Bachelor's thesis

Bachelor of Engineering: Information and Communications Technology

2021

Dorde Obradovic

CYBERSECURITY OF IOT SYSTEMS

– Analyzing Weak Points in a Cloud-Supported Embedded Environment

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Bachelor of Engineering: Information and Communications Technology.

2021 | 29, 0

Dorde Obradovic

CYBERSECURITY OF IOT SYSTEMS

- Analyzing security vulnerabilities in a Cloud Supported Embedded System Environment

Nowadays, the Internet of Things (IoT) industry is developing at a rate faster than ever. The projections state that the number of devices which are internet-connected will go up to as far as 43 billion, in only three years from the time of writing of this thesis. IoT is rapidly adopted in the enterprise sector as well as in the smart-home consumer sector.

The main objective of this thesis was to build a prototype of an internet and cloud-connected device, analyze its vulnerabilities, and finally suggest security improvements for consumer IoT devices.

Literature research was conducted and used as a reference when assessing the prototype. Physical, communication and application security were the three main domains of vulnerabilities the thesis focused on. The prototype consisted of a Raspberry Pi model 3B+, which collected data through a sensor and uploaded it to the ThingSpeak cloud service. The DREAD and STRIDE threat rating models were used to identify the prototype's most exploitable attack surfaces and attack types.

The prototype was successfully built with an established cloud connection, its security was thoroughly analyzed, and suggestions were given on mitigating the most vulnerable parts of it. The suggestions included: keeping the firmware up to date, using encryption wherever possible, limiting unused features, limiting data stored online when possible, and not using default configurations and credentials.

Considering the rapid growth of the IoT industry, it is of great importance that current and future smart-home users gain awareness of the potential threats to their home networks, and to set up their devices accordingly.

KEYWORDS:

IoT, Embedded, Cloud, Cybersecurity

CONTENTS

PICTURES

TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
CVSS	The Common Vulnerability Scoring System
DDOS	Distributed Denial of Service
GDPR	General Data Protection Regulation
IoT	Internet of Things
IP	Internet Protocol
JTAG	Joint Test Action Group
OS	Operating System
OWASP	The Open Web Application Security Project
SD	Secure Digital
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UART	Universal Asynchronous Receiver-transmitter
UDP	User Datagram Protocol
UML	Unified Modeling Language
USB	Universal Serial Bus
VPN	Virtual Private Network

1 INTRODUCTION

The Internet of things (IoT) is a system of interconnected computer devices, including digital and mechanical machines. Their main distinction from embedded devices is that those devices can transfer data over a network, without the need for human to human, or even human to computer interaction. The rise of the embedded system sector, as well as machine learning and real-time analytics, have all shaped and evolved the definition of IoT. The IoT sector has been experiencing substantial growth lately and is predicted to become a front runner in the field of information technologies; The number of businesses that have made use of IoT technologies, went from 13 percent in 2014, to about 25 percent currently (Dahlqvist, Patel, et al., 2019). The projections state that the number of devices which are IoT-related will increase to as far as 43 billion, in only three years from now. That will mark a threefold increase in just five years backed up by heavy investment into the sector (Analyticsinsight, 2019).

The term “Smart Home” is often synonymous with IoT technologies, in the consumer market. Home CCTV, temperature sensors, internet connected fridges, as well as smart speakers, all fall into the category of smart home appliances. It is an ecosystem of home assistant devices the user is in control of. Most of these already exist or are in early prototype phases. However, smart home has not yet become an everyday standard of our lives. Governments are also planning on making smart cities, by incorporating smart devices in the public transport sector, including traffic lights and parking utilities. The healthcare sector has also been adopting embedded systems, with the use of medical sensors and monitors.

The greatest concern with having more internet-connected devices transferring and storing data to cloud databases, is the privacy and security of IoT device users and their data. More data online means more exposure to vulnerabilities and hackers. We have seen massive cyber-attacks so far, such as the Mirai botnet (Kilpatrick, 2018). They can happen on a large infrastructure scale, as well as application-based user data leaks (Bekker, 2020). The attacks are most commonly a Denial of Service (DDoS), or data stealing malware.

With the introduction of mainstream internet, there have been moral debates over data protection. The European Union’s new GDRP law has become a famous example in protecting its internet users’ rights. Mega-corporations such as Facebook and Google have been involved in numerous court cases and have received fines of around one billion euros. The phenomenon of cybersecurity has been around for a long time; however, the sensitivity of the data being protected will increase. Smart home devices will store user personal, private data. Therefore, cybersecurity is becoming of great importance in our lives. It is evident that manufacturers are producing more and more IoT devices without an adequate approach to the security aspect.

This thesis focuses primarily on the Cybersecurity aspect of consumer IoT systems. The thesis aims to examine what security vulnerabilities exist currently and what the end users can do to protect our future smart systems. Enterprise IoT was considered in this

thesis; the perspective is set on how the rise of smart homes will influence users/consumer privacy and security. Overlapping security issues with previous older generation technology are also considered, especially if the user has the possibility to reduce the risks.

The initial vulnerability and case studies identified from the literature research come were classified divided into the following three segments: physical device security, communication security and application security.

For the practical part in Chapter 3, the system in case is a Raspberry Pi model 3B+, which has collected data through a sensor and uploaded it to the ThingSpeak cloud service. A DHT11 temperature and humidity sensor was used, and the program was written in Python and ran from the Raspbian operating system. The prototype's vulnerabilities were analyzed, by comparing its security features against the research. Different choices of communication protocols that the cloud software offered were compared. The operating system used was also analyzed for vulnerabilities and ways of hardening it were provided. This practical analysis aimed to highlight what users have control over when it comes to mitigating the risk of their data being hacked.

Threat modeling was carried out by defining the prototype's attack surface layers, which were then used as elements in the STRIDE threat model. The DREAD threat model was used to score the prototype's vulnerabilities against common IoT attacks. A common vulnerability scoring system (CVSS) was used to determine the most vulnerable attack surfaces as well as the most damaging types of attacks. The suggested security measures against these types of attacks were provided.

