled where the browser will prevent rendering if an attack is detected.
- Expect-CT : max-age=604800, report-uri=https://report-uri.cloudflare.com/cdn-cgi/beacon/expect-ct
  How long certificates should be regarded as safe and the Uniform Resource Identifier (URI) where browser should report Expect-CT failures.
- Server : Cloudflare
  Typically has a more revealing value, containing information about the software running on the origin server that handles requests.

The Ring web application does not receive the top grade because it is missing the *Referrer-policy* and *Permissions-Policy* headers. No referral policy can lead to leaked information from the origin page when navigating to unsafe pages, but no pages were found in the Ring application containing referrer information that should be kept safe. Permissions-Policy headers are sometimes considered more important since they also affect browser feature usage from external frames, but framing is already being handled by the *X-Frame-Options* header.

During manual testing and scanning of the web application, no vulnerabilities were discovered or any indication that security might have been neglected or of security features being insecurely configured.

4.2.6 W6: Vulnerable and Outdated Components

The use of components that have known vulnerabilities is quite common and this threat was moved to 6th placed from 9th on the previous OWASP Top Ten list. Using components with known vulnerabilities poses a big security threat, especially considering that they usually run with the same privileges as the application. Furthermore, the vulnerabilities are known, *i.e.,* they are documented, and, in some cases, exploits might have been written and shared online, providing a hacker with all the tools needed to attack the system.

4.2.6.1 Method

Since the source code was not accessible, scanners were used to gain information regarding the technology used, specifically technology reports from *BuiltWith* [123] were used as well as the Google Chrome Extension *Wappalyzer* [124]. This gave some idea of the technology used but a full assessment of the technology used cannot be done from a black-box perspective.

4.2.6.2 Result

The relevant technology identified in Ring's web application consists of:

- Widgets
  - Kustomer - Live chat
  - Qualaroo - Feedback forms and surveys
- eCommerce
  - Shopify
  - Riskified
- Frameworks
  - Adobe Enterprise Cloud
  - Ruby on Rails Token
  - Facebook Domain Verification
  - Express
  - PHP
- Content delivery network
  - CloudFront, Cloudflare
  - CDN JS

  - AJAX Libraries API
  - Amazon S3
- Content management system (CMS)
  - Atlassian Cloud
  - Contentful
  - Wrike
  - Postman
- JavaScript Libraries and Functions
  - FancyBox 3.5.6
  - jQuery 3.4.1, 3.3.1, 1.11.0 and 1.9.1
  - Modernizr
  - core-js
  - html5shiv
  - Polyfill IO
  - lazySizes
  - Webpack
  - MobX
  - lodash
  - React
- Web Servers
  - Nginx and Nginx 1.10
  - Apache
- Operating Systems and Servers
  - Ubuntu

4.2.6.3 Discussion

jQuery 3.4.1 was noted on the Ring website but with country-specific subdomains, such as sa-en.ring.com, jQuery versions 3.3.1, 1.11.0, and 1.9.1 were also noted. Quite a few vulnerabilities are known to jQuery versions prior to 3.5.0 [125]. All discovered vulnerabilities for the identified jQuery versions are XSS vulnerabilities, that require preconditions to be satisfied, specific functions to be used, and Hypertext Markup Language (HTML) being passed from untrusted sources. XSS threats have already been addressed in Section 4.2.3, where no threats were discovered, nor HTML being passed from untrusted sources. Version 3.5.0 of jQuery has patched these issues and has no known vulnerabilities and the same is true for version 3.6.0 (which was

released in March 2021). Ideally, Ring should update jQuery to a version with no known vulnerabilities, but legitimate reasons might be behind not doing it, such as compatibility or other issues. Additionally, updating the jQuery version might not be of high priority if these vulnerabilities have already been addressed or mitigated and they are not relevant to the Ring system.

FancyBox v3.5.6 was noted on the Ring website which has no known vulnerabilities. V3 has been discontinued with the last update being v3.5.7 released in March 2019 and the 4th generation of Fancybox has since been released.

Nginx was noted on the Ring website and on the sub-domain retailacademyeu.ring.com Nginx v1.10 was noted. Version 1.10 has no known vulnerabilities, but is not up to date, being released in April 2016 and the latest version is 1.20, released in April 2021.

Other technology was identified but not their specific versions. Scanners such as Wappalyzer identify technology through various fingerprinting methods, and when technologies or their versions cannot be identified it is due to a lack of information. The lack of information can be a coincidence or Ring might be actively trying to minimize exposed technology information. Methods to minimize information released can include minifying HTML, JavaScript, and Cascading Style Sheets (CSS) which can be done automatically by Cloudflare which was observed on the Ring website.

4.2.7 W7: Identification and Authentication Failures

This threat was called *Broken Authentication* in previous OWASP Top Ten lists, but now also covers vulnerabilities and weaknesses related to identification failures. It has moved down from the second position to seventh. This can be due to increased awareness from developers as well as greater availability and usage of standardized frameworks, such as 0Auth [126] and FIDO Universal Authentication Framework (UAF) [127].

This threat relates to accurately identifying the user's identity, managing sessions properly, and authenticating users. As discussed in Section 1.5, Ring users have previously been victims of credential stuffing attacks and users also noted bad session management, particularly that Ring did not end sessions when passwords were changed.

As noted in Section 4.1.9, 2-factor authentication is required for all Ring user accounts. Users can choose if they receive security codes via email or mobile text, but they cannot opt-out of 2-factor authentication. Ring sends random 6-digit security codes that need to be entered during login to access

accounts. Security codes are also required when users contact customer support to verify their identity and the devices they own. When a successful login is done, the device used is added under the "Authorized Client Devices" page for the Ring account. That page contains the list of devices authorized to access the account, the date of authorization, and if they were used to log in via the web- or mobile interface. Devices can be removed from the list by users at any time.

4.2.7.1 Method

Force browsing, URL parameter tampering, and session ID tampering were already discussed in Section 4.2.5.

OWASP provides a checklist to help identify potential web applications with identification and authentication weaknesses:

- Permits automated attacks such as credential stuffing.
- Permits brute force or other automated attacks.
- Permits default, weak or well-known passwords.
- Uses weak or ineffective credential recovery and forgot-password processes, such as knowledge-based answers.
- Uses plain text, encrypted, or weakly hashed passwords data stores.
- Has missing or ineffective multi-factor authentication.
- Exposes session IDs in the URL.
- Does not rotate session IDs after successful login.
- Does not properly invalidate session IDs. User sessions or authentication tokens are not properly invalidated during logout or following a period of inactivity.

Two items on this list will not be discussed further. First, the permission of default, weak, or well-known passwords was already discussed in Section 4.1.1. Second, password storage with plaintext, encrypted, or weakly hashed passwords cannot be verified from a black-box perspective but further information regarding the cryptographic strength of the system can be found in Section 4.2.2.

The credential recovery and forgot-password process were discussed in Section 4.2.4; hence, it is not discussed further in this section.

Automated attacks are too intrusive, so manual tests were performed to assess rate-limiting to logins. This was done by attempting logins to a legitimate Ring account, *i.e.,* using the correct email of a Ring account, but incorrect passwords were attempted. Two types of tests were performed:

- Attempting to log in, with the incorrect password, where the password was changed for each attempt. More time then passes between each attempt and this would replicate a real user that has forgotten his or her password.
- Attempting to log in, with the same incorrect password each time. Attempts are made with as minimal amount of time between attempts as possible. This replicates a bot attack rather than a real user.

User-agent spoofing was done in browser dev-tools and by modifying browser requests with *ModHeader* [128].

4.2.7.2 Result

When an incorrect password is entered, the Ring application does not provide error codes indicating that the user only has a certain number of attempts left. The two types of tests were performed on the login page in the web application:

- When attempting to log in with a different incorrect password, no limit was reached where the Ring application black-listed the user-agent, network, or Ring-account from being signed into.
- When attempting to login with the same incorrect password multiple times within a short time frame, an error message was received after 15 attempts stating “Too many attempts. Please wait a few minutes, then try again.”. After the error was produced, a few more tests were performed:
  - An attempt was made to login with the correct email address and password, which produced the same error.
  - An attempt was made to login to another legit Ring account with the correct email address and password, which produced the same error.
  - Login attempts were made for the 2 different accounts with spoofed user-agents and correct credentials and via different browsers, all resulting in the same error.
  - A login attempt was made on the original account, in the same browser, with its authentic user-agent but on a different network. The login attempt was successful.
  - The login timeout lasted more than 5 hours. When a login was attempted with the correct credentials, 5 hours after the timeout was started, the same login error was still produced.

