



DEGREE PROJECT IN TECHNOLOGY,  
FIRST CYCLE, 15 CREDITS  
*STOCKHOLM, SWEDEN 2020*

# **Ethical Hacking of a Robot Vacuum Cleaner**

**ERIC BRÖNDUM**

**CHRISTOFFER TORGILSMAN**

# Ethical Hacking of a Robot Vacuum Cleaner

## Etisk Hackning av en Robotdammsugare

Eric Bröndum, *KTH*, Christoffer Torgilsman, *KTH*

Examiner Assoc. Prof. Robert Lagerström, *KTH*, Supervisor Prof. Pontus Johnson, *KTH*

**Abstract** - This study revolves around the safety of IoT devices, more specifically how safe the robot vacuum cleaner Ironpie m6 is. The method is based on threat modeling the device, using the DREAD and STRIDE models. The threats with the highest estimated severity were then penetration tested to see which security measures are implemented to protect against them. Using client side manipulation one vulnerability was found in Trifo's mobile application "Trifo home" which could be used to harm customers property.

**Sammanfattning** - Den här studien kretsar kring IoT enheters säkerhet, mer specifikt hur säker robotdammsugaren Ironpie m6 är. Metoden är baserad på att hotmodellera enheten med hjälp av DREAD och STRIDE modellerna. Dem allvarligaste hoten blev penetrationstestade för att se vilka säkerhetsåtgärder som har blivit implementerade för att skydda produkten från dem. En sårbarhet upptäcktes i Trifos mobilapplikation "Trifo Home" som kunde exploiteras via manipulation av klient sidan. Denna sårbarhet kunde användas för att skada kunders ägodelar.

**Keywords** - IoT, Ethical Hacking, Penetration testing, Threat Model, Security, DREAD, STRIDE, Encryption, MQTT, Ironpie m6

## I. Introduction

Hacking has grown to be a real problem in recent times with the rapid increase of IoT devices. To protect consumers from exploitation, ethical hackers called white hats are used to find the vulnerabilities before they get exploited by malicious hackers called black hats. Maintaining the security of IoT devices is vital since many devices have access to private information and others have sensors and cameras that can be used to violate peoples privacy. This problem will only continue to grow as society becomes increasingly dependent on different IoT devices. This increases the demand and necessity of ethical hacking so that consumer products can remain safe from exploitation [1].

This study revolves around ethical hacking and cy-

ber security. The objective of this study is to hack a robot vacuum cleaner (Trifo Ironpie m6). This is done with the intention of evaluating the device's security measures to find out how secure the device is. The results from this study can be used both by the company when making risk assessments and regular maintenance, but also by other ethical hackers to provide them with methods and hacking techniques that they can use when evaluating other similar devices.

When analyzing the devices security measures two delimitations are used. The first delimitation of this study is to ignore hardware that can only be accessed by disassembling the robot. This delimitation is established due to three main reasons; the first reason is that in a real world scenario the hacker will most likely not have physical access to the robot, therefore physical modifications is classified as a non-threat. The second reason is to reduce the risk of damaging any internal components and finally the third reason is due to time constraints.

The second delimitation of this study is to not hack any servers. This delimitation is established so as not to break any laws that concern the act of hacking other peoples property.

This report is structured in the following way:

In section II (Background) some network theory and the general concepts of IoT and ethical hacking are described. This section also includes references to previous work done in the field of ethical hacking, for example hacks performed on robot vacuum cleaners and other IoT devices.

Section III (Ironpie m6) contains information about the targeted device (Ironpie m6) and the accompanying software.

Section IV (Method) describes both hacking methods and methods used for threat modeling and the general methods used when penetration testing.

Section V (Threat model) includes the asset description, architecture diagram and also various threats categorised by the STRIDE and DREAD models.

In section VI (Penetration testing) all the penetration tests are listed with their associated methods used,

results and discussion with a quick explanation for any software and tool used within the test.

Section VII (Results) includes a summarization of the main threats that were evaluated in a threat traceability matrix.

Section VIII (Discussion) discusses the validity, reliability and generalizability of the found results from the penetration testing.

Section IX (Sustainability and ethics) describes how the penetration testing remained "ethical" and legal.

Section X (Conclusion) includes a summarized security evaluation of the Ironpie m6 based on the findings of this study.

Section XI (Appendix 1 - Threat Traceability Matrix for mobile applications) contains a threat traceability matrix for general threats against mobile applications, more specifically applications that can control or interact with robot vacuum cleaners.

Section XII (Appendix 2 - Threat Traceability Matrix for robot vacuum cleaners) contains a threat traceability matrix for general threats against IoT devices more specifically robot vacuum cleaners.

Section XIII (Appendix 3 - Proof of Concept Server code) contains proof of concept code for a HTTP response server that sends a file on received connection.

Section XIV (Appendix 4 - The Results Threat Traceability Matrix) contains the results summarized inside of a threat traceability matrix.

## II. Background

### A. IoT

IoT which stands for Internet of Things is a communication paradigm that has recently started to gain traction and the number of IoT devices have rapidly grown for several years. IoT aims to connect all devices to the internet and therefore simplify the process required to harvest data generated by various devices such as sensors. Another reason for connecting devices to the internet is that it allows for some devices to be remotely controlled. "Smart" devices are constantly expanding their reach, new devices are constantly being developed for more and more markets resulting in smart fridges, smart watches and so on. This comes at a cost however since some devices require personal and in some cases private data to function as intended. This means the data is threatened and could be obtained by an hacker if the security of the device is lacking [2]. To prevent malicious hackers from exploiting weaknesses the IoT developer may hire ethical hackers to secure their devices.

### B. Ethical Hacking

Ethical hacking also known as white hat hacking involves the use of penetration testing on authorized devices to locate security flaws with the intention of getting them patched. White hat hacking is a completely legal practice and is often rewarded quite handsomely, hackers are mainly used to find vulnerabilities that the design team may have overlooked when designing the product. By finding weaknesses in the targeted IoT device and then getting the problems fixed quickly limits Black and Grey hat hackers from exploiting the found weaknesses in secret [1].

### C. MQTT

MQ-Telemetry Transport is a machine to machine (M2M) protocol that mainly focuses on IoT devices due to its low requirement of bandwidth. MQTT uses a broker and a publish/subscribe design, to communicate data over different devices. The broker stores all the received published messages in topics so that devices that subscribe to these topics can retrieve the message [3].

### D. HTTPS and SSL/TLS

Hypertext Transfer Protocol Secure (HTTPS) is used to communicate over computer networks in a secure way. The Security Sockets Layer (SSL) and Transport Layer Security (TLS) provides security related to authentication and data transfer. They use both the handshake and record protocol to transport all data safely. When a client sends a request to a website SSL and TLS will provide a certificate to the client. When the client receives the certificate it is further evaluated by the web browsers Certificate authority(CA) which validates that the website is correct and secure [4].

### E. ARP

Address Resolution Protocol (ARP) is a communication protocol, that binds the IP addresses of the network layer to the MAC addresses of the link layer. By using ARP tables, the protocol can translate the IP addresses used by applications to the MAC addresses used by individual nodes [5].

### F. SSH

Secure Shell (SSH) is used for secure remote login and file transfers on port 22. The SSH protocol works as a client-server model. The SSH client initiates the connection and uses public key cryptography to verify the SSH server. When the server client connection is set up, the SSH protocol uses encryptions and strong

hashes to ensure that the data sent between the client and server is secure [6].

### G. Previous work

In [7], the aim was to find vulnerabilities in consumer IoT devices with a large scaled approach. Two main tools were used; Shodan to get quantities of IoT devices to test and Nestle to determine potential vulnerabilities. The results showed that IoT devices are vulnerable and easy to exploit when compromising user data. Nearly 40% of the IoT devices showcased a 'high' risk of having vulnerabilities. There has been multiple case studies on consumer IoT devices like web cameras, smartTVs and healthcare systems. Showcasing common vulnerabilities in these devices but also highlighting the necessity for better security [8] [9] [10]. Another IoT device that has been studied extensively are robot vacuum cleaners [11] [12] [13] [14], one in particular was evaluated by Theodor Olsson and Albin Larsson Forsberg [15]. They studied the "Jisiwei i3 Robot Vacuum" with the intention of finding vulnerabilities and evaluating the security of the device. They accomplished this by using the threat model analysis methods STRIDE and DREAD and from the gathered results performed penetration testing. Two vulnerabilities were discovered, One vulnerability was regarding the protocol that the device communicated information over. The other vulnerability being predictable QR codes used to link devices to users.

Another vacuum cleaner that has recently been studied for vulnerabilities, is the Ironpie m6 which is this studies researched vacuum cleaner. In [16], Checkmarx Security Research Team found six vulnerabilities that could infringe upon a customers privacy. The most severe of the vulnerabilities which were rated eight or higher on the CVSS scale are [17];

1. An unencrypted HTTP request was sent out when querying for a software update. This enables the hackers to tamper with the request and to send a malicious version instead.
2. The ability to impersonate someone else's client ID and therefore gain remote access to the MQTT servers.
3. The ability to impersonate the MQTT server, which would give the hacker full control of the vacuum cleaner.

### III. Ironpie m6

During this study the device evaluated is Trifo's robot vacuum cleaner Ironpie m6. The Ironpie m6 is a mid

range vacuum cleaner with features such as good navigation made possible by the use of SLAM algorithms and an integrated camera. SLAM algorithms are algorithms that allow robots to simultaneously realize their position in the environment while at the same time mapping the new environment. The robot also includes multiple sensors with the intention of providing more safety for the robot, the sensors can identify stairs and other dangerous environments. The vacuum cleaner can only connect to 2.4 GHz local networks with the inbuilt WiFi receiver. To manually maneuver and control the vacuum cleaner's functionalities Trifo's mobile application "Trifo Home" is used. Functionalities that can be controlled through the mobile application are for example the camera. The camera can be controlled to record videos or be used as a live video feed, working as a surveillance camera. Other functionalities like telling the robot to charge or clean are also available in the application. The application has support for both iOS and Android.