The thesis is structured as follows: Chapter 2 introduces case studies and information about vulnerabilities within IoT. Chapter 3 compares a prototype internet connected embedded system against those same vulnerability metrics from chapter 2. Chapter 4 introduces threat models and examines areas of improvements. Chapter 5 gives suggestions about how the smart home systems could be further secured with regards to the research.

2 VULNERABILITIES IN IOT

IoT incorporates a vast amount of technological elements. While this allows for a wide range of functionalities, it also creates a magnitude of security weaknesses. An IoT device typically consists of the following layers: network, application, mobile and cloud. Each of these layers have existing security vulnerabilities that need to be regulated. The network layer's encryption and firewall need to be ensured, the application should have a proper authorization protocol and input validation. These two requirements are standard for software programs with internet connectivity. Once the cloud and mobile layers are factored in, the system architects must now deal with questions such as the security and integrity of the APIs being used in the cloud and mobile platform, as well as their encryption and authorization as well.

In this chapter, various IoT vulnerabilities are researched, which will then be used as a reference in chapter 3, where a prototype IoT device's security is examined.

2.1 Physical device security vulnerabilities

In the consumer technology field, IoT devices and their hardware such as sensors, are usually handed over to the customer. For example, smartphones are sold or lent to the consumer who takes the device home. This means that the phone, along with its hardware (sensors, actuators) and firmware are hard to secure. The company does not have immediate control over the product, which makes it hard to secure and easy access to hackers. Therefore, the engineers would try to make their product difficult or impossible to reverse engineer. Apple phones are known to be almost impossible to authenticate once stolen, thanks to their cloud-based security using public key services. This currently seems like the best option, however, there are trade-offs when having too much data controlled by one company.

Discussion about cybersecurity is often primarily focused on hacking into software over a network, penetration testing. However, physical security of an IoT device should be regarded with a great deal of care as well. Constrained devices can be found in remote locations where the physical security of the device could be hard to set up. Some of the vulnerabilities of having remote hardware are:

- Damage to the device.
- Having the device stolen.
- Cables being disconnected, disabling the communication.
- Power being cut off, disabling the device.
- Access to information leading to manufacturer's product manual.

If possible, it is always a good idea to have video surveillance on the remote device. Surveillance laws must be considered, however if it is in a company's premise, or the consumer owns the property, that should not be a problem. The idea of having surveillance means the potential thief could be identified. However, theft is not the only scenario possible. To ensure that the device does not get tampered with and malfunction, using an anti-tamper seal would help. It would be providing a visual on whether the system has been compromised. If a sensor were to be moved even slightly, it could cause it to lose calibration and produce bad and misleading results for the user. A lot of embedded system run on SD cards nowadays. If such a device were reached by criminals, they could potentially steal or destroy data from it.

In August 2017, an article on hacking firmware of IoT devices was published by Wired magazine (Newman, 2017). It outlined how to gain access to the systems firmware on many devices. The hack used eMMC flash technology, which is hybrid memory consisting of storage and programmable flash memory. By soldering five wires to the flash chip and using an SD card reader, the IoT device's software, operating system and firmware were able to be retrieved. With all of those copied and saved to a PC, they could be examined for vulnerabilities within the code, leading to further development of hacks on similar devices. After this article was published, millions of devices were affected. Manufacturers have since deployed patches to fix the problem, but it is believed that a lot of the systems involved are still vulnerable.

Hacks such as this one illustrate the need for implementing physical device security. With every revelation of the systems code, attackers will try to dig deeper and use their findings further. This can be achieved by downloading the firmware, modifying it, and re-uploading it to the device with a backdoor or other newly introduced vulnerability. Vulnerabilities of one device, can easily be transferred over to similar devices or devices which are a part of the same manufacturer. This creates a chain link effect.

2.2 Communiation security vulnerabilities

An embedded system is an application specific, electronic system design powered by a processor or controller. It usually takes input with sensors and gives an output (reading or actuator). The flow of data from the sensors or external modules, back to the main processor or controller is very important. Width of the bandwidth, direction and order of flow need to be considered. In embedded systems, various types of protocols are used for this, according to the design specifications. The wired protocols include Serial peripheral interface (SPI), inter-Integrated Circuit (I2C), universal asynchronous receiver-transmitter (UART), joint Test Action Group (JTAG) and controller area network (CAN). Wireless communication protocols include Zigbee, Bluetooth and Message Queuing Telemetry Transport *(*MQTT). The wireless communication protocols are widely used in IoT, as they enable the internet connectivity.

MQTT has support for Secure Sockets Layer (SSL), which encrypts the data being transferred online. The traditional hypertext transfer protocol (HTTP) has worked well so far as networking protocol, however in IoT it has its down sides. HTTP is great for

requesting and acquiring information, like when a client wants to get information from a source. The problem arises when a source should push a change to many clients. The text-based format requires more bandwidth, and every source that wants to push changes needs to have its own server. This can be problematic in IoT, as keeping a web server on to answer the requests will drain a lot of battery in the IoT device. MQTT solves these problems, with its publish and subscribe method of communication. Any source such as a sensor can publish data, while clients can subscribe to it. The communication is taken care of by a broker. The data is not in the form of text, but binary. This drains a lot less battery power, which is a scarce resource.

While MQTT and REST both have SSL possibilities, the encrypting has not been adopted as much as it should in the IoT sector. A report from Zscaler (Pergament, 2019) has shown that **60%** of IoT transactions are occurring on clear text HTTP. That is quite a lot of insecure transactions, considering that IoT traffic has reached around 1 billion monthly transactions. Clear text HTTP means that the transactions are not encrypted, as well as the authentication when connecting to that network. These kind of security weaknesses allow hackers to scan and enter the rest of the users' network. The act of an intruder intercepting network packets is called a man-in-the-middle attack. The intruder could simply collect information or redirect it.

Wired communication protocols in embedded systems are used by attackers to root the device. This occurs when root-level user access to the OS is obtained. An attacker with such access has complete control over the instrument. She can read the data, modify it, or delete any file on the system. This is usually a very dangerous level of access, when in the wrong hands.