An authorized device was created by logging in on a Windows 10 machine in the Edge browser. Using the same machine and staying on the same network, login attempts were made from a Chrome browser, spoofing the user-agent to look identical to the authorized Edge browser. 2-factor authentication was not bypassed, and a new authorized device was listed for the Ring account.

All accessible pages on the Ring website were examined and the session ID was never exposed in the URL. Logging in and out of Ring account revealed that the session ID was updated on each login. The session ID (cookie) was grabbed after a successful login and then the session was ended. Subsequently, making requests with the obtained cookie led to a redirect to the login page. Multiple sessions on the same account were started on different devices, and updating the password for the account resulted in all sessions being ended and a redirect to the login page. This behavior indicates that the session ID is invalidated properly.

4.2.7.3 Discussion

As discussed in Section 4.1.1, the required strength of passwords for Ring accounts makes brute-forcing infeasible. Authentication limiting was also observed when the web application was spammed. It was noted that authentication limiting was not tied to Ring accounts or user-agents, but rather to networks and the rate of requests. This means that attacks that attempt to exploit multiple accounts, such as password spraying or credential stuffing are not more likely to bypass attack detection by Ring. This also indicates that slowing down the rate of requests would make them more likely to succeed but that also means that brute-force attacks become even more infeasible. Since timeouts were tied to networks and not accounts or user-agents, attackers could potentially spoof their IP addresses and rotate each time a timeout has been issued to their current IP. Rate-limiting requests and spoofing IP addresses might have an effect, but it should be noted that Ring might have attack detection with higher thresholds that were not triggered during testing that would counter these issues.

Bypassing 2-factor authentication could not be done. Moreover, it can not be opted out of by users. Additionally, spoofing user-agents to replicate authorized devices and browsers on the Ring account was unsuccessful. Inspecting the Ring server response for the *Authorized client devices* page revealed that all information for both previously authorized devices and the spoofed devices were identical except for the device's IDs and their registration times. This means that the device, browser, and network information was all

the same, but the spoofed devices were still recognized as new devices and required 2-factor authentication to login to the Ring account.

In conclusion, automated attacks such as brute-force attacks are infeasible. Other automated attacks, such as credential stuffing might still pose a threat with rate-limited requests and/or rotated IP addresses, but since there is no way of bypassing 2-factor authentication, attackers cannot access a victim's Ring account.

All session management has been updated and is properly done. Session IDs are never exposed in the URL and are invalidated on logout and password change. This also applies to password change tokens generated which are properly invalidated and the password recovery process is deemed secure.

4.2.8 W8: Software and Data Integrity Failures

Software and data integrity failures mostly refer to internal mechanisms and processes tied to an application. This could, for example, be tied to verifying the quality of code or content delivery and integration, updating processes, and ensuring dependencies, such as libraries and plugins and the sources from which they consume can be trusted.

Verification of such mechanisms, processes, and usage cannot be verified from a black-box perspective, but loosely relates to the update mechanism of the device discussed in Section 4.1.4 and the usage of vulnerable and outdated components discussed in Section 4.2.6.

4.2.9 W9: Security Logging and Monitoring Failures

From a black-box perspective, access to security logs, logged events, or log thresholds cannot be obtained. Additionally, no logs have been found or privileges escalated to gain access on the IoT device itself or in the web applications.

It was noted in Section 4.2.7 that a threshold could be reached by attempting to login too many times with the incorrect password in the web application. That would also indicate that logs are being kept but cannot be verified from the web application's perspective.

However, when certificate-pinning was bypassed, unencrypted requests in the mobile app could be observed. Within the mobile apps traffic, quite a few log requests to the ring API were noted. Each log entry in the requests body had the label *@loglevel* which had a value of either *ANALYTICS* or *ERROR*. Logged analytic events would depend on actions taken and the duration of

those actions and logged errors include both user introduced errors, such as trying to sign in with an incorrect password as well as application failures.

It was also noted that with each failed login attempt, logs were being sent to Ring. Each failed login attempt was logged with three error entries and one analytic entry. The three error entries were further tagged as ClientsApi, PushTokenMonitor and RingAuth. The full payload being sent in with each log request can be seen in Appendix A.

This insight into the mobile application log requests can give us some idea about Rings logs. If they are logging requests as they come into the API and if thresholds or account suspensions depend on that, the logs sent in from the mobile application, or both.

4.2.9.1 Method

There were two tests performed on the mobile application, all while capturing its unencrypted traffic with mitmproxy[67].

1. Same test was performed on the web application, where login to the Ring user account is attempted with an incorrect password, 30 times. In each attempt the password is different.

2. Same test as in #1, except this time all log requests are intercepted and dropped so they don't make it to the Ring servers.

4.2.9.2 Result

Results from the first test were the same as in the web application, after 30 login attempts, the correct password was used and the Ring API responded with a *406 - Not Acceptable*. Login had been suspended and was not possible until the next day.

The outcome was different for test number two, where after 30 failed login attempts, login was successful when the correct password was entered.

4.2.9.3 Discussion

This means that the account suspension depends entirely on the logs sent in from the mobile application. It can be noted that right after the successful login, a loggout was performed and the interception of logs stopped. Right away a log request was sent to the Ring API containing all the failed login attempts from the previously performed test and login was immediately suspended, where Ring responded to the first login attempt with an incorrect

password with a *406 - Not Acceptable*, which it had previously only done when login was attempted with the correct password following 30 failed attempts.

This result gives some insights into Ring logging, but it is worth mentioning that even though unlimited attempts can be made to login without the user account being suspended, it does not pose a big security threat to users. As was mentioned in Section 4.1.1 password requirements are strong which makes brute-forcing passwords unfeasible, and even if a brute-force attempt is successful, attackers would still need to bypass the 2-factor authentication.

4.2.10 W10: Server-Side Request Forgery (SSRF)

SSRF threats can be hard to mitigate or easily missed by developers. They are even harder to verify when white-hat penetration testing from a black-box perspective. Verification of network layer mitigation, such as separations of networks to minimize the effects of a successful attack and enforcement of deny by default firewall rules, cannot be verified. On the application layer, it is also almost impossible to verify whether the application is safe, for instance by checking if URL schemas, ports, and destinations are enforced with a verified white-list. Sanitation and validation of user input were discussed in Section 4.2.3, which is one aspect of being secure against SSRF attacks. Additionally, no user input in URLs has been noted in the Ring application, nor the functionality of uploading or reviewing uploaded files from the user or directly from the server which could be a potential attack vector for SSRF. The web application could not be tested further for any SSRF vulnerabilities.

4.3 STRIDE Threats

Identified STRIDE threats can be seen in Table 3.6. Of the seven threats identified, five have already been addressed.

- **S1** in Section 4.2.7
- **S2** in Section 4.1.4
- **S4** in Section 4.1.2
- **S5** in Section 4.2.2
- **S7** in Section 4.2.7

Threat *S6*, *i.e.,* spamming doorbell with requests to empty battery or disrupt regular functionality, can not be penetration-tested, since no means of communicating with the doorbell, outside of the web and mobile applications

have been discovered. The Ring doorbell is therefor regarded as safe against this threat. The final threat to be examined is therefor *S3* which will be covered in this section.

4.3.1 S3: QR Code Injection

During the setup of the Ring device, users have the option to scan a QR-code that is located on the back of the device. When the QR-code is scanned, it can be noted that the data stored is in the form of a URL*. It can be further noted that it is a query string with five variables: ”m”, ”d”, ”n”, ”b” and ”v”. This URL also servers as a deep-link, *i.e.,* if it is scanned with the mobile camera, not within the Ring app, users get the option to open the link in the Ring app or in a browser. Therefore this QR-code serves as both an entry point to the mobile application as well as providing parameters used during the setup of the device.

4.3.1.1 Method

QRGen [129] is a tool that can be used to generate QR-codes. It has 8 word-lists that can be used to generate QR codes with malicious payloads. The word-lists contain payloads to test for attacks such as XSS, XXE, and Structured Query Language (SQL)-injections. Additionally, a custom word-list can be created which QRGen uses to generate QR-codes.