## IV. Methodology

As software security has gained more attention many different models have been defined and created to analyze the safety of the product. This is called Threat modeling which is the technique used when modeling, identifying and reducing risks within the product. Threat modeling can be accomplished manually or by using automatized tools. Some manual approaches would be to use STRIDE, DREAD or PASTA [18][19]. While some automatized approaches could be to use MulVal, the TVA tool or NetSecuritas [20]. In this project the manual methods STRIDE and DREAD will be used due to their simplicity and efficiency of identifying threats in smaller IoT ecosystems.

The STRIDE model suggested by Microsoft is used to identify threats in a given product by highlighting the six different threats categories that a product can encounter. These threats are categorised as Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS) and Elevation of Privilege [18]. These threat categories are explained below [21].

- Spoofing is the act of stealing or identifying as an other person or computer, to get illegitimate access or advantages.
- Tampering information is when legitimate information is modified or edited.
- Repudiation is to discard or deny certain actions in a given system.
- Information Disclosure regards data breaches

which is when the hacker gets unauthorized access to confidential information.

- DoS is to attack or disrupt different services to interfere with the legitimate users.
- Elevation of Privilege is to reach higher privilege access to a system, from a user with restrictive authority.

The threat assessment model DREAD is a tool that can be used when prioritizing which threats to be handle first. The DREAD model is made up of five different factors that are assessed individually and then summed up producing a number that it uses for risk assessment. These factors can then be further split up into three factors that describe the likelihood of the threat occurring and two factors that describe the severity of the threat. The first factor is damage potential, this factor measures the impact and the damage that an exploitation could cause. It is therefore one of the factors used for assessing the severity of the threat. The second factor that is assessed is reproducibility which is one of the likelihood factors. It is evaluated in regards to how easy it would be to produce attacks. The third factor is exploitability this is also an likelihood factor and measures how easy the threat would be to exploit. The fourth factor is affected users this is a severity factor that measure how many users that would be affected by the threat. The final factor discoverability is a likelihood factor that measure the possibility of the threat being found [18] [22] [23].

The methodology followed in this study is divided into two parts: Threat modeling and penetration testing. The threat modeling methodology is based on chapter 2 of [24] the "IoT penetration testing cookbook". The threat modeling method described in this chapter uses both the STRIDE and DREAD models and contains the following steps which will be followed in this study:

1. Identify all the assets in the device ecosystem
2. Visualize the architecture of the device
3. Decompose the IoT device
4. Identify threats using the STRIDE model
5. Document the found threats
6. Rate the threats using the DREAD model

After a threat model has been created and the different threats are listed and rated the next step is to penetration test the most severe of them. There exists multiple different approaches to penetration testing but the most common ones are: Black box testing and white box testing [25].

When applying Black box testing, the hacker will access the network infrastructure without being aware of any internal technologies created by the organization. Since the information about the device is limited, it is generally a more time consuming and expensive service. Vulnerabilities can be identified and exploited through multiple testing phases and the use of real world hacking techniques [25].

White box testing is done with the full knowledge of all the internal and underlying technologies that is part of the system. Thanks to this the knowledge required to utilize this testing method is much lower than other methods such as black box testing. The testing can be done with minimal effort and are much more accurate since the time spent testing can be fully utilized testing potential vulnerabilities instead of investigating the protocols used for example [25].

During the penetration testing phase of this study the black box testing approach will be used since this study is not supported by Trifo. The threats with the highest priority will be tested individually by utilizing common attack techniques. Some of these attack techniques are:

#### *Brute Force Attack*

To brute force something such as a password or pin code is to try every single input combination that could be an possible login. Brute force attacks are inefficient since the difficulty of actually finding the correct input grows exponentially in regards to the input length [26]. Therefore the time complexity for this sort of algorithm is  $\mathcal{O}(\gamma^\eta)$ , where  $\gamma$  is the character set size, and  $\eta$  is the length of the input.

#### *Dictionary Attack*

A dictionary attack is similar to a brute force attack, the difference being that instead of trying every digit combination, a list of common usernames and passwords are tried with each other until one combination matches. There exists multiple popular password lists to use such as "Rockyou" containing more than 14 million passwords currently and "John the Ripper".

#### *Man-in-the-middle*

Man-In-The-Middle (MITM) is a attack where a third party listens and takes control of the communication between others while remaining hidden. The attacker has the ability to intercept the target by using Distributed denial of service (DDOS). But also to modify, change or replace the targets communication traffic.

Since the person has such control when conducting the MITM attack, it is possible to receive user IDs and passwords. A passive MITM attack is when the attacker's presence remains hidden, with the intent of capturing the data and then sending it to the correct user. An active attack is where the content received gets manipulated before being sent out to the original destination. MITM is mainly used on a wireless connection, it is much easier to hook up to different hot spots in comparison to wired connections [27].

#### *Replay attack*

A replay attack is a kind of MITM attack where the attacker sniffs the local network traffic with the intention of storing or manipulating packets sent or received by the target. These packets are then later used in the replay attack and sent back to the target. The reason why replay attacks are used is because they can bypass encryption such as SSL or TLS. The attack bypasses the encryption due to never creating any new packets and instead reuses packets that the target themselves created or requested [28].

#### *Port scanning*

By port scanning a certain network, open ports and devices connected to the network can be identified. Each port provides a certain service, more common ones are port 80 (HTTP), port 22 (SSH) and port 23 (Telnet). By identifying the open ports, the attacker can attack the port with the intentions of gaining access to the device.

#### *Phishing a mobile application*

Phishing is a form of identity theft in that it tries to copy the look of another web page or application. A phishing attack tries to exploit the human factor of not recognizing small differences in familiar environments. This can cause users to accidentally give account information to the attacker [29].

## V. Threat model

The first step in creating a threat model is to divide and identify the different assets that the evaluated system contains or interacts with. In this study's case the Ironpie m6 is evaluated. The Ironpie m6's assets can be seen in Table 1.

The next step is to visualize the architecture of the device by creating use cases and an architectural diagram. The use cases are created in order to get a better grasp of how the different assets and functionalities work together. A few use cases of the Ironpie m6 are mentioned below.

### **Use case 1: To clean, start a video or charge the robot**

1. Install Trifo's mobile application "Trifo home".
2. Register an account with the use of email and desired password.
3. Sign in to the account and press add device.
4. Select Ironpie m6, and your local network.
5. Use the provided QR code from the application to provide the Ironpie m6 with internet access and to link the account with the Ironpie m6.
6. Select the robot in the main menu.
7. Press "Clean", "Recharge" or "Start Video" button depending on what the user desires.

### **Use case 2: Install firmware updates on Ironpie m6**

- 1-6. Same as use case 1.
7. Go to "Device Settings".
8. Press "Software update".

### **Use case 3: Install and upgrade the application version**

- 1-5. Same as use case 1.
6. Go to "Settings".
7. Press "About".
8. Press on "Version"
9. Query and perform the application update.

### **Use case 4: Change the password for the mobile application**

- 1-5. Same as use case 1.
6. Go to "Settings".
7. Press "About".
8. Press on "Change Password" and enter in the new password.

Table 1: Description of the vacuum cleaners assets.

Assets	Description
Ironpie m6	The ironpie m6 manufactured by Trifo comes with an inbuilt camera, two different sensors: a cliff sensor and a bumper sensor. Navigation is done through the usage of SLAM algorithms that are in turn supported by the usage of the inbuilt camera. Manual control over the robot and access to the video feed can be obtained through connecting the robot to an Trifo account in the mobile application. The robot has WiFi capabilities for this reason.
Application	There exists a mobile application for both Android and IOS devices. The app gives the user the ability to navigate the robot but also get a live feed of the camera.
Firmware	The firmware is used to control speakers, video feeds, lights and movement in the robot.

After creating the use cases, the architectural diagram is created to showcase how the data in the system is transferred both internally and externally. The diagram highlights what network protocols that the evaluated system uses to transport data i.e unencrypted TCP is used to transport version control requests and responses. HTTPS is used to transport sensitive information such as account credentials when logging in. All MQTT traffic is encrypted using SSL and transported using TCP. The robot also sends out UDP broadcasts. According to the Checkmarx research team the MQTT server acts as a bridge for all network traffic passing between the application, back-end server and robot [16]. Ironpie m6’s architectural diagram is showcased in Figure 1.

With the architectural diagram in mind, the next step in the threat model methodology is to decompose the IoT device. The purpose of this step is to highlight entry points that are vulnerable for the device or application. In this case the Ironpie m6 contains a multitude of entry points where some are easier to exploit than others. The most apparent one being the mobile application which is used to control most of the robots functionalities with the only requirement being account credentials. The communication between the mobile application and server can be monitored but is encrypted with SSL. For the robot to communicate with the mobile application the robot has to be connected to a 2.4 Ghz WiFi network. Another entry point is the Ironpie m6 itself, since it might have ports

open that can be exploited. The last entry point is the firmware since it controls all the robots actions. This also includes when the mobile application wants to assume control, where the commands sent from the application are enforced by the firmware.

The next step in the threat model creation process is to discover and identify threats within the system. A general threat overview of the different attack vectors in IoT devices and mobile applications were analyzed. Two of OWASP’s top 10 lists were used, the top 10 IoT threats and the top 10 mobile application threats. These lists are used to make sure that no essential attack vectors are left out [30] [31]. The listed threats were then filtered for ones that can affect robot vacuum cleaners and their eco-systems. These threats were summarized and placed into two threat traceability matrices which can be found in Appendix 1 and 2.