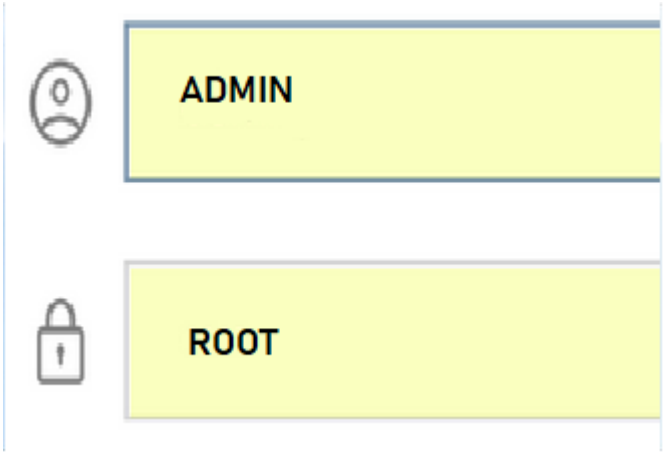
After gaining access to the hardware, attackers could make use of communication protocols between devices and modules to gain administrator access of the firmware. UART is a wired communication protocol that is found in much hardware. Most IoT technology has a system where authentication is required when trying to link external modules to the main board. However, some insecure systems have not implemented that layer of security, which enabled external hardware to access and run commands on the main device. There were instances of smart refrigerators and IoT doorbells falling victim to this method of hacking.

JTAG is a common hardware interface that provides your computer with direct communication with the onboard chips (Senrio, 2019). The JTAG and UART interfaces are common attack vectors for gaining root access to the device. A laptop, tablet or similar tool can be used to connect to the IoT target platform, using either JTAG or UART serial interface. This can be done as simple as connecting the wires to the receive, transmit and ground connector pins. Which enables her to read the memory and modify the firmware of the embedded system.

2.3 Application security vulnerabilities

The necessary embedded software that IoT devices run is called firmware. Firmware contains a minimal OS and programs that control the device, related to the OS. IoT device firmware can contain security exploits just like other software such as websites or larger operating systems. These exploits and vulnerabilities are usually discovered after their release. With the market expanding rapidly, there have been countless examples of various IoT hacking and security problems. Hacking and security exploits come in many forms and sizes, however firmware vulnerabilities seem to be prevalent. Firmware vulnerabilities are like vulnerabilities of network devices and other computer systems. They include the following:

Default Username and Password



Picture 1. Example of insecure default credential login.

The majority of consumer IoT attacks are made possible or aided by the fact that the login credentials remained default values, such as those in picture 1. To change them, the user must first link with the device by establishing a connection. Commonly, the manufacturers will set a default username and password such as admin and admin. These default passwords can very often be found in the manual of the product, which is one web search away. In these manuals, it is also normally strongly advised to change and personalize the credentials. It is evident from cases such as the Mirai botnet (Kilpatrick, 2018) that many users skip this step. Doing so, they have allowed a hacker to gain access very quickly. Adding a weak password might prolong the attacking process, as the attacker would have to make use of a password challenging software. To fully secure their system, users must have strong criteria for their credentials. Not only that, but every internet connected device in the network should have its own unique password.

Distributed Denial of Service (DDoS) Attacks

DDoS attacking requires the use of botnets, which are unified systems composed of infected devices. Botnets are often global, compromising systems all over the world. Because of the vast use of default passwords, IoT systems are common targets for botnets. A hacker could make a linked list between product lines and default passwords, and then make a script that would automatically log in to remote IoT systems over the internet. The script would most likely brute force the most common default credentials. Once the script manages to authenticate, it could then also infect the system with malware. The Mirai botnet attack is the latest biggest example of a DDoS attack involving IoT. It was a series of attacks in 2016, that affected default credential IoT systems running Linux, such as IP cameras and home routers. Millions of devices were affected, and the attack disrupted services of major corporations such as Twitter, the Guardian, Netflix, CNN and many others in Europe and the United States.

Outdated Firmware

If an attacker is attempting to target a specific device or branch of IoT devices, he might check if the firmware is not updated to the latest release. He could also check if there are any exploits that have yet to be addressed in an upcoming patch. The source code of the Mirai botnet was published online, for the purpose of learning from it and securing IoT systems based on it. Unfortunately, this allowed copycat actors to use the same hack on devices which have not been patched yet (outdated firmware). It also allowed them to build on top of the hack, developing it further and affecting different kind of devices.

Patching and updating the firmware of an IoT device, to fix security vulnerabilities, is an important component of secure networking. The aspects to consider when dealing with firmware updates are how many devices will get updated, whether the updates can be automated, encryption of the updates and whether the update is coming from an authentic source (signed and verified).

IoT cybersecurity has not been able to keep up with the rapid growth rate of IoT applications and hardware. In a lot of cases, patches do not even exist for the exploits of the device. For example, one of the first internet connected devices was a Wi-Fi kettle, allowing remote boiling of water. It was later discovered that it had major security flaws which allowed hackers as far as into the home network. Once in the home network, the hacker could intercept packets over the network and compromise the rest of it. Projects such as Wardrive, where security experts drive around scanning networks to map them online, further enable hackers to locate where such insecure IoT kettles are located. However, these kettles were not designed to be updatable. The company behind the product ended up releasing two more versions of it, before it became truly secure. A company may be responsible for numerous IoT devices, even up to tens of thousands of them. Updating and patching a big number of devices is a challenge on its own (Boehm, 2020).

Buffer Overflow Attacks

A buffer overflow attack happens when software developers fail to neglect the size of the maximum input a user can make. An attack of this kind could cause data to be corrupted, DDoS, or it could even enable malicious code into the system being targeted.