A custom word-list was created, some of the strings did not hold malicious intent, but rather removed variables or slightly changed values of the original Ring QR-code. Other strings contain attempts to inject code. The full custom word-list can be seen in Appendix B. For each of the generated QR-code, three tests were performed and an additional fourth test was performed with a portion of the QR-codes:

1. Scan the QR-code within the setup flow in the Ring mobile application.
2. Scan the QR-code from the mobile camera while the Ring app is running in the background.
3. Scan the QR-code from the mobile camera and use the link to launch the Ring application.
4. Scan the QR-codes while not being logged into an account in the Ring application.

*https://ring.com/s?m=54E01913B395&d=rvd3&n=BHRG62010CN003488&b=t&v=2

Note that QR-codes that were not on the custom word-list were not scanned outside the Ring application, since they do not have the ”https://ring.com” prefix and are therefore not recognized as deep-links, which means the Ring application is not launched when they are scanned.

4.3.1.2 Result

For all QR-codes, the value seems to be sanitized and no injection was successful. Nevertheless, some unexpected behavior was noted when values were tampered with.

1. Value ”m”
    - **Setup mode**: If the value was changed at all, the setup flow would go like normally, except that the Ring AP was never found. If the value was changed to something short, like ”test” or ”null”, the Ring app would crash when trying to connect to the Ring AP.
    - **Deep-link**: The app would always open up on the setup screen, and then the same behaviour would occur as is described when in ”Setup mode”.
2. Value ”d”
    - **Setup mode**: Flow went like normally and could be completed.
    - **Deep-link**: If the app was running in the background, everything worked as expected, but if the app was closed, it would crash on launch.
3. Value ”n”
    - **Setup mode**: Flow went like normally and could be completed.
    - **Deep-link**: Independent of if app was running in the background or not, when the value was changed, the app would open on a default screen, not in the setup flow. When value was completely removed and the app was not running, it would crash on launch.
4. Values ”b” and ”v”
    - Changing or removing these values had no affect.

To further examine why the Ring application was crashing, adb and *logcat* [130] was used to dump error logs from the Android mobile device. The fatal error stack trace can be seen in Figure 4.4. It was mentioned in Section 3.4 that obfuscated source code from the Android Ring mobile application was captured. Using the source code, this stack trace could be examined further. The functions referenced in Figure 4.4 can be seen in Appendix C.

Figure 4.4: Error logs from Ring App crash

It was noted that the Ring application is crashing when it is building the SSID of the Ring AP that it wants to connect to. It was noted in Section 3.2 that the postfix of the Ring's AP SSID is the second half of the Ring's MAC address. It was then noted that the value of "m" in the Ring's original QR-code looks like the noted MAC address for the Ring doorbell, except that the second character is 6 here as opposed to 4 in the scan results. It was concluded that the value of "m" is the MAC address of the Ring AP and another test was attempted.

A mobile hotspot was set up with the name *Ring-{XXXXXX}* where the X's are the last six characters of the MAC address of the device hosting the hotspot. A QR-code where the value of "m" had been set as the hotspot's host's full MAC address was then scanned and setup continued. This resulted in the Ring application instructing the mobile device to connect to the hotspot, which it did successfully. The setup flow within the Ring application showed a loading state as if it were connecting to the Ring AP, even though it was already connected to the hotspot, until the user canceled the setup flow. Further testing revealed that the value of "m" was actually not used when connecting to the hotspot, *i.e.,* setting the value of "m" to "123456RANDOM" meant that the mobile device tried to connect to an AP with the SSID *Ring-random*.

A final test was done to see what request the Ring mobile application makes and if the regular setup flow would continue if there is a service listening on the hotspot it connects to. As mentioned in Section 4.1.2 the only open port during setup is port 443. It can also be mentioned that the Ring mobile application always makes a request to address "192.168.240.1" independent of if that is the default gateway on the AP or hotspot it connects to. Therefore a simple Express JavaScript (JS) [131] server was setup to listen on address

”192.168.240.1” on port 443 on the device that hosted the hotspot that the mobile device connected to when QR-code was altered. This simple server can be seen in Appendix D. It was noted that as soon as the mobile device connects to the hotspot, a GET request is made to ”/ring/system/config/network”. Different variations of replies to that GET request were tested but each resulted in the Ring mobile application attempting the same request repeatedly and a loading state is shown by the application’s interface. As was mentioned in Section 1.5.1, during setup anyone connected to the Ring AP could navigate to ”/gainspan/system/config/network” and see wireless network configurations in XML format. This reply was also attempted from the Express server on the hotspot without any difference in the Ring mobile application behaviour.

4.3.1.3 Discussion

As mentioned, it can be concluded that the value of ”m” is the MAC address of the Ring AP. Functionality stayed the same when the value of ”d” was kept as ”rvd3” or changed to ”rvd2”. References to both of these values can be found in the Androids application source code. Both are a part of string names containing URLs, seemingly used as endpoints to fetch *.mp4* files shown in the setup flow. No further information was found on variable ”n”. However, since it only affected the Ring app when scanned outside the app and the landing page when the app was opened, it is assumed to have something to do with routing within the app or fetching information needed for the setup flow, similar to variable ”d”. No assumptions could be made about the values of ”b” and ”v”.

Regarding being able to trick the Ring application into connecting to another AP than the Ring AP during setup, it can be somewhat assumed that this is an accepted risk by Ring, since the app needs to be able to recognize what AP it should connect to during setup.

It can also be noted that from an adversary’s perspective, this type of attack would be extremely difficult to pull off. The adversary would need to replace the QR-code on the back of the camera and force the victim to redo their setup. Then they would need to have an active AP in range of the victim. After the victim connects to the adversary’s AP, the adversary would need to construct a message and send it to the victim’s mobile phone so that the setup flow continues. If all these steps are completed, then the adversary could read the victim’s Wi-Fi credentials when they finish the setup flow, since they would be sent to the adversary’s AP instead of the Ring AP. It will be noted, that this attack does not necessarily need to involve the QR-code, since the same

result could be achieved by sending a user the deep-link that is stored within the QR-code, but the user would still need to go through the setup flow after pressing such a link, and the attacker would still need to be in range.

Chapter 5

Discussion

Two states were observed and tested on the Ring doorbell, setup state and post-setup state. Results of scans in Section 4.1.2 show that only one port is open on the doorbell during setup indicating that an SSL certificate is needed to connect to that port. Results from captured traffic in Section 4.1.7 and from QR-code injection in Section 4.3.1 also show that the Ring mobile application constructs the SSID of the Ring AP during setup, to know which network to connect to, and establishes a secure connection with it. This process can be noted to be somewhat insecure since the MAC address obtained from the scanned QR-code is not validated at all before a connection is made to the Ring AP. However, as was noted the likelihood of a successful attack using this vulnerability is low since a lot of preconditions need to be met. Additionally, from an attacker's perspective, the only information gained from such an attack would be the victim's SSID and credentials.

It was mentioned in Section 3.2 that post setup all ports on the Ring doorbell are closed or filtered and that spoofing IP addresses, source ports, and MAC address did not result in any further information gain. Furthermore, in the captured traffic Section 4.1.7, it was observed that as soon as the Ring doorbell connects to the internet, it establishes a connection with the Ring servers and keeps that connection alive. This indicates that the Ring doorbell actually does not need to keep any ports open since it does not have to listen for any incoming connections, as it only initiates outgoing connections. This approach considerably reduces the attack surface of the Ring doorbell. From a security perspective, this is a desired feature of a device, but it greatly limits areas of exploration from a research perspective. Since there was no means of communicating with the Ring doorbell itself, the only thing left to consider was the traffic flowing to and from it. All captured traffic was deemed secure

and since the Ring Doorbell seems to connect to the Ring Servers and keep the connection alive, MITM attacks such as ARP poisoning are not effective against it, as was observed in Section 4.1.7. Following these observations, it can be further noted that the update mechanisms were secure, as discussed in Section 4.1.4.

With no more options to consider with regard to the Ring Doorbell, the security research could be expanded to the Ring ecosystem by further examining the Ring mobile and web applications. The traffic from the Ring applications was thoroughly inspected, to the point where further tests would have been too intrusive, in search of cryptographic failures or misconfiguration without any significant findings.

The web application was closely examined for injection vulnerabilities both manually and with the help of tools. Session management and designs of flows such as the *password reset flow* were especially considered without discovering any means of exploiting the current setup. With the help of tools and browser extensions, an attempt was made to identify the tech-stack used to build the Ring web application in Section 4.2.6. Only a portion of the stack could be identified with the recognized libraries and frameworks not having any known vulnerabilities. Additionally, with only a portion of the tech-stack identified, it was suggested that Ring might be actively concealing some of the technology they use.