After achieving a general understanding of common found threats in robot vacuum cleaners, the STRIDE model was used to highlight more specific threats against the Ironpie m6. The STRIDE model can be seen in Table 2.

Table 2: Threats described using the STRIDE model.

Threat Category	Description
Spoofing	Confidential information can be gathered such as login details, the connection can be disrupted using specific forwarding.
Tampering	Reverse engineering the android application to modify the shell code or binary files. Analyzing and modifying the packets being sent between the phone and router to have malicious content.
Repudiation	The integrity of the data is not secured by means such as up to date data hashes i.e. SHA2, SHA3 and BLAKE2. This will allow hackers to tamper with data unnoticed.
Information Disclosure	Confidential information is sent unencrypted allowing for malicious exploits to be performed. For example when querying for software updates or sending account credentials when logging in.
Denial of Service	
Elevation of Privilege	Horizontal privilege escalation, using brute force methods to retrieve account information via open SSH ports. This information is then misused to exploit the robots functionalities.

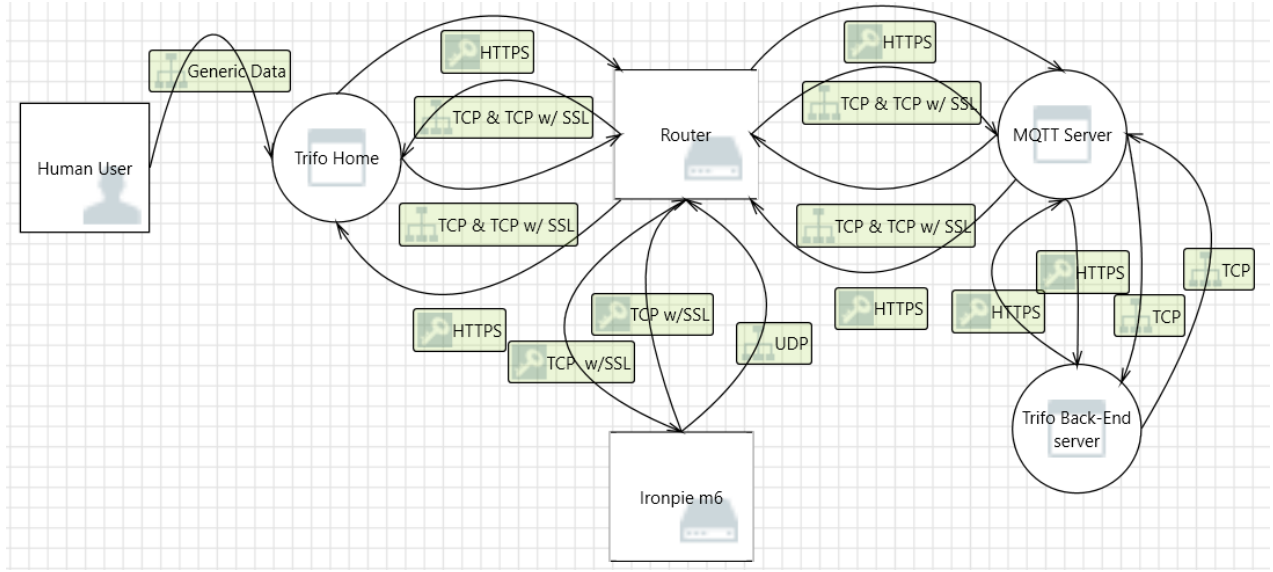


Figure 1: The architectural diagram of Trifo's Ironpie m6.

Table 4: Threat #2 Documentation

After creating the STRIDE model the next step is to document the threats. In this documentation the threats: description, threat target, attack technique(s) and countermeasure(s) will be listed. This threat table can be seen in Table 3-7.

Table 3: Threat #1 Documentation

Threat description	The attacker gains account credentials for the mobile application and therefore controls the robot or gets the client ID from the robots MQTT traffic.
Threat target	Mobile application and Robot.
Attack technique	The attacker can perform a MITM attack using ARPspooft and tools like mitmproxy or Burp Suite to sniff the packets sent from or received by the mobile application or the robot.
Countermeasures	By using encryption tools such as SSL or TLS to turn regular HTTP traffic into HTTPS, the network traffic becomes safer.

Threat description	The attacker gains root access to the robot via open ports.
Threat target	The robot's open ports.
Attack technique	By using nmap's port scanning functionality the robots open ports can be identified. These open ports can then be brute forced with the help of programs like Hydra to gain root access.
Countermeasures	Uses secure ports like SSH.

Table 5: Threat #3 Documentation

Threat description	The attacker downloads the APK files. By decoding the APK, the source code can be used to find weaknesses in the mobile application but can also be modified.
Threat target	Mobile application for Android.
Attack technique	By using APK decoders the source code can be obtained and then be read or modified in a text editor. Static code analysis tools can be run on the source code to identify bugs or vulnerabilities.
Countermeasures	The code is obfuscated, which hinders the attacker from understanding the source code.



Table 6: Threat #4 Documentation

Threat description	The attacker sends their own files when a client queries for an software update of the mobile application.
Threat target	Mobile application.
Attack technique	The HTTP request can be intercepted using Burp Suite and then tampered with so that the request gets sent to the attacker instead of Trifos own servers.
Countermeasures	The software update query should redirect the user to Google play or App Store. If not possible it should use HTTPS.

Table 7: Threat #5 Documentation

Threat description	The attacker impersonates the MQTT server and through this gains full control of the robot
Threat target	The robots wireless MQTT connection.
Attack technique	The attacker spoofs the robot into thinking that it is the MQTT server and therefore enables the attacker to send direct commands to the robot.
Countermeasures	The network traffic should be encrypted.

The last step of the threat model is to prioritize and evaluate the discovered threats, the DREAD model was used in this study as can be seen in Table 8. Each DREAD category was scaled from one to three, where a one corresponds to "low risk", two corresponds to "medium risk" and a three corresponds to "high risk". The threats that gets an evaluation of twelve or higher are considered "high risk" vulnerabilities, threats that get an evaluation between 8-11 are "medium risk" and finally an evaluation of 5-7 being "low risk" [24].

Table 8: Threats evaluated according to the DREAD model.

Threat Description	D	R	E	A	D	Total
The attacker listens and receives packets containing confidential information i.e. account credentials by utilizing ARPspooft to redirect the network traffic between the gateway host and the target.	3	3	3	2	3	14
The attacker sees that a user wants to download an software update while packet sniffing, and sends their own malicious version [32].	3	2	2	3	3	13
The attacker gains root access to the robot by using brute force programs such as Hydra on open ports that can be found via port scanning.	3	3	3	1	3	13
The attacker downloads the android applications APK file and decodes it to get the source code and resource files. These files can then be analyzed and modified for malicious purposes.	3	3	1	2	3	12
The attacker impersonates the MQTT server and through this gains full control of the robot[32].	3	1	1	3	3	11

#### *Confidence of success*

When estimating the probability of success for the different threats three factors were taken into consideration; complexity, requirements and previous work based on other studies. A threats probability of success could then be rated as low, medium or high.

The first threat, being to spoof and sniff account credentials and clientID scored a 14 on the DREAD model

with the estimated probability of success being high. The DREAD scores are based on the definitions stated in [24]. Where a high on both exploitability and reproducibility meant that the attack could be performed by a novice at anytime. This is one of the reasons that the threats success rate is estimated to be high. The other reason being that MITM attacks has previously been shown to work on the mobile applications and similar robot devices as seen in [15] [16].

The second threat, to send a malicious application during a software update query scored 13 on the DREAD model with the estimated probability of success being high. In [16] a recent exploitation of this threat was successfully performed on the Ironpie m6. This is the primary reason why the estimated probability of success is rated as high. Because in the DREAD model this threat was rated as medium in both exploitability and reproducibility. Which means that only a skilled attacker could reproduce the attack repeatedly and that it could only be performed on specific conditions.

The third threat is to brute force an open SSH port using a dictionary attack. This threat was given a score of 13 in the DREAD model, with the estimated probability of success being low. The estimation is rated as low because dictionary attacks can be rather inconsistent. The chance of success is dependant on the passwords complexity and the amount of passwords contained within the dictionary. Although the threat was rated as high in both exploitability and reproducibility it does not change the fact that the method itself is inconsistent.

The fourth threat is to analyze and modify the APK files for the mobile application. The DREAD score of this threat is 12 and the estimated probability of success is low. The reason behind this estimation is the high complexity of the attack, only a skilled attacker with in depth knowledge would be able to perform it. This can be seen in the low evaluated score for exploitability given to this threat in the DREAD model.

The final threat is to impersonate the server and gain full control of the robot. The DREAD score was 11 with the estimated probability of success being low. This estimation is mainly due to the severe complexity of executing the attack. The test requires specific conditions to line up outside the attackers control and for the attacker to be skilled with in depth knowledge of the attack for it to succeed [16]. This can also be seen in the low DREAD model rating of exploitability and reproducibility.

## VI. Penetration testing

### A. Test 1 - MITM on the mobile application with no client side manipulation

#### *Introduction*

The attack vector to be explored is spoofing. Since the robot vacuum cleaner uses a mobile application to communicate, a man in the middle attack can be performed. The aim of this test is to discover if the network traffic is encrypted. To perform this test Kali Linux was used inside an VM together with Ettercap and ARPspoofer.