Backdoor Installation

A backdoor installation is usually installed after an attacker manages to remotely access the IoT device. Backdoor is type of malware that avoid normal authentication steps to access the system, giving perpetrators the power to remotely issue commands and update malware on the system. Netcat is a command that could be used on a Linux-based operating system for writing to network connections using TCP or UDP. Using this, the perpetrator can cause attacks over the network. Linux is a very popular OS for embedded and IoT systems. This is because of its open-source nature as well as its ability to be stripped down and made into a smaller lightweight OS. Developers must ensure that their Linux distribution makes use of the kernel security features such as privilege levels, file permissions and network security. Lack of these features in IoT devices running Linux is what enabled the Mirai botnet.

One recent and famous example of a Windows OS backdoor install came in the form of a ransomware attack in 2017. A hacker group by the name of Shadow Brokers leaked a backdoor implant tool call EternalBlue, which was created by the U.S. National Security Agency (NSA). This enabled a threat actor to use the backdoor implant tool to lock computers and demand bitcoin payments. It was possible due to a vulnerability in outdated firmware; it was only possible on older vulnerable Windows versions. The attack was worldwide, affecting more than 200,000 computers and causing damages estimated up to a billion dollars. The attack allegedly originated in North Korea, with several western countries concluding so. The devastating damage demonstrates the importance of keeping operating system software up to date, as well as using a good antivirus and firewall. A strong network monitoring tool would also help take care of suspicious activity. The WannaCry attack only affected desktop computers, the monetary damages were mostly business and government facilities that could not operate due to the denial of service. A computer for personal use which gets ransomed would not be as bad, because it most likely would not have important documents. With IoT on the rise and our cars, medical equipment or kitchens becoming internet connected, the ransom factor of those would be worse by a much bigger magnitude. What if we get fired from our jobs because our car is locked, and a hacker demands payment for it to be unlocked? Or the devastating effects medical and kitchen IoT failures would leave us with. These are important issues we must consider when slowly transitioning to a world full of smart devices.

Encryption

Any internet connected device which relays data needs to be encrypted. When devices or facility machines communicate with each other without encryption, they provide a serious doorway for crackers to steal data information, deliver harmful updates and even take control of the system.

Nowadays, websites that store sensitive data like passwords usually implement hashing. Hashing is the act of taking a plain text input and changing it to another value, which is not readable by humans. Hash functions ensure that the generated value can

only be decoded through hash lookup tables, which can be in the form of an array, database, or other data structures. Ideally, a hash function should be non-revertible, which means that it should not be possible to reverse engineer it. Reverse engineering a hash function would allow hackers to decode users' passwords and see them in plain text. Unfortunately, passwords nowadays are mostly compromised in this manner. Every few months, large websites with numerous users gets its database of hashed passwords made public.

Just in 2020 already, there seems to have been at least 11 significant data leaks. Below is a list of some of those, from Wikipedia (Table 1).

Table 1. List of significant data leaks from 2020, (Bekker, 2020)

Entity	Records	Organization Type	Method
Microsoft	280,000,000	Technology	accidentally exposed
T-Mobile	Unknown amount of sensitive customer data	mobile carrier	hacked
General Electric	280,000	Technology	Data breach
Zoom	500,000	Technology	Hacked
Facebook	267,000,000	phone accessories	Data leak
Nintendo	160,000	Gaming	Hack / poor security
Activision	500,000	Gaming	Hack / poor security
U.S. Marshals	387,000	Government	Hack / poor security

The data got exposed accidentally, hacked, or hacked through poor security. If that data happened to be hashed, crackers would run their tools against it, in hopes to unmask that data. Not all the database data is hashed, but passwords most likely are. Cyber-criminals benefit from cracking big, hashed databases of passwords because they can learn from them:

- What the most common passwords are. (Which they can try using elsewhere)
- What methods and techniques people use to try and harden their password codes (for example adding numbers / using symbols).

- How often unique passwords are used (around 0.0001% of the time).

This gives the crackers an accurate view of ways in which users choose their passwords and gives them a better chance of cracking more passwords in return. They can for example use the findings to operate brute force attacks, which use extensive computer power to automatically enter and guess words.

One common technique criminals use to brute guess passwords is a dictionary attack. This type of attack will try every entry from a customized word list. It could be an English language dictionary, or perhaps a list of words that frequently come up online. Advanced dictionary attacks implement the techniques users use to harden their passwords:

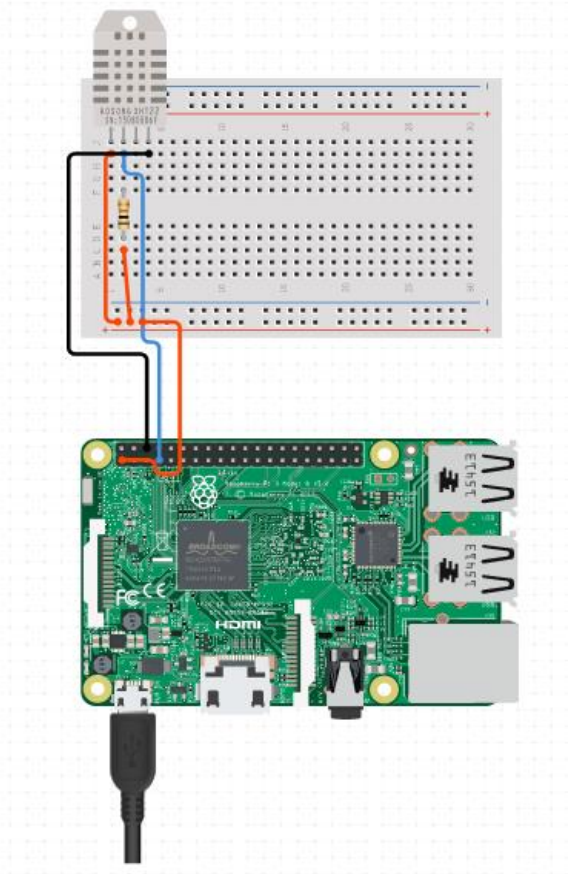
- Using more words stringed together.
- Adding numbers at the end of words.
- Adding symbols before, after, or even between words.
- Replacing letters with symbols ($ instead of s, 3 instead of e... etc.).
- Adding the platforms name onto the end of the password.
- Any many more, guessing billions of times per second.