As has been seen from the tests performed and the observations made in this project, Ring seems to follow security best practices and be an overall secure device. That being said, it is not only the security of the device and the applications that make them secure but also the fact that Ring forces its users to follow security best practices as much as possible. Examples of this were seen in Section 4.1.1 where it was mentioned that Ring enforces strong password requirements and does not allow passwords to be reused, in Section 4.1.9 it was mentioned that users are required to use 2-factor authentication and in Section 4.2.7 it was noted that users have an overview of which devices have been used to access their account, and they can manage those devices.

Chapter 6

Conclusions and Future work

This chapter states the conclusions of the project and its limitations. Suggestions for future work are discussed. Lastly, reflections regarding this project work are discussed.

6.1 Conclusions

The goal of this project work was to answer the question: How secure is the Ring doorbell 3 Plus? It has now been demonstrated that all previously discovered vulnerabilities in the Ring doorbell have been fixed and privacy concerns regarding Ring's handling of private user data have also been addressed. Additionally, there were no major new vulnerabilities discovered during this project work; therefore, it is safe to assume that the Ring doorbell 3 Plus is very secure.

It can never be stated with absolute confidence that a system is completely secure; however, with the constructed threat model and threats examined in this project it can be noted that Ring has done an excellent job of keeping the attack vector small, limiting points of entry, and following good security practices.

6.2 Limitations

This project was performed from a black-box perspective and that entails some disadvantages as was mentioned in Section 2.3.1. Mainly this meant that tests are built on assumptions and large parts of the system cannot be tested without the permission of Ring. This of course means that the security evaluation

cannot be as thorough and the conclusion that the system is indeed secure has to come with the caveat that the analysis was performed from a black-box perspective.

Additionally, testing assumptions is a time-consuming task which means that not all aspects of the system could be fully examined and had to be considered as out of the scope of this project.

6.3 Future work

As was mentioned in the thesis, with traffic being securely encrypted, using a pinned certificate on the Ring doorbell, the firmware of the Ring doorbell could not be captured. With hardware attacks being outside the scope of the thesis, the most likely means of capturing the firmware and/or the pinned certificates were also missed. As Amazon themselves suggests, the IoT device'a identity needs to be added somewhere in the production chain [132]. This could mean that either the certificates themselves are added or that an identity along with some form of authorization tokens are added to the device in one of the production chain steps, as seems to be the case with the Ring Alarm system previously mentioned [71]. In the latter case, it would mean that the Ring Doorbell initially connects to the Ring servers *without* a certificate and downloads the certificates along with its initial firmware update. This implementation leaves the door open for a MITM attack and gives researchers and attackers the chance to potentially capture partial firmware and certificates. If the certificates are added to the device during production, the only means to capture them and the firmware would be by extracting them from the hardware. In a teardown of the Ring Doorbell [133] it looks as though it contains a TI MSP430 processor with model number FR2155, which includes both JTAG and UART interfaces [134]. Potential means of extracting the firmware from the Ring doorbell could therefore be via a JTAG connection and dumping flash memory or extracting it via the UART port [135].

Another aspect that would have been interesting to dive deeper into is QR-code injection. To complete the injection, by sending requests from the "malicious" AP to the mobile application to see if the rest of the setup flow could have been triggered instead of the flow stopping in a loading state where the application says it is trying to connect to the Ring AP.

If appropriate permission was obtained from Ring, then there would be a lot of interesting work that could be done, mainly testing the Ring servers and conducting more intrusive tests on the web and mobile applications.

6.4 Reflections

The goal of many security researchers is not only to find and report vulnerabilities but also to raise awareness and the security standard set by companies. This is also true for this project, where the underlying goal was to raise awareness and not simply evaluate the security of the specific device under inspection. It is important that regular consumers not only consider features, prices, or aesthetics when buying tech products but also consider security. Both in the sense of what products are considered secure but also how they can be set up and used securely.

Security researchers can also find themselves in an ethical dilemma when vulnerabilities are discovered, of how those vulnerabilities should be disclosed. There are essentially two ways to disclose vulnerabilities, *full disclosure* and *responsible disclosure*. The former is when researchers disclose their findings without a warning to the vendor of the product they found the vulnerability in, giving them no time to patch the vulnerability before it is released to the public. The latter is when a researcher informs the vendor about the vulnerability and a time frame is negotiated during which the vendor has an opportunity to release a fix for the vulnerability before knowledge of the vulnerability is released to the public. There is a case to be made for either method and the approach taken can depend heavily on the context of the discovery. The vulnerability found during this research project can be considered minor, and it might even be a conscious system architectural decision. Therefore, it was decided to notify Ring of the findings without delaying the publication of this report, *i.e., full disclosure*.

References

[1] Cisco, "Cisco annual internet report (2018-2023) white paper," Cisco, Tech. Rep., 2018. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html [Page 1.]

[2] K. L. Lueth, "State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time," IoT Analytics, Tech. Rep., November 2020. [Online]. Available: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/ [Page 1.]

[3] I. Lee and K. Lee, "The internet of things (IoT): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015. doi: https://doi.org/10.1016/j.bushor.2015.03.008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007681315000373 [Page 1.]

[4] P. P. Ray, "A survey of IoT cloud platforms," *Future Computing and Informatics Journal*, vol. 1, no. 1, pp. 35–46, 2016. doi: https://doi.org/10.1016/j.fcij.2017.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2314728816300149 [Page 1.]

[5] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, "Big IoT data analytics: Architecture, opportunities, and open research challenges," *IEEE Access*, vol. 5, pp. 5247–5261, 2017. doi: 10.1109/ACCESS.2017.2689040 [Page 1.]

[6] E. Ahmed, I. Yaqoob, I. A. T. Hashem, I. Khan, A. I. A. Ahmed, M. Imran, and A. V. Vasilakos, "The role of big data analytics in internet of things," *Computer Networks*, vol. 129, pp. 459–471, 2017.

doi: https://doi.org/10.1016/j.comnet.2017.06.013 Special Issue on 5G Wireless Networks for IoT and Body Sensors. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128617302591 [Page 1.]

[7] L. Chettri and R. Bera, "A comprehensive survey on internet of things (IoT) toward 5g wireless systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020. doi: 10.1109/JIOT.2019.2948888. [Online]. Available: https://ieeexplore.ieee.org/document/8879484 [Page 1.]

[8] P. Collela, "5G and IoT: Ushering in a new era," Ericsson, Tech. Rep., March 2017. [Online]. Available: https://www.ericsson.com/en/about-us/company-facts/ericsson-worldwide/india/authored-articles/5g-and-iot-ushering-in-a-new-era [Page 1.]

[9] D. Goodin, "When coffee makers are demanding a ransom, you know IoT is screwed," *ars Technica*, Sep. 2020. [Online]. Available: https://arstechnica.com/information-technology/2020/09/how-a-hacker-turned-a-250-coffee-maker-into-ransom-machine/ [Page 2.]

[10] A. Owaida, "50,000 home cameras reportedly hacked, footage posted online," *We Live Security*, Oct. 2020. [Online]. Available: https://www.welivesecurity.com/2020/10/14/50000-home-cameras-reportedly-hacked-footage-posted-online/ [Page 2.]

[11] J. Fruhlinger, "The Mirai botnet explained: How teen scammers and cctv cameras almost brought down the internet," *CSO*, Mar. 2018. [Online]. Available: https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html [Page 2.]

[12] P. Engebretson, *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*, 2nd ed. Elsevier, 2013. ISBN 978-0124116443 [Pages 2 and 11.]

[13] D. Kostadinov. (2016) Ethical hacking vs. penetration testing. [Online]. Available: https://resources.infosecinstitute.com/topic/ethical-hacking-vs-penetration-testing/ [Page 2.]

[14] L. Irwin. (2020) Ethical hacking vs penetration testing: what's the difference? [Online]. Available: https://www.itgovernance.eu/blog/en/ethical-hacking-vs-penetration-testing-whats-the-difference [Page 2.]

[15] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the internet of things (IoT)," *IEEE Internet Initiative*, vol. 1, no. 1, pp. 1–86, 2015. [Online]. Available: https://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Issue1_14MAY15.pdf [Page 3.]

[16] K. K. Patel, S. M. Patel *et al.*, "Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges," *International journal of engineering science and computing*, vol. 6, no. 5, 2016. doi: 10.4010/2016.1482 [Page 3.]

[17] BBC, "Santa hacker speaks to girl via smart camera," *BBC*, Dec. 2019. [Online]. Available: https://www.bbc.com/news/technology-50760103 [Page 3.]