#### *Kali Linux*

Kali Linux (kali) is its own Linux distribution based on the Debian Linux distribution. Kali is specialized towards performing penetration testing and includes multiple penetration testing tools i.e to find information, potential vulnerabilities, sniffing and spoofing[25].

#### *ARPspoofer*

ARPspoofer is a tool that is often used when attacking devices on an local area network (LAN). This tool can be used to modify Address Resolution Protocol (ARP) tables and therefore redirect packets that travel over the network so that they pass through the attacker. This can be done because ARP does not contain any authentication of where the reply packet came from. It is therefore possible to redirect traffic to the attacker and then generate forged reply packets. These replies can then be sent to the original destination with no one in the network detecting the attack [33].

#### *Ettercap*

Ettercap is a program that is used to create MITM attacks in a simple way. The program can detect nearby devices and hosts, but also sniff and filter the content sent over live connections[34].

#### *Method*

This test is performed to check if the packets sent by the application are unencrypted. To test this the packets sent by the robots application "Trifo Home" are redirected to the attacker by utilizing ARPspoofer and then displayed using Ettercap.

#### *Results*

This MITM test verified that the data was not transported via any non encrypted means such as HTTP. All of the packets received from Ettercap were encrypted,

using a SSL certificate called "GoDaddy". No passwords or usernames could be viewed in cleartext.

### Discussion

This MITM penetration test was performed to see if the packets sent were unencrypted and to see if any compromising information such as account credentials were shown. This test determines how vulnerable the application is regarding account security. The reliability of this test is high thanks to the simplicity of the attack, only requiring the usage of basic spoofing techniques.

### B. Test 2 - MITM on the mobile application with client side manipulation

#### Introduction

The attack vector of this test is also spoofing. This test was performed as an expansion of the first MITM test, to evaluate what the encrypted packets contained. Since the first test showed that the packets are encrypted using SSL, mitmproxy could be used to bypass it.

#### mitmproxy

mitmproxy is a program that allows the user to view http and https requests and responses that pass through their computer thanks to the use of proxy settings or other tools such as ARPspoofer. mitmproxy can also generate CA-certificates.

#### Method

The MITM test is performed to bypass the TLS and SSL encryption used in HTTPS traffic, for this mitmproxy is utilized. The first step in this attack is to set up the attackers computer as the proxy server on the mobile device. When the proxy settings are correctly set up, mitmproxy's CA-certificate can be obtained from the website "mitm.it". The generated certificates are used to notify websites and applications that use HTTPS that the proxy is safe. The final step is to set up a transparent proxy, this is done through IP forwarding and ARPspoofer. To set up IP forwarding mitmproxy's documentation on how to set up a transparent proxy is followed [35]. And to set up ARPspoofer this [36] guide is followed. Because ARPspoofer is used the proxy settings on the attacked device are turned off.

#### Results

The MITM test successfully bypassed the encryption used in HTTPS, as mitmproxy could gather

the HTTPS packets and display the information. The account credentials could partially be gathered from listening to the traffic sent when logging in to the application on the mobile phone via WiFi. The username (email) was displayed in clear text while the password still remained unreadable as seen in Figure 2. The mobile app uses a MD5 hashing function to make the password more safe. The two final digits of the password are removed or hidden, making it harder to reverse the hash.

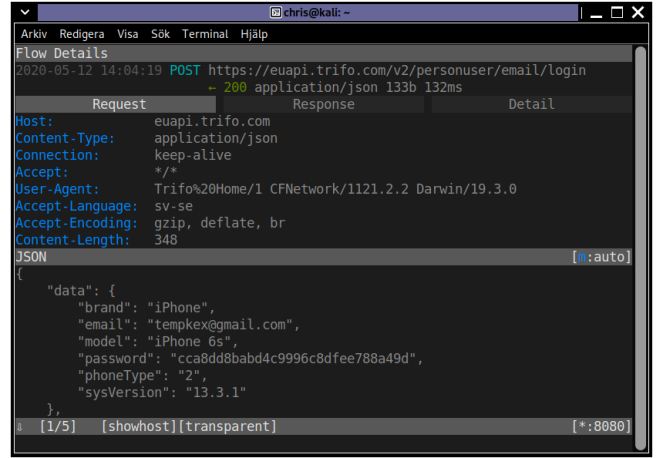


Figure 2: Captured HTTPS login request sent from application to the server displaying the email (Username) in plain text and showing that the password remains encrypted after removing the SSL encryption with mitmproxy.

### Discussion

The MITM attack was used to see the content of the requests sent without the SSL/TLS encryption. But also to see if any further encryption was used to protect the account credentials. The method suggested is believed to be reliable since it is heavily based on mitmproxy's guidelines on how to set up a transparent proxy [35]. These guidelines are made to educate other hackers on how to be able to perform a MITM attack. The results from this test were expected since there has been many examples and studies showing how vulnerable HTTPS is to MITM attacks [37].

### C. Test 3 - MITM attack on the robot's MQTT traffic

#### Introduction

The attack vector for this test is spoofing. The aim of this test is to perform a MITM and a replay attack on the robot vacuum cleaner. The reason behind this test is to verify if the vulnerability found by Checkmarx research team [16] still exists, to exploit unencrypted MQTT traffic by impersonating the MQTT server. Three main programs are used for this test,

ARPspooof, Wireshark and Packet Sender. ARPspooof is used to redirect the packets to the attacker while Wireshark displays all the redirected network traffic and Packet Sender is used to send packets to specified IP addresses and ports.

### Method

The set up for this MITM attack is to start Wireshark on the interface connected to the internet, in this case WiFi. To access the network traffic that is sent from the robot, monitor mode is enabled. After setting up the Wireshark settings correctly, ARPspooof is used to redirect the traffic. Since ARPspooof can disrupt or dramatically slow down the internet connection, IP forwarding is used to speed up the redirection process. When the network traffic is monitored any unencrypted MQTT packets can easily be spotted. To exploit the MQTT traffic a replay attack is performed. This is done through copying a MQTT packet captured by Wireshark and sending it to the robot directly or to the robot's MQTT broker or to a MQTT broker created by the attacker (this can be done using Mosquitto).

### Results

The results gathered from this MITM attack is that no explicit MQTT packets can be identified within the network traffic. During the test the only traffic being sent or received by the vacuum cleaner's IP address was UDP broadcasts and unreadable TCP packets, which can be seen in Figure 3. No common patterns were found in the TCP packets, neither could any valid information be seen in the data segment of the packets due to the encryption. When trying to exploit the MQTT traffic with the use of a replay attack, no successful commands could be sent to the robot or it's MQTT broker or the MQTT broker created using Mosquitto.

[ ip.addr==192.168.1.116 && !tcp.len==0						
No.	Time	Source	Destination	Protocol	Length	Info
288	46.307112931	18.184.201.211	192.168.1.116	TCP	141	10883 → 41908 [PSH, ACK]
289	46.307135814	18.184.201.211	192.168.1.116	TCP	141	[TCP Retransmission] 10883 → 41908 [PSH, ACK]
290	46.307150443	18.184.201.211	192.168.1.116	TCP	141	10883 → 41908 [PSH, ACK]
291	46.307156299	18.184.201.211	192.168.1.116	TCP	141	[TCP Retransmission] 10883 → 41908 [PSH, ACK]
296	46.694101185	18.184.201.211	192.168.1.116	TCP	224	10883 → 41908 [PSH, ACK]
297	46.694137070	18.184.201.211	192.168.1.116	TCP	224	[TCP Retransmission] 10883 → 41908 [PSH, ACK]
298	46.700978777	192.168.1.116	18.184.201.211	TCP	58	41908 → 10883 [PSH, ACK]
299	46.7009825108	192.168.1.116	18.184.201.211	TCP	58	[TCP Retransmission] 41908 → 10883 [PSH, ACK]
300	46.701687072	192.168.1.116	18.184.201.211	TCP	1514	41908 → 10883 [ACK] Seq=
301	46.701709973	192.168.1.116	18.184.201.211	TCP	1514	[TCP Retransmission] 41908 → 10883 [ACK] Seq=
[Checksum Status: Unverified]						
0000	08 78 73 2d eb f5 20 32 33 55 75 1f 00 00 45 00	.xs... 2 3Uu...E.				
0010	00 2c 1d a0 40 90 40 00 78 04 c0 a8 01 74 12 b8	, @ -...t.				
0020	c9 d3 a3 b4 2a 83 9c e6 b1 09 16 82 c0 03 56 18	...*...i...P.				
0030	0a ac 5e 5c 00 00 40 02 75 08	..A\..@ u.				

Figure 3: The Ironpie m6's network traffic captured and showed using Wireshark

### Discussion

The purpose of this test was to verify if the MQTT packets sent by the robot still remained unencrypted,

which was stated by Checkmarx Research team [16]. This test was performed due to multiple threats using this insecure communication to exploit the system. For example using the traffic to calculate the client ID and through this ID being able to impersonate the MQTT server. The ID can also be used when monitoring the traffic and through this obtaining a devkey which can be used to unencrypt all of Trifo's network traffic [16]. Since the results did not align with the conclusions stated by Checkmarx, that the MQTT traffic is unencrypted. An additional test had to be performed to validate that the described method for the MITM attack was correct and that general MQTT traffic actually could be sniffed. This additional test used two computer with one running an MQTT publisher and the other running an MQTT subscriber both connected to the MQTT broker HIVEMQ. The source code used for the publisher and subscriber was taken directly from the MQTT broker HIVEMQ's client library encyclopedia [38]. By using the described method for this additional test most if not all MQTT traffic could be seen which validates that the method for this MITM attack works. Since the MQTT traffic can be seen using this test, but not during the MITM attack on the robot it could mean that the MQTT pings sent from the robot are encrypted and seen as TCP packets instead.