Encryption is one of the most important aspects of cybersecurity, because a well-thought out and long password is still useless if it is transferred in packets of plain text over the network. Without it, it is undefended against man in the middle attacks.

3 CASE STUDY OF EXAMPLE IOT SYSTEM

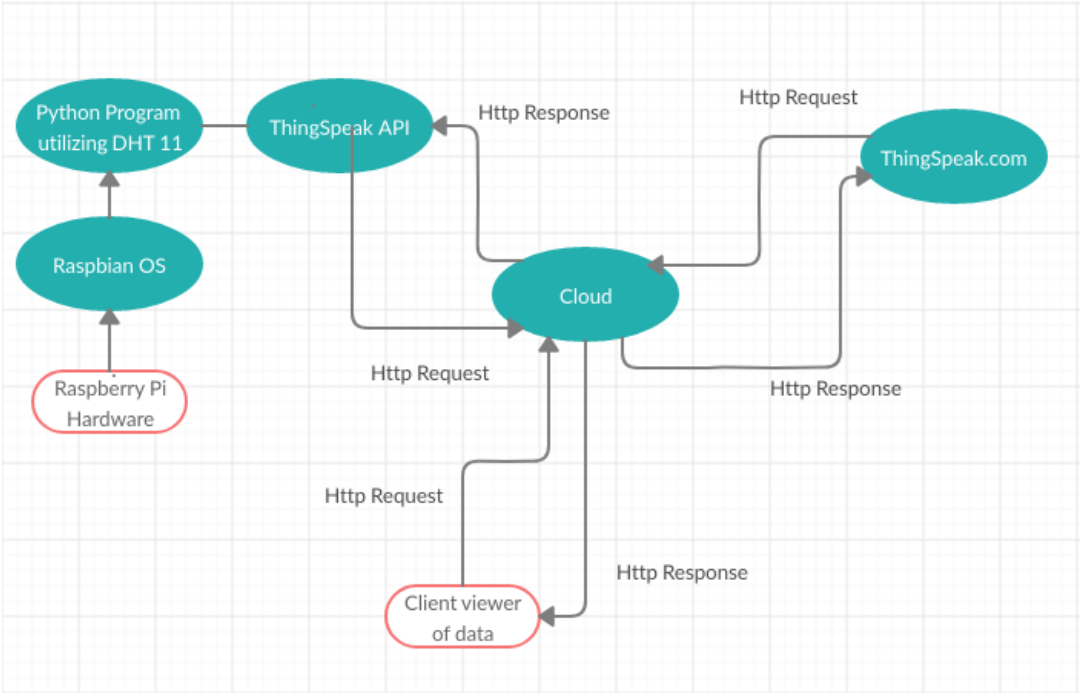
3.1 Raspberry Pi prototype system

A skeleton prototype device has been built, which is connected to a cloud service called Thingspeak. A Raspberry Pi model 3B + was selected as the main driver of the IoT device. The model 3B + is a relatively powerful single board computer, capable of collecting data though general purpose input out pins (GPIO) and uploading it online through its Wi-Fi capabilities. Thingspeak was chosen as the cloud platform because it had good integration with the Raspberry Pi. It was also simple to implement and it received and sent data from the cloud to the device and vice versa. The data being transferred through packets online was the main purpose, as the security of the data was being investigated. The system was also analyzed, for security vulnerabilities in the firmware and physical design. The prototype's schematic can be seen in the picture below.



Picture 2. Schematic of Raspberry Pi system.

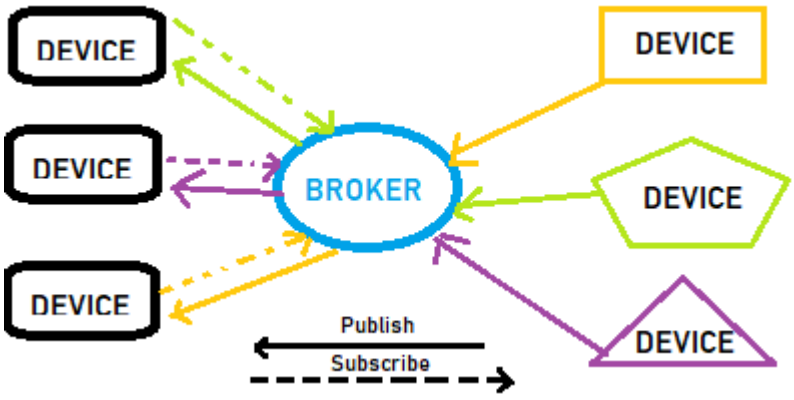
The ThingSpeak API offers two types of data protocols. REST API and MQTT API. An UML diagram showing the data flow of the REST API is used is shown in the picture below.



Picture 3. REST API data flow.

If the REST API were to be selected for the data transfer, the data would flow in HTTP packets. This would be problematic because that would mean that the data is not encrypted. If the packets were to be intercepted, the intruder could read that data in plain text. Fortunately, it is possible to use a secure HTTP (HTTPS), which encrypts the packets. When users want to implement a REST API, they must be careful to select the secure option of HTTPS. This provides greater security.

A data flow diagram of a system using MQTT API is shown in the picture below.



Picture 4. Flow diagram showing MQTT API data flow.

The MQTT API is better suited for IoT for several reasons. It is more suitable for mobile devices as it is a lightweight protocol. It sends data as an array of bytes, and along with its publish and subscribe system it lowers power consumption. MQTT offers support for payload encryption. The payload is the data that is being sent to and returned from API requests, it usually informs the user what the service did or returns data that the user asked for. This information that is directly communicated to the user, can be sensitive data depending on what it holds. While there is support for the payload encryption and HTTPS, there are a lot of vulnerabilities out of the box. Firstly, authentication to the broker is completely optional, it does not need to be implemented. Secondly, even if authentication is used, it is not encrypted by default. The payload and the rest of the network information is also not encrypted by default. On the MQTT website it explained how additional security measures can be implemented by the user, if he were to make use of an application that encrypts the data that MQTT sends and receives. However, this was not built as part of MQTT, because they wanted to keep it simple and lightweight by design.