[18] ——, "Mirai botnet: Three admit creating and running attack tool," *BBC*, Dec. 2017. [Online]. Available: https://www.bbc.com/news/technology-42342221 [Page 3.]

[19] ——, "Fiat Chrysler recalls 1.4 million cars after Jeep hack," *BBC*, Jul. 2015. [Online]. Available: https://www.bbc.com/news/technology-33650491 [Page 3.]

[20] H. Badran, "IoT security and consumer trust," in *Proceedings of the 20th Annual International Conference on Digital Government Research*. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3325112.3325234. ISBN 9781450372046 p. 133–140. [Online]. Available: https://doi.org/10.1145/3325112.3325234 [Page 3.]

[21] M. Hung, "Leading the IoT, Gartner insights on how to lead in a connected world," *Gartner Research*, vol. 1, pp. 1–5, 2017. [Page 4.]

[22] California State Legislature, "Sb-327 information privacy: connected devices," 2018, https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180SB327. [Page 4.]

[23] Matt Warman, "Proposals for regulating consumer smart product cyber security - call for views," 2020, https://www.gov.uk/government/publications/proposals-for-regulating-consumer-smart-product-cyber-security-call-for-views/proposals-f

or-regulating-consumer-smart-product-cyber-security-call-for-views#security-requirements. [Page 4.]

[24] Pen Test Partners. (2016) Steal your Wi-Fi key from your doorbell? IoT WTF! [Online]. Available: https://www.pentestpartners.com/security-blog/steal-your-wi-fi-key-from-your-doorbell-iot-wtf/ [Page 6.]

[25] Bitdefender, "White paper: Ring Video Doorbell Pro under the scope," Bitdefender, Tech. Rep., 2019. [Online]. Available: https://www.bitdefender.com/files/News/CaseStudies/study/294/Bitdefender-WhitePaper-RDoor-CREA3949-en-EN-GenericUse.pdf [Page 7.]

[26] k0t, "ring.com Cross Site Scripting Vulnerability," *Open Bug Bounty*, Jan. 2017. [Online]. Available: https://www.openbugbounty.org/reports/207787/ [Page 7.]

[27] A. Carman, "Ring's smart doorbell doesn't immediately revoke access when an account password changes," *The Verge - Circuit breaker*, May 2018. [Online]. Available: https://www.theverge.com/circuitbreaker/2018/5/11/17345972/ring-smart-doorbell-password-change-revoke-app-permission-access [Page 7.]

[28] C. Mihalcik, "Thousands of Ring owners had personal info exposed in data leak, report says," *CNET*, Dec. 2019. [Online]. Available: https://www.cnet.com/news/privacy/thousands-of-ring-owners-had-personal-info-exposed-in-data-leak-report-says/ [Page 8.]

[29] Ring LLC. (2019) Ring's Services have not been compromised - here's what you need to know. [Online]. Available: https://blog.ring.com/about-ring/rings-services-have-not-been-compromised-heres-what-you-need-to-know/ [Page 8.]

[30] "CVE-2019-9483." Available from MITRE, CVE-ID CVE-2019-9483., Feb. 2019. [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9483 [Page 8.]

[31] Cision. (2019) Dojo By BullGuard Cybersecurity Experts Identify Major Vulnerability In Amazon's Ring Video Doorbell. [Online]. Available: https://www.prnewswire.com/news-releases/dojo-by-bullguard-cybersecurity-experts-identify-major-vulnerability-in-amazons-ring-video-doorbell-300803283.html [Page 8.]

[32] D. L. Hoffman, T. P. Novak, and M. Peralta, "Building consumer trust online," *Communications of the ACM*, vol. 42, no. 4, pp. 80–85, 1999. doi: 10.1145/299157.299175 [Page 9.]

[33] J. M. Blythe, S. D. Johnson, and M. Manning, "What is security worth to consumers? investigating willingness to pay for secure internet of things devices," *Crime Science*, vol. 9, no. 1, pp. 1–9, 2020. [Page 9.]

[34] TRUSTe, "TRUSTe/National Cyber-Security Alliance: U.S. consumer privacy index," TRUSTe / National Cyber Security Alliance, San Francisco, Tech. Rep., 2016. [Online]. Available: https://staysafeonline.org/wp-content/uploads/2017/09/TRUSTe-National-Cyber-Security-Alliance-U.S.-Consumer-Privacy-Index-2016-Infographic.pdf [Page 9.]

[35] B. Budington. (2020) Ring Doorbell App packed with third-party trackers. [Online]. Available: https://www.eff.org/deeplinks/2020/01/ring-doorbell-app-packed-third-party-trackers [Page 9.]

[36] G. Weidman, *Penetration testing: a hands-on introduction to hacking*. No Starch Press, 2014. ISBN 978-1593275648 [Pages 11 and 12.]

[37] Nmap, "Nmap: the Network Mapper - Free Security Scanner," https://nmap.org/, 2021, accessed: 2021-02-27. [Page 13.]

[38] ——, "Nmap: Port Scanning Basics," https://nmap.org/book/man-port-scanning-basics.html, 2021, accessed: 2021-02-27. [Page 13.]

[39] ——, "Nmap: Port Scanning Techniques," https://nmap.org/book/man-port-scanning-techniques.html, 2021, accessed: 2021-02-27. [Page 14.]

[40] ——, "Nmap: OS Detection," https://nmap.org/book/man-os-detection.html, 2022, accessed: 2022-12-3. [Page 14.]

[41] ——, "Nmap: the Network Mapper - Free Security Scanner," https://nmap.org/book/firewall-subversion.html, 2021, accessed: 2021-02-27. [Page 14.]

[42] ——, "Nmap: the Network Mapper - Free Security Scanner," https://nmap.org/book/man-bypass-firewalls-ids.html, 2021, accessed: 2022-11-24. [Page 14.]

[43] Nettitude. Using Frida to Bypass Snapchat's Certificate Pinning. [Online]. Available: https://labs.nettitude.com/tutorials/using-frida-to-bypass-snapchats-certificate-pinning/ [Page 15.]

[44] Google Developers – Android Developers Android Studio – User guide, "Android Debug Bridge (adb)." [Online]. Available: https://developer.android.com/studio/command-line/adb [Page 15.]

[45] Open Source, "dex2jar." [Online]. Available: https://github.com/pxb1988/dex2jar [Page 15.]

[46] ——, "Java Decompiler." [Online]. Available: http://java-decompiler.github.io/ [Page 15.]

[47] APKPure. (2014). [Online]. Available: https://apkpure.com/ [Page 15.]

[48] N. Higi, "apk-mitm." [Online]. Available: https://github.com/shroudedcode/apk-mitm [Page 16.]

[49] W. Xiong and R. Lagerström, "Threat modeling–a systematic literature review," *Computers & security*, vol. 84, pp. 53–69, 2019. doi: 10.1016/j.cose.2019.03.010 [Page 16.]

[50] A. V. Uzunov and E. B. Fernandez, "An extensible pattern-based library and taxonomy of security threats for distributed systems," *Computer Standards & Interfaces*, vol. 36, no. 4, pp. 734–747, 2014. [Page 16.]

[51] OWASP. Threat modeling. [Online]. Available: https://owasp.org/www-community/Threat_Modeling# [Page 16.]

[52] S. Nidhra and J. Dondeti, "Black box and white box testing techniques-a literature review," *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 2, pp. 29–50, 2012. doi: 10.5121/ijesa.2012.2204 [Page 17.]

[53] M. E. Khan, F. Khan *et al.*, "A comparative study of white box, black box and grey box testing techniques," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, no. 6, 2012. doi: 10.14569/IJACSA.2012.030603 [Page 17.]

[54] A. Shostack, *Threat Modeling: Designing for Security*, 1st ed. Wiley Publishing, 2014. ISBN 1118809998 [Pages 17, 19, and 22.]

[55] Threatmodeler, "ThreatModeler: Six Steps to Threat Modeling for Secure Data Assets," https://threatmodeler.com/six-steps-to-threat-modeling-for-secure-data-assets/, 2021, accessed: 2021-02-27. [Page 18.]

[56] OWASP, "OWASP: Application Threat Modeling," https://owasp.org/www-community/Application_Threat_Modeling, 2021, accessed: 2021-02-27. [Page 18.]

[57] Microsoft, "Microsoft: The Stride Threat Model," https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20), 2021, accessed: 2021-02-27. [Page 19.]