To bypass this encryption a replay attack was performed to test if the MQTT traffic could still be exploited even though it was encrypted. However this attack failed, using a fake MQTT broker created using Mosquitto and redirecting the traffic to it did not accomplish anything due to the fake broker not comprehending the encryption. Trying to impersonate the MQTT server through the use of client ID like how the Checkmarx research team did was not possible either due to the encryption. And trying to send the MQTT commands to the robot's own MQTT broker resulted in nothing. None of the MQTT commands could be sent directly to the robot either since no MQTT ports were open.

### D. Test 4 - Dictionary attack on open SSH port

#### Introduction

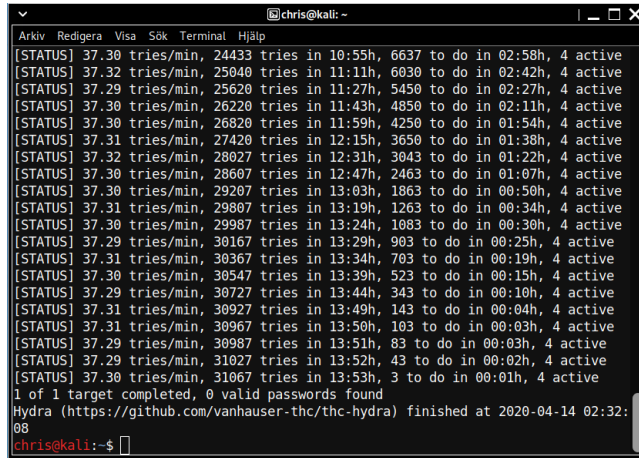
The attack vector that this test is meant to target is elevation of privilege. This test is performed to evaluate how secure the robots root access is. This test is executed with the help of the password guessing tool Hydra which can perform automatised dictionary attacks on IP addresses and ports. Nmap was used to scan the network for used IP addresses and open ports.

## Method

This dictionary attack is performed by scanning for the targeted device's IP address and then afterwards scanning for open ports using Nmap. If the SSH port is open, a Hydra dictionary attack can be launched targeting the device's IP address and the SSH port. To perform a dictionary attack, lists of passwords and usernames need to be acquired first. In this test two password lists are used; "John the Ripper" and "rockyou". Both lists consists of simple and commonly used passwords. For the username list, Nmaps included username list is used.

## Results

The first dictionary attack used the "John the Ripper" password list and finished after 14 hours. The second dictionary attack using "rockyou" ran for 96 hours but was cancelled prematurely. It was cancelled prematurely due to the lists size requiring the test to run for approximately 50000 hours. Both attacks failed since no username and password combination tested were valid. The result of the first dictionary attack can be seen in Figure 4.



```
chris@kali: ~  
[STATUS] 37.30 tries/min, 24433 tries in 10:55h, 6637 to do in 02:58h, 4 active  
[STATUS] 37.32 tries/min, 25040 tries in 11:11h, 6030 to do in 02:42h, 4 active  
[STATUS] 37.29 tries/min, 25620 tries in 11:27h, 5450 to do in 02:27h, 4 active  
[STATUS] 37.30 tries/min, 26220 tries in 11:43h, 4850 to do in 02:11h, 4 active  
[STATUS] 37.30 tries/min, 26820 tries in 11:59h, 4250 to do in 01:54h, 4 active  
[STATUS] 37.31 tries/min, 27420 tries in 12:15h, 3650 to do in 01:38h, 4 active  
[STATUS] 37.32 tries/min, 28027 tries in 12:31h, 3043 to do in 01:22h, 4 active  
[STATUS] 37.30 tries/min, 28607 tries in 12:47h, 2463 to do in 01:07h, 4 active  
[STATUS] 37.30 tries/min, 29207 tries in 13:03h, 1863 to do in 00:50h, 4 active  
[STATUS] 37.31 tries/min, 29807 tries in 13:19h, 1263 to do in 00:34h, 4 active  
[STATUS] 37.30 tries/min, 29987 tries in 13:24h, 1083 to do in 00:30h, 4 active  
[STATUS] 37.29 tries/min, 30167 tries in 13:29h, 903 to do in 00:25h, 4 active  
[STATUS] 37.31 tries/min, 30367 tries in 13:34h, 703 to do in 00:19h, 4 active  
[STATUS] 37.30 tries/min, 30547 tries in 13:39h, 523 to do in 00:15h, 4 active  
[STATUS] 37.29 tries/min, 30727 tries in 13:44h, 343 to do in 00:10h, 4 active  
[STATUS] 37.31 tries/min, 30927 tries in 13:49h, 143 to do in 00:04h, 4 active  
[STATUS] 37.31 tries/min, 30967 tries in 13:50h, 103 to do in 00:03h, 4 active  
[STATUS] 37.29 tries/min, 30987 tries in 13:51h, 83 to do in 00:03h, 4 active  
[STATUS] 37.29 tries/min, 31027 tries in 13:52h, 43 to do in 00:02h, 4 active  
[STATUS] 37.30 tries/min, 31067 tries in 13:53h, 3 to do in 00:01h, 4 active  
1 of 1 target completed, 0 valid passwords found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-04-14 02:32:08  
chris@kali:~$
```

Figure 4: Results of a dictionary attack performed using Hydra with John the Rippers password list and Nmaps username list.

## Discussion

This dictionary attack was performed to test if root access could be gained through the open ports, namely the SSH port. This tests reliability and probability of success increases with the length of the dictionary used. But increasing the length will also increase the time it takes to perform the test, making it a rather inefficient way to gain root access. There has been similar methods with success in gaining root access through dictionary attacks, but also failures [39].

## E. Test 5 - Tampering with the software update

### Introduction

The targeted attack vector for this test is tampering. The aim of this test is to evaluate if the previously found vulnerability within the update function discovered by the Checkmarx research team still exists [16]. The main tool used for this test is Burp Suite. Burp Suite allows the attacker to forward, drop and modify requests sent from the mobile application to the server. This test is performed on both Android (Samsung S4) and iOS (iPhone 7). The application version used for both phones is 2.0.6.

### Method

Since Trifos mobile application mainly communicates via HTTPS a proxy server had to be used on both phones that redirected the traffic to a computer running Burp Suite. The method to set up proxy settings on the Android phone is described in PortSwiggers tutorial [40]. By using Burp Suite as a proxy and downloading the Burp Suites CA-certificate at "http://burp", security measurements like TLS and SSL can be avoided. After the proxy has been setup a software update can be requested as long as the phone does not run the newest version. This request can then through Burp Suite be tampered with and redirected so that it downloads a file from the attacker. For this a server is setup on the attacker's computer that can handle the request and send out a response containing the attacker's file. The server used for this attack is just a proof of concept server written in Java and the code can be seen in the Appendix 3.

### Results

The test failed on the iOS device since the applications update goes through the App store instead of Trifo's own servers. On the Android device the test was successful. The update request did not redirect to Google Play and instead went though Trifos own servers. During this test the request was successfully hijacked and the attackers file was downloaded which can be seen in Figures 5 and 6.

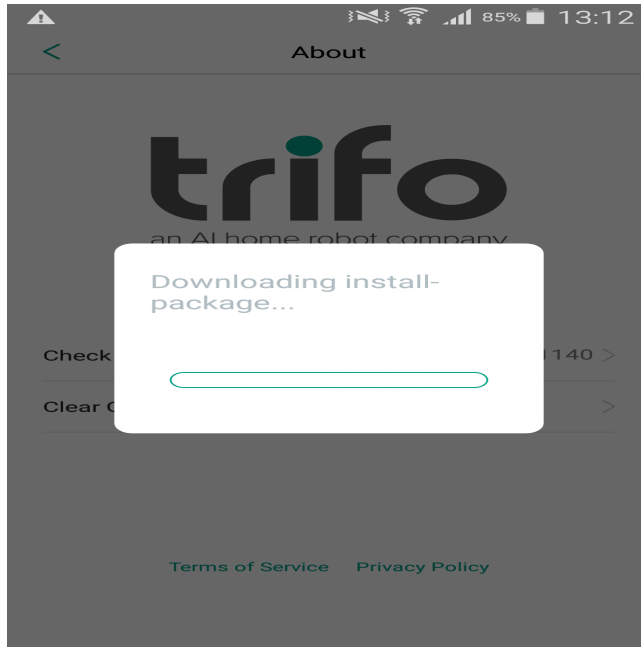


Figure 5: The hijacked software update downloading the attackers file.

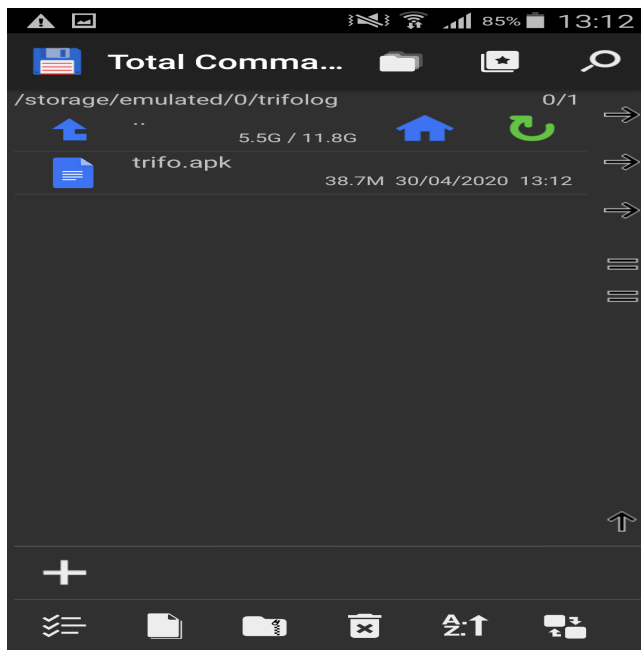


Figure 6: A file manager showing that the attackers file was downloaded.