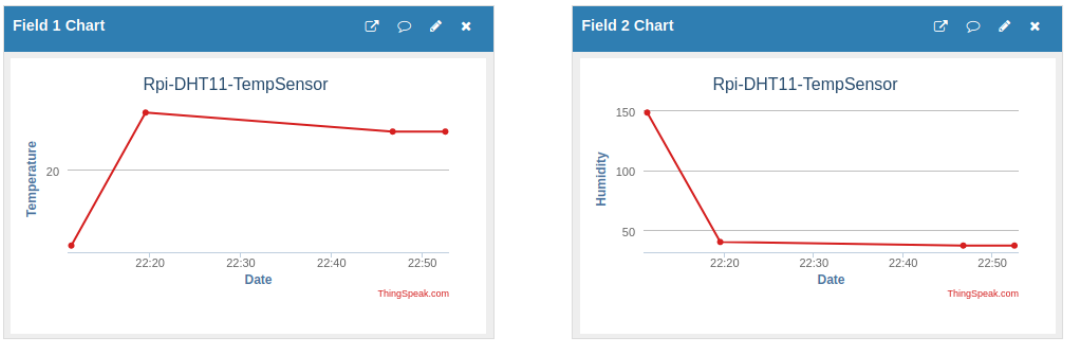
The python program to collect data was simple. At the top of the file, the libraries Thingspeak, DHT and time were included. Thingspeak library was used for the cloud connection, DHT for the temperature sensor and the time library was used for timing the uploads. Thingspeak has a minimum limit of 15 seconds per interval for free accounts, which is what was used in this prototype. The interval is not important as the data integrity which was the measured variable. Below the includes are the declarations for channel id, read and write API keys. These are used when sending the data to ensure authenticity. There are two functions after that and the sensor declarations. The first one is a standard one for collecting the temperature data, and the second one is for using Thingspeak to send it online. The code sample can be seen in the picture on the following page. The channel id and API read and write keys have been purposely left out.

*thingspeak_example.py

```
#!/usr/bin/env python
import thingspeak
import time
import Adafruit_DHT

channel_id =  # PUT CHANNEL ID HERE
write_key  = '' # PUT YOUR WRITE KEY HERE
read_key   = '' # PUT YOUR READ KEY HERE
pin = 4
sensor = Adafruit_DHT.DHT11

def measure(channel):
    try:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        # write
        response = channel.update({'field1': temperature, 'field2': humidity})

        # read
        read = channel.get({})
        print("Read:", read)

    except:
        print("connection failed")


if __name__ == "__main__":
    channel = thingspeak.Channel(id=channel_id, write_key=write_key, api_key=read_key)
    while True:
        measure(channel)
        # free account has an api limit of 15sec
        time.sleep(15)
```

Picture 5. Code for Python program which collects and sends data.

Below are the data points generated by the code above, which ran on the Pi with the sensor. The first data point appears to be faulty, most likely due to the timings and startup of the program.



Picture 6. Representation of data collected on the Thingspeak website.

3.2 Improving the security of the example IoT system

Improvements for the three main layers were given.

3.2.1 Physical improvements

A lot of higher end sensors come with an alarm feature that enables them to alert or trigger an alarm if they happen to be misaligned. When possible, it is good practice to devices with such capabilities. In the context of the Raspberry Pi system, it is possible to implement a low-cost Pi camera which can be programmed to detect physical changes around the sensor. The temperature sensor program can also be modified to send alerts if the sensor is not calibrated properly. This type of tamper proof alarm system could also be used to protect the data on the storage of the device. The prototype example does not have a physical enclosing around it, due it being out of scope of the project. However, it is not unheard of that threat actors destroy the plastic surrounding a device, to steal or destroy data from an SD card or any other memory standard.

Securing the hardware requires less effort when done in an environment that can be controlled easily. Ensuring that the maintainers always have access to the device allows them better physical security over it.

Standard surveillance and security protocols should be implemented as a first layer of defense. Consider implementing battery backup to power IoT devices that rely on fixed power sources in case of power outages.

3.2.2 Communication interfaces improvements

When it comes to choosing communication interfaces, it is recommended to pick those with encryption. It is crucially important for when data is transmitted over the internet as in the case of this prototype. With the ThingSpeak cloud, there are MQTT and REST API options to choose from. Both support HTTPS protocol, and the user should verify that the packets are encrypted, to avoid data being stolen in man in the middle attacks.

3.2.3 Application Security Improvements

In the application security, the upkeep of firmware is the most important measure recommended. Many of the updates to firmware happen for the sake of patching vulnerabilities. When developing larger complex systems which involve multiple IoT instruments, it is recommended that the developers and other IT personnel keep a database of their devices. It would be ideal if that database contained the firmware

versions of the respective devices, and a plan were in place for checking for updates regularly.

The Raspberry Pi system uses the Raspbian operating system, which is a fork of another Linux operating system called Debian. It has been made to be more compact by stripping off unnecessary features. This is a common practice when it comes embedded firmware because the hardware running it is lighter than the one on a typical personal computer. The issue with Raspbian is that unlike Debian, they do not have a security team working on fixing common vulnerabilities and exposures. This means that Raspbian is potentially less secure than other options. Some of its packages are older or outdated, for example the Chromium software. This is because the software has not been ported for the Raspberry Pi hardware. If greater security is required, use of another operating system could be beneficial. However, it will most likely be at the expense of convenience, as the Raspbian firmware is specifically tailored for the Pi's hardware. Other firmware might not run as good. Out of the box, the Raspbian system is secure. There are several steps that are recommended for proper device security. Those are the following:

- Change your default credentials

- Making the use of sudo require a password.

- Ensuring you have the latest security fixes (up to date OS).

- Improving SSH security

SSH is a common way to access Linux devices remotely. It is widely used in IoT and smart home systems. Changing the default credentials has a direct impact on the security of SSH, as the threat actor will not be able to log in. Making sure that the internet protocol (IP) address is not leaked through insecure networks. When connecting to a private IoT device from a public network, it is recommended to make use of a virtual private network (VPN). This is applicable in an instance where the user wishes to configure his smart home remotely from an insecure network. The VPN would ensure protected connectivity by rerouting the traffic through the virtual server rather than the server from which ISP the user is connected to.