[58] OWASP. OWASP top ten web application security risks. [Online]. Available: https://owasp.org/www-project-top-ten/ [Pages 21 and 50.]

[59] ——. OWASP Top Ten IoT. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project [Pages 21, 39, and 45.]

[60] Vulscan, "Vulscan: Vulnerability Scanning with Nmap," https://github.com/scipag/vulscan, 2021, accessed: 2021-02-27. [Page 22.]

[61] Metasploit, "Metasploit Project," https://www.metasploit.com/, 2021, accessed: 2021-02-27. [Page 22.]

[62] Ring LLC. Video doorbell 3. [Online]. Available: https://se-en.ring.com/products/video-doorbell-3 [Pages 26 and 27.]

[63] ——. Creating and deleting privacy zones. [Online]. Available: https://support.ring.com/hc/en-us/articles/360028035532-Creating-and-deleting-Privacy-Zones [Page 27.]

[64] ——. Controlling ring devices with users and roles. [Online]. Available: https://support.ring.com/hc/en-us/articles/115005235186-Controlling-Ring-Devices-With-Users-and-Roles [Page 27.]

[65] R. LLC. The protocols and ports used by Ring devices. [Online]. Available: https://support.ring.com/hc/en-us/articles/205385394-The-Protocols-and-Ports-Used-by-Ring-Devices [Page 29.]

[66] Open Source, "AFWall+." [Online]. Available: https://github.com/ukanth/afwall [Page 30.]

[67] Aldo Cortesi, “mitmproxy.” [Online]. Available: https://mitmproxy.org/ [Pages 30 and 74.]

[68] Ring LLC. Terms of service. [Online]. Available: https://se-en.ring.com/pages/terms [Page 30.]

[69] ——. Third party service providers FAQs. [Online]. Available: https://support.ring.com/hc/en-us/articles/360039536952-Third-Party-Service-Provider-FAQs [Page 30.]

[70] Exploitee. (2016) Ring Doorbell. [Online]. Available: https://www.exploitee.rs/index.php/Ring_Doorbell [Pages 37 and 46.]

[71] N. Miles. (2020, September) Inside Amazon’s Ring Alarm System. [Online]. Available: https://medium.com/tenable-techblog/inside-amazons-ring-alarm-system-9731bc519974 [Pages 37 and 84.]

[72] Ring LLC. Creating a strong password and a more secure account. [Online]. Available: https://support.ring.com/hc/en-us/articles/360049104331-Creating-a-strong-password-and-a-more-secure-account [Page 40.]

[73] National Institute of Standards and Technology, “Digital identity guidelines,” U.S. Department of Commerce, Washington, D.C., Tech. Rep. Special Publication (SP) 800-63B, 2021. [Page 41.]

[74] Eric Andrew Young, “OpenSSL: Cryptography and SSL/TLS Toolkit.” [Online]. Available: https://www.openssl.org/ [Page 41.]

[75] SecWiki. (2012) FAQ tcpwrapped. [Online]. Available: https://secwiki.org/w/FAQ_tcpwrapped [Page 43.]

[76] Microsoft, “Security signals,” Microsoft, Tech. Rep., March 2021. [Page 45.]

[77] I. C. Martínez. The key to everything: Firmware on IoT devices. [Online]. Available: https://www.puffinsecurity.com/the-key-to-everything-firmware-on-iot-devices/ [Page 45.]

[78] N. Malviya. (2018, May) IoT firmware analysis — firmwalker. [Online]. Available: https://resources.infosecinstitute.com/topic/iot-firmware-analysis-firmwalker/ [Page 45.]

[79] ——. (2019, Dec.) IoT firmware analysis — firmwalker. [Online]. Available: https://securityboulevard.com/2019/12/anatomy-of-a-firmware-attack/ [Page 45.]

[80] E. Mixon and C. Steele. (2020, Oct.) OTA update (over-the-air update). [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/OTA-update-over-the-air-update [Page 46.]

[81] Ettercap Project, "Ettercap." [Online]. Available: https://www.ettercap-project.org/ [Page 47.]

[82] Wireshark, "Wireshark." [Online]. Available: https://www.wireshark.org/ [Page 48.]

[83] National Institute of Standards and Technology, "Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations: NIST SP 800-52 Rev. 2," U.S. Department of Commerce, Washington, D.C., Tech. Rep., 2019. [Online]. Available: https://csrc.nist.gov/news/2019/nist-publishes-sp-800-52-revision-2 [Page 49.]

[84] National Cyber Security Center (NBSC). (2021) IT Security Guidelines for Transport Layer Security (TLS). [Online]. Available: https://english.ncsc.nl/publications/publications/2021/january/19/it-security-guidelines-for-transport-layer-security-2.1 [Page 49.]

[85] H. Bidgoli, *Encyclopedia of Information Systems: E-J*, ser. Encyclopedia of Information Systems. Academic Press, 2003. ISBN 9780122272400. [Online]. Available: https://books.google.se/books?id=EoxUAAAAMAAJ [Page 51.]

[86] IBM. (2021, Jun.) Pkcs padding method. [Online]. Available: https://www.ibm.com/docs/en/zos/2.4.0?topic=rules-pkcs-padding-method [Page 51.]

[87] M. Bellare, J. Kilian, and P. Rogaway, "The security of cipher block chaining," in *CRYPTO*, 1994. [Page 52.]

[88] R. Heaton. (2013, Jul.) The Padding Oracle attack. [Online]. Available: https://robertheaton.com/2013/07/29/padding-oracle-attack/ [Page 52.]

[89] C. Young. (2019, Mar.) TLS CBC Padding Oracles in 2019. [Online]. Available: https://www.tripwire.com/state-of-security/vert/tls-cbc-padding-oracles/ [Page 52.]

[90] T. A. Nidecki. (2020, Jul.) What is the POODLE attack? [Online]. Available: https://www.acunetix.com/blog/web-security-zone/what-is-poodle-attack/ [Page 52.]

[91] National Institute of Standards and Technology. (2019, Aug.) CVE-2019-5592 detail. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-5592 [Page 52.]

[92] ——. (2019, Feb.) CVE-2019-5592 detail. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-6593 [Page 52.]

[93] F. Valsorda. (2016, May) CVE-2019-5592 detail. [Online]. Available: https://blog.cloudflare.com/yet-another-padding-oracle-in-openssl-cbc-ciphersuites/ [Page 52.]

[94] Digineo GmbH, "SSL-Tools - Poodle Test." [Online]. Available: https://ssl-tools.net/poodle-test [Page 52.]

[95] Qualys Inc, "SSL Server Test." [Online]. Available: https://www.ssllabs.com/ssltest/ [Page 52.]

[96] OWASP. A3:2017-Sensitive Data Exposure. [Online]. Available: https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure [Page 52.]

[97] J. Somorovsky. (2019, Oct.) TLS-Padding-Oracles. [Online]. Available: https://github.com/tls-attacker/TLS-Padding-Oracles [Pages 53 and 56.]

[98] National Institute of Standards and Technology. (2014) CVE-2014-3566. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2014-3566 [Page 54.]

[99] D. Beattie. (2014, Dec.) POODLE vulnerability expands beyond SSLv3 to TLS 1.0 and 1.1. [Online]. Available: https://www.globalsign.com/en/blog/poodle-vulnerability-expands-beyond-sslv3-to-tls [Page 54.]

[100] National Institute of Standards and Technology. (2019) CVE-2019-5592. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-5592 [Page 54.]

[101] ——. (2019) CVE-2019-6593. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2019-6593 [Page 54.]

[102] I. Ristic. (2012, Oct.) CRIME: Information leakage attack against SSL/TLS. [Online]. Available: https://blog.qualys.com/product-tech/2012/09/14/crime-information-leakage-attack-against-ssltls [Page 54.]

[103] D. Fisher. (2013, Mar.) Attack exploits weakness in RC4 Cipher to decrypt user sessions. [Online]. Available: https://threatpost.com/attack-exploits-weakness-rc4-cipher-decrypt-user-sessions-031413/77628/ [Page 54.]

[104] National Institute of Standards and Technology. (2014) CVE-2014-0160. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2014-0160 [Page 54.]

[105] ——. (2016) CVE-2016-9244. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-9244 [Page 54.]

[106] ——. (2014) CVE-2014-0224. [Online]. Available: https://nvd.nist.gov/vuln/detail/cve-2014-0224 [Page 54.]

[107] ——. (2016) CVE-2016-2107. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-2107 [Page 54.]