### Discussion

This attack was performed to validate if a previously found vulnerability within the software update function still exists [16]. A successful execution of this attack allows the application to download unintended files. This

is a risk factor since the files can have malicious intent for example a virus or a compromised version of the application. This attack has a high reliability on Android devices, but it requires the hacker to set up multiple steps correctly. For example one step is to configure proxy settings on to the users phone and another one is to set up a HTTP response server that can handle the software updates request. The main problem with this test is that the attacker has a limited time window where it is possible to tamper with the request. If it takes too long the application cancels the software update resulting in an error.

### F. Test 6 - Read and Modify the APK files

#### Introduction

The attack vector of this test is reverse engineering. The Android Package Kit (APK) is a compressed folder that contains the resources and a dex file that contains the source code of a Android application. The aim of this test is to analyze the source code for vulnerabilities and to potentially modify the code. By using online APK decoders or downloadable tools like Jadx or APKTools, the APK file can be uncompressed and decoded. To access the source code the dex file needs to be decompiled, this can be done by using tools such as Jadx or dex2jar. Static code analyzers such as "Findbugs" can be used when analyzing the source code to simplify the process of finding potential bugs.

#### Common source code vulnerabilities

Finished projects and applications often consist of a great amount of code causing multiple vulnerabilities to be overlooked, these are some common ones [41]:

- The use of hard-coded credentials such as passwords, usernames and API keys.
- Bugs in the source code, i.e. Buffer and Integer overflows or incorrect use of comparisons.
- The use of third party dependencies, these may be outdated or contain flaws that can be exploited.
- If and how the application uses encryption to transfer sensitive data, no encryption or an outdated encryption method can be exploited.
- If some function sends user data to the database, then injection can potentially be used to trick the database to send, delete or change data in the application.



## Method

To access the source code, the APK file had to be decoded and the contained dex file had to be decompiled. This can all be done by utilizing Jadx, by using the graphical user interface (GUI) version of Jadx the APK file when opened is automatically decoded and the contained dex file is decompiled. The GUI version will therefore allow the user to view the source code and resources inside the tool. Jadx also includes a deobfuscation tool that can make the code easier to read. Since the source code has an abundance of class files, only a few of the core functionalities of the applications activities are studied extensively. For example the Android manifest xml file, MQTT server files, login and software update procedures are examined. When examining the functionalities special attention is paid towards finding hard-coded credentials and finding any new information regarding the encryption methods used. To navigate through the files, the inbuilt search tool is used. A static code analysis program Findbugs is also used throughout this test to find and highlight potential bugs.

## Results

The findings from this test showed that the source code was obfuscated. This obfuscation could be removed with some degree of success using the deobfuscation tool included in Jadx. But it did not make the source code completely readable, leaving multiple functions and some classes with cryptic names. Due to the obfuscation no weaknesses were identified and the code could not be modified in any meaningful way. No credentials such as password, usernames, account or client IDS were hard coded in the source code. Only previously known security measures were found such as MD5, SSL, SHA1 and RSA. The scan performed by FindBugs resulted in 14 "Scariest", 110 "Scary", 162 "Troubling" and 18 "Of Concern" bugs being found. Out of the 14 scariest bugs, eight were due to the IDE used not supporting certain android packages thus causing false positives. The remaining bugs were due to poor java practices; Two bugs were found due to misuse of Javas inbuilt equals method comparing two different data types. Two bugs were found due to an incorrect way of using Javas inbuilt data type "Set" having a set contain itself, which resulted in a if state always being false. One bug was caused when self assigning a variable to itself. The final bug was found in a stop function, calling a method where the value returned from the method never is used. None of the found bugs were tested for exploitability, rather the ones that were not false positives were tested for logical failures and how they could impact the whole.

## Discussion

This test was performed to find potential vulnerabilities in the source code through finding out which security measures has been implemented and locating severe bugs. The results from this test showed that the source code was protected from reverse engineering attacks by obfuscating the code. Although the code was partly deobfuscated, no sensitive information could be retrieved and no bugs that could be used to exploit the application were found. The method used for this attack is simplistic and only requires the attacker to use Jadx to get the source code and FindBugs to search for bugs. But the results from analyzing the source code and investigating them may vary depending on the skill level of the person that is analyzing them.

## VII. Sustainability and Ethics

One way to maintain the integrity of a study and to stay ethical is to follow established guidelines such as IEEE's code of ethics [42]. But being an ethical hacker also comes with big challenges, the main one being on how to find and present vulnerabilities in a way that does not infringe upon peoples privacy or break any laws. The main law that affected this study and had to be followed was "Brottsbalken 4 kap. 9c" Lag (2014:302) [43]. This is the law that prohibits people from hacking others property. To avoid breaking this law all the hacking done throughout the study only affected personal devices or devices owned by KTH i.e. no other vacuum cleaners or servers were hacked. Private networks were used throughout the study to avoid disrupting any other peoples network traffic since it would get slowed down during some of the penetration tests. When publishing vulnerabilities there always exists the risk of rebound effects, one such effect would be that hackers can and would exploit the vulnerabilities before the company manages to patch them. This problem can be avoided by contacting the company about the vulnerabilities instead of publishing them.

## VIII. Results

After performing the penetration tests one vulnerability was found. By querying for a software update the attacker could make the mobile application download their own file. One minor vulnerability was also found during a MITM attack on the mobile application with client side manipulation. The MITM attack showed that the username could be retrieved as plain text and that the password as a not fully complete MD5 hash. When performing a MITM attack on the robot device no explicit MQTT traffic was seen but multiple TCP packets were seen, which is believed to be encrypted

MQTT pings. The MQTT traffic could not be exploited, replaying captured MQTT commands failed during the replay attack, and the MQTT server could not be impersonated. No valid combination of passwords and usernames were found when performing the dictionary attacks on the open SSH port. The final test showed that the decompiled APK file was obfuscated, causing no vulnerabilities to be found. There was no use of hard coded account credentials in the source code and none of the bugs found by "Findbugs" were exploited. A summary and overview of the threats can be seen in the threat traceability matrix found in Appendix 4.

## IX. Discussion

The results from the penetration testing phase correlated well with the probability of success described in section V (Threat model). In the first two penetration tests MITM attacks were performed with intention of gaining account credentials at the login screen. These tests also contributed information regarding the encryption status of the communication and if there were any other additional safety measures used. The results from these two tests were that HTTPS is used to secure the communication and that the password is further protected using MD5 hashing. These tests have a high reliability with little deviation of the results thanks to the abundant availability of documentation on how to perform the attack for example [35][36].

In the third penetration test a MITM attack was performed on the robot, to see if the MQTT traffic sent from the robot still remains unencrypted and exploitable which was stated by Checkmarx research team. No explicit MQTT packets were identified during the MITM attack which did not align with Checkmarx's results. This can be due to the packets now being encrypted and Wireshark not being able to differentiate them from regular TCP packets. The reliability of this test is believed to be high since another test was performed to validate that the described method worked properly. The MQTT traffics exploitability was further tested by performing three replay attacks of which none succeeded.

In the fourth penetration test two dictionary attacks were performed to evaluate how secure the access to root functionalities are. To protect the robots root access the more secure SSH is used instead of telnet, SSH has the strength of using encryption over all internet traffic reducing the possibility for hackers to listen to the traffic. SSH can also restrict IPs from trying to log in which can diminish the threat that brute force attacks poses. The results from these dictionary at-

tacks showed that no simple default password is used to protect the root access since none of the dictionary attacks were successful. This kind of attack has a high reliability due to the fact that brute force will always give the same result if the attacker uses the same input.

The fifth penetration test was performed to examine if a vulnerability previously found by the Checkmarx research team still exists [16]. This vulnerability exist within the software update function and can be used to force the application to download unintended files. The result from this test can imply that Trifo has a lack of time, money or that they prioritize other things. The proof of concept server used in this test is rather unreliable mainly due to the fact that packet loss can occur resulting in incomplete file transfers. On the other hand the method to tamper with requests using Burp Suite is reliable thanks to the simplicity.

The sixth penetration test was performed to identify vulnerabilities in the mobile applications APK file and to potentially modify it. This was done by decoding the APK file and then decompiling the dex file through the use of Jadx, the code could then be analyzed both manually and by using tools such as FindBugs. Although a thorough manual analysis was performed on a few core functionalities in the mobile application, no additional vulnerabilities or unknown security measures were found due to the code being obfuscated. When using the tool Findbugs on the source code, it resulted in 14 "scariest" bugs being found. The bugs were not exploited in this test, they were only tested for logical failures. The reliability of this test is dependant on three main factors; the extent of the source code that has been analysed (Quantity), the parts that has been analysed (Quality) and finally the attackers skill level.

The research question of this study was to evaluate which security measures were used in the Trifo Ironpie m6 and through this find out how secure it is. Due to the delimitation's established for this study no hardware aspects were investigated. Instead most of the testing revolved around finding what security measures are used in the mobile application and what measures are used to secure the robots wireless communication. The results showed that Trifo's Ironpie m6 uses multiple security measures such as obfuscated source code, SSL, SSH and hashed passwords. Although these security measures are used one vulnerability was found that could be exploited if the attacker had access to the victims phone. The hacker has to have access to the phone because HTTPS is used for all sensitive network traffic and a CA-certificate has to be installed to circumvent this.