The ultimate way to keep sensitive data hidden, is to keep it off the network whenever possible. Separating IoT devices to their own ecosystem network ensures that the user's home network will not get compromised if the IoT one does, or vice versa.

4 THREAT MODELING AND MITIGATION

Threat-modeling methods are used to prevent threats and threat actors from abusing system vulnerabilities. System administrators can make use of those methods to inform about defensive actions. The threat-modeling practices can help obtain a clear overview of the system, make profiles of potential attackers and threats that might come.

A variety of models have been made to rate threats. Combining multiple models produces a more rounded overview of the potential threats and it is good practice. Some models focus only on privacy concerns, some models tend to be more oriented towards people. There is variety in specialty.

Producing threat models should be accomplished early in the development of IoT systems. Potential vulnerabilities could be caught and dealt with earlier, which would prevent more expensive damages later. Therefore, it is important to implement the models during the designing and decision-making phase of the system's architecture.

IoT smart home appliances benefit from threat-modeling because manufacturers of traditional home appliances (fridges, ovens, heating systems...etc.) may not consider the new cyber threats that could come their way.

4.1 Attack surface

A cyber threat attack surface refers to the vulnerabilities in a particular hardware or software environment (Aria Cybersecurity Solutions, 2020). It represents the total number of vulnerabilities and unauthorized user can potentially use access and exploit the system.

It is the developers' and software security professionals' responsibility to minimize the exploitability of the code they produce. End users must also take security into consideration when using the software, especially when it is handling sensitive data. Developers safeguarding their systems might not be enough if the user is opting for insecure configurations and set-ups.

Table 2 shows relevant attack surfaces and their vulnerabilities for the prototype system from chapter 3. Physical vulnerabilities have been left out because it is assumed that the smart home product is stored safely in the user's home.

Table 2. Possible attack surfaces.

Attack Surface	**Vulnerabilities**
Firmware	Hidden backdoor, buffer overflow, unauthorized access.
Log in Credentials	unauthorized access, data breach, elevation of privilege.
System Configuration	Default credentials, improper permissions, and encryption.
Event logs	Information about system leaked to threat actors.
Device Interface	Authentication, authorization, weak encryption, and a lack of input and output filtering
Network Comms	Integrity, authenticity of information, unauthorized remote control of IoT devices, eavesdropping.
Certificates and Keys	Brute force attacks, unauthorized access

4.2 STRIDE

Stride is a model for identifying computer security threats. It was invented in 1999 and currently the most mature threat modeling method. It was adopted by Microsoft and is currently included with their threat modeling tools, even though Microsoft no longer maintains the project. The model has 6 threat categories:

- Spoofing: illegally obtaining and then making use of another user's log in credentials, such as username and password.

- Tampering: tampering with data, for example unauthorized changes to databases, or modifying network traffic.

- Repudiation: Deny performing a malicious action.

- Information disclosure: Privacy breach or data leak, information is exposed to individuals that are not meant to see it.

- Denial of service: Denying or disrupting service to valid users.

- Elevation of privilege: The vulnerability of the attack surface to be penetrated and allow the threat into the trusted system, for example when an unauthorized user gains privileged access.

Table 3. STRIDE threat rating

	S	T	R	I	D	E
Firmware		3		2	3	3
Log in Credidentials	2	2	2	2		2
System Config	2	3		2	3	2
Event Logs	1	2	2	1		1
Device Interface		2		1	2	2
Network Comms	1	2	1	1		2
Certificates and keys		3		2	2	3

CVSS Legend:	
3	High
2	Med
1	Low
0	/

STRIDE has been used to identify potential threats. The attack surfaces worth mitigating risks for additionally include firmware, system config, certificates and keys. Those scored high in the STRIDE model. They have combined scores of 11, 12 and 10, respectively.

A Common vulnerability scoring system (CVSS) score is made up from values an analyst gives, for each metric of vulnerabilities. It is good practice to combine the CVSS scoring with other methods of threat rating; in this instance it was combined with STRIDE metrics. The CVSS rating was given on a scale from 0 to 3 based on the severity of potential impact these threats would cause.

4.3 DREAD

DREAD is a mnemonic and qualitative risk assessment model. The risk refers to the probable frequency and the probable magnitude of a future loss event. The framework is broken into five main categories:

- Damage: Total damage caused to the business.

- Reproducibility: How easy it is to replicate the attack.

- Exploitability: How much time and energy are required to exploit the threat in question.

- Affected Users: How many people will be affected by this threat.

- Discoverability: How easy it is to discover the threat.

Table 4. DREAD model table.

	D	R	E	A	D
Privilege escalation	3	2	3	3	1
Eavesdropping	2	3	2	1	1
Brute-force password attac	2	2	2	1	2
Firmware hijacking	3	1	3	3	3
DoS	1	2	2	3	3

CVSS Legend:	
3	High
2	Med
1	Low

This risk assessment model has been used to assess common IoT cyber-attacks, which have been identified in the research. The attacks include: Privilege escalation, Eavesdropping (man in the middle), Brute-force password attack, Firmware hijacking and denial of service. Dread model is typically used to estimate the damage done to a business because of the attack (Maze, 2018). However, in this case we are dealing with consumer smart home appliances. This means that the damage rating is based on the user's loss, with the assumption that the user is carrying important data over the cloud. There is also the possibility that the threat actor's intent is not only to steal the data, but also to cause damage to the smart home. This was also considered. The attacks with the most potential damage are privilege escalation, firmware hijacking and denial of service. Their scores were 12, 13 and 11, respectively.

5 CONCLUSION

The goal of this thesis was to build a prototype IoT device, analyze its vulnerabilities, and offer valuable suggestions to consumers on how to increase their device's security at home. To do achieve these objectives, literature research was conducted, and an internet connected device was assembled as a prototype. The prototype's physical, application, and communication security were assessed based on the research. The DREAD and STRIDE threat rating models were used to identify the most exploitable attack surfaces and attack types. Security suggestions for the most vulnerable attack surfaces and against all the common IoT attacks have been summarized in Tables 5 and 6 below. The research carried out in this thesis also showed the frequent occurrence and outcomes of these attacks, which warrant extra security measures.