[108] ——. (2017) CVE-2017-13099. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-13099 [Page 54.]

[109] Ciphersuite. Secure cipher suite. [Online]. Available: https://ciphersuite.info/cs/TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256/ [Page 54.]

[110] C. Young. (2019, Mar.) What is Zombie POODLE. [Online]. Available: https://www.tripwire.com/state-of-security/vert/zombie-poodle/ [Page 54.]

[111] B. Kiprin. (2019, Apr.) How to prevent SSL POODLE attack. [Online]. Available: https://crashtest-security.com/prevent-ssl-poodle-attack/ [Page 55.]

[112] I. Ristic. (2014, Oct.) SSL 3 is dead, killed by the POODLE attack. [Online]. Available: https://blog.qualys.com/product-tech/2014/10/15/ssl-3-is-dead-killed-by-the-poodle-attack [Page 55.]

[113] IBM. (2022, Feb.) Protection against padding oracle attacks. [Online]. Available: https://www.ibm.com/docs/en/datapower-gateway/10.0.1?topic=protection-against-padding-oracle-attacks [Page 55.]

[114] R. Merget, J. Somorovsky, N. Aviram, C. Young, J. Fliegenschmidt, J. Schwenk, and Y. Shavitt, "Scalable scanning and automatic classification of TLS Padding Oracle vulnerabilities," in *Proceedings of the 28th USENIX Conference on Security Symposium*, ser. SEC'19. USA: USENIX Association, 2019. ISBN 9781939133069 p. 1029–1046. [Page 56.]

[115] OWASP. Testing for reflected cross site scripting. [Online]. Available: https://owasp.org/www-project-web-security-testing-guide/v41/4-Web_Application_Security_Testing/07-Input_Validation_Testing/01-Testing_for_Reflected_Cross_Site_Scripting.html [Page 58.]

[116] ——. A4:2017-XML External Entities (XXE). [Online]. Available: https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE) [Page 58.]

[117] Open Source, "XSS-Radar." [Online]. Available: https://github.com/bugbountyforum/XSS-Radar [Page 59.]

[118] Tamper Chrome, "Tamper dev." [Online]. Available: https://tamper.dev/ [Pages 59 and 61.]

[119] OWASP. OWASP secure headers project. [Online]. Available: https://owasp.org/www-project-secure-headers/#quickly-check-security-http-headers-for-applications-exposed-on-the-internet [Page 62.]

[120] Screaming Frog Ltd, "Screaming Frog." [Online]. Available: https://www.screamingfrog.co.uk/ [Page 64.]

[121] R. Hodgman. (2016, Mar.) The attacker's dictionary. [Online]. Available: https://www.rapid7.com/blog/post/2016/03/01/the-attackers-dictionary/ [Page 64.]

[122] Probely, "Security headers." [Online]. Available: https://securityheaders.com/ [Page 65.]

[123] BuiltWith Pty Ltd, "Built With." [Online]. Available: https://builtwith.com/ [Page 67.]

[124] Wappalyzer, "Wappalyzer." [Online]. Available: https://www.wappalyzer.com/ [Page 67.]

[125] MITRE. (2007) Jquery: security vulnerabilities. [Online]. Available: https://www.cvedetails.com/vulnerability-list/vendor_id-6538/Jquery.html [Page 68.]

[126] Auth0 Inc. (2013). [Online]. Available: https://auth0.com/ [Page 69.]

[127] FIDO Alliance. (2017, Feb.) FIDO UAF architectural overview. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-overview-v1.1-id-20170202.html [Page 69.]

[128] ModHeader LLC, "ModHeader." [Online]. Available: https://modheader.com/ [Page 71.]

[129] h0nus, "QRGen." [Online]. Available: https://github.com/h0nus/QRGen [Page 76.]

[130] Google Developers – Android Developers Android Studio – User guide, "Logcat command-line tool." [Online]. Available: https://developer.android.com/studio/command-line/logcat [Page 78.]

[131] TJ Holowaychuk, "Express." [Online]. Available: https://expressjs.com/ [Page 78.]

[132] Amazon - AWS. Provisioning device identity in the manufacturing supply chain. [Online]. Available: https://docs.aws.amazon.com/whitepapers/latest/device-manufacturing-provisioning/provisioning-device-identity-in-the-manufacturing-supply-chain.html [Page 84.]

[133] Dan IFIXIT. (2020, Decemnber) Ring doorbell 3 teardown. [Online]. Available: https://www.ifixit.com/Teardown/Ring+Doorbell+3+Teardown/137191 [Page 84.]

[134] Texas Instrument. (2020, April) Msp430fr235x, msp430fr215x mixed-signal microcontrollers. [Online]. Available: https://www.ti.com/lit/ds/symlink/msp430fr2155.pdf?ts=1674512934931&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FMSP430FR2155 [Page 84.]

[135] S. Vasile, D. Oswald, and T. Chothia, "Breaking all the things: a systematic survey of firmware extraction and modification techniques for IoT devices," in *CARDIS 2018: Smart Card Research and Advanced Applications*, ser. Lecture Notes in Computer Science. Springer, Mar. 2019. doi: 10.1007/978-3-030-15462-2_12 pp. 171–185, 17th Smart Card Research and Advanced Application Conference ; Conference date: 12-11-2018 Through 14-11-2018. [Page 84.]

Appendix A

Ring mobile application logging - Failed login attempt

Listing A.1: Ring mobile application logging - Failed login attempt from 2022-11-15T21:47:42.772+0100


```
{
"@timestamp":"2022-11-15T21:47:42.772+0100",
"pid":"11937",
"tid":"17536",
"@loglevel":"ERROR",
"tag":"ClientsApi",
"message":"Received response [401]",
"user_id":"54563659",
"hardware_id":"a4f40660-5571-34e9-b27e-7cb46713246e",
"app_version_code":"27041495",
"app_version_name":"3.40.0",
"app_brand":"Ring",
"os":"android",
"connectivity_type":"wifi",
"cellular_technology":null,
"carrier_name":null,
"request_id":"59",
"url":"https:\\/\\/api.ring.com\\/clients_api\\/device",
"method":"PATCH",
"req_headers":
"'User-Agent: android:com.ringapp:3.40.0(27041495)',
'Hardware_ID: a4f40660-5571-34e9-b27e-7cb46713246e',
'X-API-LANG: en',
```

```
'Content-Type: application\\/json',
'app_brand: ring',
'accept: application.vnd.api.v11+json'",
"req_body":"{
\\"device\\":{
\\"metadata\\":{
\\"api_version\\":\\"11\\",
\\"app_brand\\":\\"ring\\",
\\"app_build\\":\\"27041495\\",
\\"app_instalation_date\\":\\"2022-05-26T14:37:14.083Z\\",
\\"app_version\\":\\"3.40.0\\",
\\"architecture\\":\\"arm64-v8a\\",
\\"device_model\\":\\"SM-G935F\\",
\\"device_name\\":\\"Galaxy S7 edge\\",
\\"h264\\":\\"baseline:52 main:52 high:52 \\",
\\"is_tablet\\":false,
\\"language\\":\\"en_US\\",
\\"linphone_initialized\\":false,
\\"manufacturer\\":\\"samsung\\",
\\"os_version\\":\\"8.0.0 (26)\\",
\\"pn_service\\":\\"fcm\\",
\\"resolution\\":\\"1080x1920\\",
\\"supports_echo_canceler\\":true},
\\"os\\":\\"android\\",
\\"push_notification_token\\":\\"\\"}}",
"code":"401",
"elapsed":"181 ms",
"resp_headers":"'date: Tue, 15 Nov 2022 20:47:42 GMT',
'content-type: application\\/json',
'content-length: 42',
'server-version:
   ↪ aef92e4a237e35f15f7d846187d0b12fd06b4796',
'server-region: us-east-1',
'x-frame-options: SAMEORIGIN',
'x-content-type-options: nosniff',
'x-xss-protection: 1; mode=block',
'x-request-id: f5a7c8d13d142c0f55fd6379b0c44d20',
'server: api.ring.com api.prod.ring.net'",
"resp_body":"{\\"error\\":\\"Unauthorized\\",\\"status_code\\":401}"}
```