## X. Conclusion

This study was performed with the objective of evaluating how secure the robot vacuum cleaner Trifo Ironpie m6 is. This was done by first creating a threat model and then during the penetration testing phase identifying which security measures are used to protect against these threats. To protect against MITM attacks the mobile application used HTTPS to communicate sensitive data to the server and the passwords are hashed using MD5. The MQTT packets sent from the robot are now believed to be encrypted. To protect against Brute force attacks on open ports the more secure SSH port is used instead of telnet. Finally to protect the mobile application against reverse engineering, the source code was obfuscated. Although these measures are used one vulnerability was found in how software updates are performed on the Android OS. Therefore in conclusion if any hardware or server attacks are disregarded since they weren't tested, the system is secure against basic attacks that doesn't require any client side manipulation. But if an attacker manages to get passed the HTTPS encryption through the usage of generated CA-certificates the security drops significantly and can cause threats towards the privacy of consumers.

## Acknowledgements

We want to thank and express our sincere gratitude towards our supervisor Professor Pontus Johnson and our examiner Associate Professor Robert Lagerström for their continuous supportive feedback and guidance throughout this thesis.

## XI. Appendix 1 - Threat Traceability Matrix for mobile applications

<b>Threat category</b>	Unsafe communication	Insufficient Cryptography	Poor quality of code	Code tampering	Reverse engineering
<b>Threat Agent</b>	An attacker that has access to the local network that the target is located on.	An Attacker that has access to improperly or weakly encrypted data.	An attacker that can send inputs to the mobile application during method or function calls.	An attacker that can trick or make the user download and install foreign programs or applications.	An attacker that has access to the mobile applications APK files.
<b>Affected Asset</b>	Mobile application	Mobile application	Mobile application or Firmware	Mobile application	Mobile Application
<b>Attack</b>	MITM attacks	Reverse engineering or MITM attacks	Reverse engineering	Code tampering & Phishing attacks	Reverse engineering
<b>Attack surface</b>	WiFi	WiFi	WiFi & the source code	APK & WiFi	APK
<b>Attack Goal</b>	Sniff or steal packets in the insecure network to steal credentials such as usernames and passwords.	Successfully decrypt sensitive data.	Due to poor quality of the source code, vulnerabilities/bugs are found and exploited.	To Perform binary changes to the APK files or to make the system API's execute foreign code.	Find information regarding how the data is encrypted or how functionalities such as the device, mobile application or back end servers are set up.
<b>Impact</b>	Account theft, which gives the attacker control of the robot.	Account or information theft, which can give the attacker access to the robot.	Remote server endpoint DoS and/or execution of foreign code.	Information theft and unwanted programs or features can be installed.	Communication between back end servers and the application is compromised.
<b>Security Controls</b>	Use trusted CA-certificates and SSL/TLS to protect the communication.	Avoid storing sensitive data on the mobile application & Use well secure and well known encryption methods.	Prevent buffer overflows by validating the lengths of incoming data.	Detect code changes at run-time and react appropriately to them.	Obfuscation of the source code to make it harder to understand and analyze.
<b>Related Work</b>	Jisiwei i3 hacked due to poor communication security [15]. Successful MITM attack on Xiaomi RoboRock S55 [44].	Dyson 360 Eye has no encryption for local network traffic [45].	Neato's Botvac previously had a buffer overflow vulnerability in its firmware [46].	Ironpie m6's application update can be forced to download foreign code [16].	In Xiaomi's mobile application "Xiaomi Smart Home Set" sensitive data could be found in multiple log messages[44].

## XII. Appendix 2 - Threat Traceability Matrix for robot vacuum cleaners

<b>Threat category</b>	Weak or hard coded passwords	Unsafe update mechanism	Unsafe data transfer	Lack of physical security	Injection
<b>Threat Agent</b>	An attacker that has access to the local network that the target is located on.	An attacker that can identify when the client updates their device and intercept the request.	An attacker that has access to the local network that the target is located on.	The attacker that has access to the device.	An attacker that can send data to an interpreter.
<b>Affected Asset</b>	Robot vacuum cleaner	Firmware or Mobile application	Robot vacuum cleaner	Robot vacuum cleaner	Robot vacuum cleaner & Server
<b>Attack</b>	Brute force attacks	MITM attack	MITM attack	Disassembly	SQL or command injections.
<b>Attack surface</b>	Open ports	WiFi	WiFi	External ports and the data storage	Code flaws & WiFi
<b>Attack Goal</b>	Gain unauthorized access to the system by brute forcing weak passwords.	To hijack firmware updates on the system or software updates on the mobile application.	By listening to the unencrypted communication sensitive data between the server and robot can be seen.	Access the data stored on the device or through the use of the external ports update the device with malicious code.	To inject unauthorized code that is executed.
<b>Impact</b>	The attacker can gain root access to the device, giving full control of the robot to the attacker.	The attacker through their modified firmware gains control of the system, or through their modified application steals information.	A local attacker can through impersonating the server take control of the device.	Information theft or control of the device can be achieved due to a lack of physical security.	The attacker can gain control of the device or cause data loss and corruption of files.
<b>Security Controls</b>	Strong passwords, use of encryption.	SSL or TLS is used to encrypt the communication.	SSL or TLS can be used to encrypt the communication.	The device is hard to disassemble, no unnecessary external ports and ensuring that the external ports that exist can not be used maliciously.	Using safe APIs, removing legacy code and using positive server side validation.
<b>Related Work</b>	The Diqee 360 robot vacuum cleaner used a hard coded admin password which was easily brute forced.[11]	Trifo's mobile application "Trifo home" was hijacked during an software update. [16], the firmware update of the Xiaomi Mi robot vacuum cleaner could be hijacked and the attackers own firmware could be installed since the password used for encryption was weak.[13].	No encryption used for MQTT traffic in the local network, for either Dyson's 360 eye and Trifo's Ironpie m6 [45] [16].	The Diqee 360 vacuum cleaner could be made into a spying device by disassembling the device and inserting a microSD card [11]. The Xiaomi Mi robot's flash drive could be read or written to by short circuiting some of the processes on the motherboard [13].	Neato's Botvac could be exploited and execute unauthorized commands by using command injection in the "ntp" field as previously seen in [14].

### XIII. Appendix 3 - Proof of Concept Server code

```
import java.net.*;
import java.io.*;

public class HTTPAsk {
    public static void main( String[] args) throws IOException {
        ServerSocket serverSocket = new ServerSocket(80);
        File myFile = new File("/home/$USER/Downloads/Trifo.apk");
        while(true) {
            byte[] buffer = new byte[1024];
            BufferedInputStream bis = new BufferedInputStream(new FileInputStream(myFile));
            Socket connection = serverSocket.accept();
            BufferedWriter out = new BufferedWriter(new OutputStreamWriter(connection.getOutputStream()));
            OutputStream os = connection.getOutputStream();
            BufferedOutputStream bos = new BufferedOutputStream(os);
            out.write("HTTP/1.1 200 OK\r\n");
            out.write("\r\n");
            out.flush();
            try{
                for(int bytesToRead;(bytesToRead = bis.read(buffer)) != -1;){
                    bos.write(buffer, 0,bytesToRead);
                }
            }catch(IOException ex){
                ex.printStackTrace();
            }
            finally {
                bis.close();
                bos.close();
            }
            connection.close();
        }
    }
}
```

#### XIV. Appendix 4 - The Results Threat Traceability Matrix

<b>DREAD Score</b>	14	13	13	12	11
<b>Affected Asset</b>	Mobile application & Robot	Mobile application	Robot	Mobile application	Robot
<b>Attack</b>	MITM	MITM	Dictionary attack	Reverse engineering	MITM
<b>Attack surface</b>	WiFi	WiFi	WiFi Open ports	Mobile application	WiFi
<b>Attack Goal</b>	Gain account credentials or clientID.	Tamper with the software update request, so that the target downloads a malicious version.	Gain root access to the robot via open ports.	Analyzing the apps security risks and to add malicious content to the APK.	Impersonate the robots MQTT server.
<b>Impact</b>	Gain full control of the mobile app and therefore control of the robot. Gain clientID and therefore the ability to send fake commands to the robot.	Gain full control of the mobile app and therefore control of the robot. A malicious application can hold the mobile hostage.	Gain full control of the robot.	Identify hash or encryption methods which could benefit the success rate of other pentests. The modified APK, if downloaded could hold a mobile hostage and send sensitive data back to the attacker.	Gain full control of the robot through impersonating the server and therefore having the ability to send commands.
<b>Security Controls</b>	SSL, TLS and MD5 hashing.	SSL, TLS and App Store(iOS)	SSH	Obfuscated source code.	SSL/TLS
<b>Attempted</b>	Penetration test one, two and three.	Penetration test five.	Penetration test four.	Penetration test six.	Penetration test three
<b>Probability of success</b>	High	High	Low	Low	Low
<b>Confidence of success probability</b>	Section V (Threat model)	Section V (Threat model)	Section V (Threat model)	Section V (Threat model)	Section V (Threat model)
<b>Results</b>	With the use of CA-certificates a username in plain text & password hashed with MD5 can be seen. No explicit MQTT traffic were seen.	Successfully downloaded an unintended file during a software update.	No successful combination of username and password found.	No additional security measures were found in the source code. Modification of the code not attempted due to obfuscation. 14 "Scariest" bugs were identified by using the static code analyzer tool Find-Bugs.	Due to encryption no client ID could be calculated and used to impersonate the MQTT server.