Table 5 summarizes the suggested security measures against the attack types with the highest CVSS scores from the DREAD table in Chapter 4. Table 6 summarizes suggested security measures to mitigate vulnerabilities from the most vulnerable attack surfaces from the STRIDE model.

Table 5. Recommendations based on most severe attack types from the DREAD model.

Attack Type	**Suggested Security Measures**
Privilege Escalation	- Keep critical information on the server side. - Encrypt the data transmitted
Firmware Hijacking	- Keep the firmware up to date - Use a secure and lockable bootloader.
DDoS	- Allow little to no room for user error (check inputs) - Use a secure firewall for the network - Have a response plan and alerts in case a denial of service happens.

Table 6. Mitigation based on STRIDE model.

Attack Surface	Suggested Security Measures
Firmware	- Keep firmware up to date. - Limit connection ports that are not in use, for example USB ports. - Use secure BIOS and bootloaders, which manage and run the operating systems or firmware.
System Config	- Do not use default credentials on your firmware and router. - Use a strong encryption method for Wi-Fi password on your router, like WPA2. - Isolate / separate the IoT network from your home network. - Avoid using plug and play / out of the box settings for your devices. - Disable features you don't need, such as remote access. - Implement 2 step authentication where possible.
Certificates and Keys	- Use SSL where possible. - Store private and public keys, as well as digital certificates, in a secure location.

Considering the fast growth of the consumer IoT industry, it is important for end users to gain knowledge and awareness of possible vulnerabilities to their home networks, and set up their devices accordingly.

This thesis has examined and analyzed in depth the topic of consumer IoT security, has demonstrated it with an internet connected prototype device, and has provide sufficient

recommendations for users to keep their devices safe from threat actors. If every consumer took the time to learn about IoT hacks from the past, the potential vulnerabilities of plug and play IoT devices, and then configured their systems accordingly, the number of IoT hacks would be severely reduced. This goes to show how important this subject is.

In conclusion, the project fulfilled the objectives set. It could be further improved upon by dissecting each attack surface and conducting further research for every smaller element as well.

REFERENCES

Analyticsinsight., 2019. Top 10 Latest Predictions For Iot In 2020. [online] Available at: https://www.analyticsinsight.net/top-10-latest-predictions-for-iot-in-2020 [Accessed 15 March 2020].

Aria Cybersecurity Solutions, 2020. What is a Threat Attack Surface? And How Can You Minimize Your Risk?. Available at: https://blog.ariacybersecurity.com/blog/what-is-a-threat-attack-surface-blog [Accessed 31 March 2020].

Bekker, E., 2020, Jan 03, 2020 Data Breaches | The Most Significant Breaches of the Year . Available at: https://www.identityforce.com/blog/2020-data-breaches. [Accessed 15 March 2020].

Bell, T., 2018. 5 Myths Of API Security. [online] CSO Online. Available at: https://www.csoonline.com/article/3268111/5-myths-of-api-security.html [Accessed 14 December 2020].

Dahlqvist, F., Patel M., Rajko, A. and Shulman, J., 2019. Growing Opportunities In The Internet Of Things. [online] Mckinsey. Available at: https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things [Accessed 5 March 2020].

Boehm, E., 2020. The Top Iot Vulnerabilities In Your Devices - Keyfactor - Security Boulevard. [online] Security Boulevard. Available at: https://securityboulevard.com/2020/10/the-top-iot-vulnerabilities-in-your-devices-keyfactor [Accessed 22 November 2020].

Kilpatrick, H., 2018. 5 INFAMOUS IOT HACKS AND VULNERABILITIES. [online] IOT Solutions World Congress. Available at: https://www.iotsworldcongress.com/5-infamous-iot-hacks-and-vulnerabilities [Accessed 14 April 2020].

Newman, L., 2017. The $10 Hardware Hack That Wrecks IoT Security. [online] Wired. Available at: https://www.wired.com/story/sd-card-hack-iot-zero-days/ [Accessed 10 April 2020].

Marisetty, S., 2019. Five Steps To Successful Threat Modelling. [online] Community.arm.com. Available at: https://community.arm.com/iot/b/internet-of-things/posts/five-steps-to-successful-threat-modelling [Accessed 28 April 2020].

Maze, T., 2018. How To Use DREAD Analysis With FAIR. [online] Fairinstitute.org. Available at: https://www.fairinstitute.org/blog/how-to-use-dread-analysis-with-fair [Accessed 24 November 2020].

Pasknel, V., 2017. Hacking The Iot With MQTT. [online] Morphuslabs. Available at: https://morphuslabs.com/hacking-the-iot-with-mqtt-8edaf0d07b9b [Accessed 14 December 2020].

Pergament, L., 2019. How Much Traffic Is Being Generated By Things (Iot)? | Zscaler. [online] Zscaler. Available at: https://www.zscaler.com/blogs/corporate/how-much-traffic-being-generated-things-iot [Accessed 14 December 2020].

Price, S., 2016. These Are The Weakest Points In Your Iot Security. [online] IoT Central. Available at: https://www.iotcentral.io/blog/these-are-the-weakest-points-in-your-iot-security [Accessed 14 December 2020].

Santos, R., Gongora, E., Adams, D. and Santos, S., 2017. What Is MQTT And How It Works | Random Nerd Tutorials. [online] Random Nerd Tutorials. Available at: https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works [Accessed 7 April 2020].

Senrio. 2019. JTAG Explained (finally!): Why "IoT", Software Security Engineers, and Manufacturers Should Care. [ONLINE] Available at: https://blog.senr.io/ [Accessed 31 March 2020].

Writer, G., 2020. How Encryption Is Powering The Future Of IoT. [online] IoT For All. Available at: https://www.iotforall.com/future-iot-encryption [Accessed 1 April 2020].