Listing A.2: log from 2022-11-15T21:47:42.780+0100


```
{
"@timestamp":"2022-11-15T21:47:42.780+0100",
"pid":"11937",
"tid":"11937",
"@loglevel":"ERROR",
"tag":"PushTokenMonitor",
"message":"Error while trying to send push token. -
    ↪ HttpException - HTTP 401 ",
"user_id":"54563659",
"hardware_id":"a4f40660-5571-34e9-b27e-7cb46713246e",
"app_version_code":"27041495",
"app_version_name":"3.40.0",
"app_brand":"Ring",
"os":"android",
"connectivity_type":"wifi",
"cellular_technology":null,
"carrier_name":null,
"backoff_ms":"20000"
}
```

Listing A.3: log from 2022-11-15T21:47:48.810+0100


```
{"@timestamp":"2022-11-15T21:47:48.810+0100",
"pid":"11937",
"tid":"17707",
"@loglevel":"ERROR",
"tag":"RingAuth",
"message":"Authenticate FAILED with response.",
"user_id":"54563659",
"hardware_id":"a4f40660-5571-34e9-b27e-7cb46713246e",
"app_version_code":"27041495",
"app_version_name":"3.40.0",
"app_brand":"Ring",
"os":"android",
"connectivity_type":"wifi",
"cellular_technology":null,
"carrier_name":null,
"auth_request_id":"2",
"username":"tester5656789@gmail.com",
"password":"...@",
"tfa":"...",
"error_response":"Response{protocol=h2,
code=401,
message=,
    ↪ url=https:\\/\\/oauth.ring.com\\/oauth\\/token}"}
```

Listing A.4: log from 2022-11-15T21:47:48.824+0100


```
{
"@timestamp":"2022-11-15T21:47:48.824+0100",
"pid":"11937",
"tid":"11937",
"@loglevel":"ANALYTICS",
"tag":"AnalyticsServices",
"message":"event",
"user_id":"54563659",
"hardware_id":"a4f40660-5571-34e9-b27e-7cb46713246e",
"app_version_code":"27041495",
"app_version_name":"3.40.0",
"app_brand":"Ring",
"os":"android",
"connectivity_type":"wifi",
"cellular_technology":null,
"carrier_name":null,
"id_tag":{
"event":"Log In Attempted",
"props":{
"Was Successful":"False",
"2fa Prompted":"False",
"Response Code":"401"}}
}
```

Appendix B

Custom word-list for QR-code generator

Listing B.1: Custom word-list for QR-code generator

```
https://ring.com/s?m=54D11913X399&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=96AB40952FDA&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=1&d=rvd3&n=BHRG62010CN003488&b
    ↪ =t&v=2
https://ring.com/s?m=null&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m&d=rvd3&n=BHRG62010CN003488&b=t
    ↪ &v=2
https://ring.com/s?m=%27%3B&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=%27%3BTest&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=%3Bsystem(%27id%27)&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=123456RANDOM&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?d=rvd3&n=BHRG62010CN003488&b=t&v
    ↪ =2
https://ring.com/s?m=54E01913B395&d=null&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd5&n=
```

```
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd1&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=54E01913B395&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=54E01913B395&d=%27%3BTest&n=
    ↪ BHRG62010CN003488&b=t&v=2
https://ring.com/s?m=54E01913B395&d=%27%3Cscript%3
    ↪ Ealert%281%29%2F%2F&n=BHRG62010CN003488&b=t&v
    ↪ =2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BTXG62010CN003499&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=null&b=t
    ↪ &v=2
https://ring.com/s?m=54E01913B395&d=rvd3&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=%27%3B&b
    ↪ =t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=<iframe
    ↪ src=javascript:alert(1) />&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=%27<
    ↪ iframe src=javascript:alert(1) />&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=%27%3B<
    ↪ iframe src=javascript:alert(1) />&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=%27-
    ↪ alert(1)-%27&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=%27%3
    ↪ Cscript%3Ealert%281%29%2F%2F&b=t&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=x&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=tererrer&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=null&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=%27%3Cscript%3Ealert
    ↪ %281%29%2F%2F&&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&v=2
https://ring.com/s?m=54E01913B395&d=rvd3&n=
```

```
    ↪ BHRG62010CN003488&b=t&v=0
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=100
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=null
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=%27%3Cscript%3Ealert
    ↪ %281%29%2F%2F&
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t&v=%27%3Cerror
https://ring.com/s?m=54E01913B395&d=rvd3&n=
    ↪ BHRG62010CN003488&b=t
```

Appendix C

Ring Android application functions gathered from APK files

Listing C.1: currentSsidContainsAndConnected

```
1  public boolean currentSsidContainsAndConnected(String str
   ↪ , String str2, String str3) {
2      Intrinsics.checkNotNullParameter(str, "ssidPrefix");
3      Intrinsics.checkNotNullParameter(str2, "ssidSuffix");
4      return getConnectivityApi().
   ↪ currentSsidContainsAndConnected(getWifiConfig(str, str2
   ↪ , str3));
5  }
```

Listing C.2: ConnectivityApi.WifiConfig getWifiConfig

```
1  private final ConnectivityApi.WifiConfig getWifiConfig(
   ↪ String str, String str2, String str3) {
2      ConnectivityApi.WifiConfig build = ConnectivityApi.
   ↪ WifiConfig.newBuilder().prefix(str).posfix(str2).build(
   ↪ str3);
3      Intrinsics.checkNotNullExpressionValue(build, "
   ↪ WifiConfig.newBuilder()\…n        .build(macAddress)");
4      return build;
5  }
```

Listing C.3: class WifiConfig

```
    public static class WifiConfig implements WifiFilter,
↪ Serializable {
        public String posfix;
        public String prefix;
        public String ssid;

        public static final class Builder {
            public String posfix;
            public String prefix;
            public String ssid;

            public WifiConfig build() {
                return new WifiConfig(this);
            }

            public Builder posfix(String str) {
                this.posfix = str;
                return this;
            }

            public Builder prefix(String str) {
                this.prefix = str;
                return this;
            }

            public Builder ssid(String str) {
                this.ssid = str;
                return this;
            }

            public Builder() {
            }

            public WifiConfig build(String str) {
                if (str == null) {
                    return new WifiConfig(this);
                }
                if (!TextUtils.isEmpty(this.prefix) && !
↪ TextUtils.isEmpty(str) && !TextUtils.isEmpty(this.
↪ posfix)) {
                    if (str.matches("^(..:?)+$")) {
                        str = str.replaceAll(
↪ ClickStreamHelper.COLON_DELIMITER, "");
                    }
                    this.ssid = this.prefix + str.substring(
↪ str.length() - this.posfix.length(), str.length());
```

```
42                      }
43                      return new WifiConfig(this);
44                  }
45              }
```

Appendix D

Simple Express JS server

```
const express = require('express')
const fs = require('fs');
const https = require('https');
const xml = require(”xml”)

const address = ”192.168.240.1”
const port = 443
const ringPath = '/ring/system/config/network'
const networkResponse = `
<network>
  <script />
  <mode>limited-ap</mode>
  <ap_mode>user-ap</ap_mode>
  <object_id/>
  <client>
    <wireless>
      <channel>11</channel>
      <ssid>TestWifi</ssid>
      <security>wpa-personal</security>
      <webauth/>
      <password>superSecurePass</password>
      <eap_type/>
      <eap_username/>
      <eap_password/>
    </wireless>
    <ip>
      <ip_type>static</ip_type>
      <ip_addr>192.168.137.201</ip_addr>
      <subnetmask>255.255.255.0</subnetmask>
      <gateway>192.168.137.1</gateway>
      <dns_addr>192.168.137.1</dns_addr>
```

```
    </ip>
    <secret_key/>
  </client>
</network>
`

const app = express()

app.use((_req, res, next) => {
    res.header('Access-Control-Allow-Origin', '*');
    res.header('Access-Control-Allow-Methods', 'GET, POST,
    ↪ PATCH, DELETE');
    res.header('Access-Control-Allow-Headers', 'Origin, X-
    ↪ Requested-With, Content-Type, Accept, Authorization');
    next();
  });

app.get('/', (_req, res) => {
    console.log('Index request');
    res.send('Index page');
})

app.get(ringPath, (req, res) => {
  console.log(`Request to ${ringPath}`)
  console.log("Request Body: ", req.body)
  console.log("Request Headers: ", req.headers)

  res.set('Content-Type', 'text/xml');
  res.status(200).send(xml(networkResponse))
})


https.createServer({
  key: fs.readFileSync('server.key'),
  cert: fs.readFileSync('server.cert')
}, app)
  .listen(port, address, () => console.log(`HTTPS server is
    ↪ listening on https://${address}:${port}`));
```

TRITA-EECS-EX- 2023:51