## References

- [1] Patil S et al. “Ethical Hacking: The Need for Cyber Security”. In: *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*. Chennai: IEEE, 2017, pp. 1602–1606.
- [2] Meneghello F et al. “IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices”. In: *IEEE Internet of Things Journal*. Vol. vol. 6. no. 5. IEEE, Oct. 2019, pp. 8182–8201.
- [3] Sadio O, Ngom I, and Lishou C. “Lightweight Security Scheme for MQTT/MQTT-SN Protocol”. In: *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. 2019, pp. 119–123.
- [4] Sirohi P, Agarwal A, and Tyagi S. “A comprehensive study on security attacks on SSL/TLS protocol”. In: *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*. Dehradun: IEEE, 2016, pp. 893–898.
- [5] Kimmatkar K and Shrawankar U. “Modified address resolution protocol for delay improvement in distributed environment”. In: *2013 Students Conference on Engineering and Systems (SCES)*. 2013, pp. 1–5.
- [6] *SSH Protocol*. SSH. URL: <https://www.ssh.com/ssh/protocol/> (visited on 04/15/2020).
- [7] Williams R et al. “Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach”. In: *IEEE International Conference on Intelligence and Security Informatics (ISI)*. Beijing: IEEE, 2017, pp.179–181.
- [8] Seralathan Y et al. “IoT security vulnerability: A case study of a Web camera”. In: *20th International Conference on Advanced Communication Technology (ICACT)*. Chuncheon-si Gangwon-do: IEEE, 2018, pp. 1–2.
- [9] L. Rutledge R, K. Massey A, and I. Antón A. “Privacy Impacts of IoT Devices: A SmartTV Case Study”. In: *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*. Beijing: IEEE, 2016, pp. 261–270.
- [10] Strielkina A, Kharchenko V, and Uzun D. “Availability models for healthcare IoT systems: Classification and research considering attacks on vulnerabilities”. In: *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*. Kiev: IEEE, 2018, pp. 58–62.
- [11] Paganini P. *Experts disclose dangerous flaws in robotic Dongguan Diqee 360 smart vacuums*. Security affairs. July 20, 2018. URL: <https://securityaffairs.co/wordpress/74592/hacking/hacking-smart-vacuums.html> (visited on 04/21/2020).
- [12] Austin M. *Is your high-tech robot vacuum letting hackers into your home?* Digitaltrends. Oct. 28, 2017. URL: <https://www.digitaltrends.com/home/lg-hom-bot-vacuum-hacked/> (visited on 04/21/2020).
- [13] Drozhzhin A. *Xiaomi Mi Robot vacuum cleaner hacked*. kaspersky daily. Jan. 4, 2018. URL: <https://www.kaspersky.com.au/blog/xiaomi-mi-robot-hacked/19309/> (visited on 04/21/2020).
- [14] Kielpinski J. *Security in a Vacuum: Hacking the Neato Botvac Connected, Part 1*. 2018. URL: <https://www.nccgroup.trust/us/about-us/newsroom-and-events/blog/2018/march/security-in-a-vacuum-hacking-the-neato-botvac-connected-part-1/> (visited on 05/19/2020).
- [15] Olsson T and L. Forsberg A. “IoT Offensive Security Penetration Testing. Hacking a Smart Robot Vacuum Cleaner”. Bachelor Thesis. KTH Royal Institute of Technology, 2019.
- [16] Umbelino P. *Checkmarx Research: Smart Vacuum Security Flaws May Leave Users Exposed*. Checkmarx. Feb. 26, 2020. URL: <https://www.checkmarx.com/blog/checkmarx-research-smart-vacuum-security-flaws-leave-users-exposed> (visited on 04/21/2020).
- [17] *Common Vulnerability Scoring System version 3.1: Specification Document*. FIRST. URL: <https://www.first.org/cvss/specification-document> (visited on 04/21/2020).
- [18] Hussain S, Erwin H, and Dunne P. “Threat modeling using formal methods: A new approach to develop secure web applications”. In: *2011 7th International Conference on Emerging Technologies*. Islamabad: IEEE, 2011, pp. 1–5.
- [19] Shevchenko N et al. *Threat modeling: a summary of available methods*. Tech. rep. Carnegie Mellon University Software Engineering Institute Pittsburgh United ..., 2018.
- [20] Johnson P et al. “pwnPr3d: An Attack-Graph-Driven Probabilistic Threat-Modeling Approach”. In: *2016 11th International Conference on Availability, Reliability and Security (ARES)*. 2016, pp. 278–283.

- [21] Khan R et al. "STRIDE-based threat modeling for cyber-physical systems". In: *IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*. Torino: IEEE, 2017, pp. 1–6.
- [22] Sonia, Singhal A, and Banati H. "Fuzzy Logic Approach for Threat Prioritization in Agile Security Framework using DREAD Model". In: *International Journal of Computer Science Issues* vol. 8.no. 1 (July 2011), pp. 182–190.
- [23] A. Ingalsbe J et al. "Threat Modeling: Diving into the Deep End". In: *IEEE Software* vol. 25.no. 1 (Jan. 2008), pp. 28–34.
- [24] Aaron G and Aditya G. *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices*. Packt Publishing Ltd, 2017.
- [25] Allen L, Heriyanto T, and Ali S. *Kali-Linux - Assuring Security by Penetration Testing*. Packt Publishing Ltd, 2014.
- [26] Patil A et al. "A multilevel system to mitigate DDOS, brute force and SQL injection attack for cloud security". In: *2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)*. Indore: IEEE, 2017, pp. 1–7.
- [27] Pingle B, Mairaj A, and Y. Javaid A. "Real-World Man-in-the-Middle (MITM) Attack Implementation Using Open Source Tools for Instructional Use". In: *2018 IEEE International Conference on Electro/Information Technology (EIT)*. Rochester,MI: IEEE, 2018, pp. 192–197.
- [28] Hendricks B. *Replay Attack: Definition, Examples & Prevention*. URL: <https://study.com/academy/lesson/replay-attack-definition-examples-prevention.html> (visited on 05/25/2020).
- [29] Khonji M, Iraqi Y, and Jones A. "Phishing Detection: A Literature Survey". In: *IEEE Communications Surveys Tutorials* 15.4 (2013), pp. 2091–2121.
- [30] *OWASP Mobile Top 10*. OWASP. 2016. URL: <https://owasp.org/www-project-mobile-top-10/> (visited on 05/11/2020).
- [31] *Internet of Things (IoT) Top 10 2018*. OWASP. 2018. URL: <https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf> (visited on 05/14/2020).
- [32] O'Donnell L. *Unpatched Security Flaws Open Connected Vacuum to Takeover*. threatpost. URL: <https://threatpost.com/unpatched-security-flaws-open-connected-vacuum-to-takeover/153142/> (visited on 04/15/2020).
- [33] A. Assegie T and S. Nair P. "COMPARATIVE STUDY ON METHODS USED IN PREVENTION AND DETECTION AGAINST ADDRESS RESOLUTION PROTOCOL SPOOFING ATTACK". In: *Journal of Theoretical and Applied Information Technology* vol. 97.no. 16 (Aug. 2019), pp. 4259–4269.
- [34] *WELCOME TO THE ETTERCAP PROJECT*. ettercap-project. URL: <https://www.ettercap-project.org/> (visited on 04/07/2020).
- [35] *Transparent Proxying*. mitmproxy. URL: <https://docs.mitmproxy.org/stable/howto-transparent/> (visited on 04/07/2020).
- [36] *MITM using arpspoof + Burp or mitmproxy on Kali Linux*. 2017. URL: <https://www.valbrux.it/blog/2017/11/26/mitm-using-arpspoof-burp-or-mitmproxy-on-kali-linux/> (visited on 05/08/2020).
- [37] Huang H et al. "A look into the information your smartphone leaks". In: *2017 International Symposium on Networks, Computers and Communications (ISNCC)*. Marrakech: IEEE, 2017, pp. 1–6.
- [38] Light R. *Paho Python - MQTT Client Library Encyclopedia*. HIVEMQ. 2015. URL: <https://www.hivemq.com/blog/mqtt-client-library-paho-python/> (visited on 05/18/2020).
- [39] Kakarla T, Mairaj A, and Y. Javaid A. "A Real-World Password Cracking Demonstration Using Open Source Tools for Instructional Use". In: *2018 IEEE International Conference on Electro/Information Technology (EIT)*. 2018, pp. 0387–0391.
- [40] *Configuring an Android Device to Work With Burp*. 2020. URL: <https://portswigger.net/support/configuring-an-android-device-to-work-with-burp> (visited on 05/08/2020).
- [41] Sangwan S. *Finding vulnerabilities in Source Code*. 2019. URL: <https://medium.com/@somdevsangwan/finding-vulnerabilities-in-source-code-8575ece385dc> (visited on 05/19/2020).
- [42] *7.8 IEEE Code of Ethics*. IEEE. URL: <https://www.ieee.org/about/corporate/governance/p7-8.html> (visited on 05/05/2020).
- [43] *Brottsbalk (1962:700)*. Riksdagen. URL: [https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/brottsbalk-1962700\\_sfs-1962-700](https://www.riksdagen.se/sv/dokument-lagar/dokument/svensk-forfattningssamling/brottsbalk-1962700_sfs-1962-700) (visited on 05/05/2020).

- [44] Clausing E. *From the Land of Smiles – Xiaomi Roborock S55*. 2019. URL: <https://www.iot-tests.org/2019/02/from-the-land-of-smiles-xiaomi-roborock-s55/> (visited on 05/11/2020).
- [45] Clausing E. *Dyson 360 Eye – The Rolls Royce of Robot Vacuums?* 2019. URL: <https://www.iot-tests.org/2019/02/dyson-360-eye-the-rolls-royce-of-robot-vacuums/> (visited on 05/11/2020).
- [46] Ullrich F et al. “Vacuums in the Cloud. Analyzing Security in a Hardened IoT Ecosystem”. MA thesis. TU Darmstadt, 2019.



TRITA-EECS-EX-2020:208