NTNU
Norwegian University of
Science and Technology

Internet of Things - Cybersecurity at Home

Tormod Bjørnhaug

Master of Science in Communication Technology
Submission date: June 2017
Supervisor: Lillian Røstad, IIK
Co-supervisor: Eirik Thormodsrud, Sopra Steria

Norwegian University of Science and Technology
Department of Information Security and Communication Technology

Title: Internet of Things - Cybersecurity at Home

Student: Tormod Bjørnhaug

Problem description:

Internet of Things (IoT) devices have lately become accessible for the general consumer. Devices can be bought both on the Internet and over-the-counter. The introduction of IoT has made home-automation, home-surveillance, and home-security easy to implement without any prior knowledge of such systems or Internet-enabled devices. Specialized devices, like IoT devices, are made for a single purpose and have a limited amount of resources built-in. The lack of resources forces a producer to choose what to implement and to what extent. Limited resources can be problematic as prioritizing user-wanted features may reduce security features. Also, users have to go through a device setup process which may affect the initial security settings. Even if an IoT device is considered secure from the time of purchase, security issues are discovered on a regular basis. Updates that remove software glitches and bugs, as well as patch security issues, are to be expected. It is, therefore, reasonable to believe that this allows for a user to change or update their IoT device's settings.

The state of IoT security at home will be considered during this thesis. The thesis will try to establish a best practice through a literary study of the field, as well as find a recommended minimum of implemented security features. Security testing will be done with a set of devices and findings will be used to compare differences in security between different types of devices. Finally, this thesis will consider whether or not a consumer can update or make changes to the device configuration. This last part will also look into the initial configuration of each device.

Responsible professor: Lillian Røstad, IIK

Supervisor: Eirik Thormodsrud, Sopra Steria

Abstract

IoT devices has several times been proven to lack proper security mechanisms. Consumers are often not aware of the risk these devices pose to their privacy and security. This thesis takes on the problem of securing IoT devices targeting consumers through four research questions:

RQ1 What is the current state-of-the-art within consumer-grade IoT security?

RQ2 What security controls are used in consumer-grade IoT devices?

RQ3 How does the type and amount of implemented security controls differ between consumer-grade IoT enabled devices?

RQ4 Can a consumer affect the security of their IoT device through updates or configuration changes?

Through a literary study of the current state of IoT security, two sets of best practices are presented, with a basis in the challenges that consumer-grade IoT devices are facing. The two best practices are divided into a consumer related and a manufacturer related.

To put these proposed best practices to the test, experiments were performed on four different IoT devices, each from a different manufacturer. The selected devices are in this case either electric heaters or baby monitors. Combining the individual experiment results reveals that baby monitors are better at security, but lack the ability of complete continuous operation in case of network failure. Electric heaters, on the other hand, lack support for updates and password complexity. In general none of the devices completely comply with the best practice. End-users can improve security through updates and configuration, but this would, for the most part, require knowledge of, or interest in, technology and IoT.

The lack of a standard guideline, and agreed upon best practice, are two of the reasons for consumer-grade IoT devices lack of security. Since end-users cannot be expected to become proficient in securely configuring their devices, pressure must be put on manufacturers to agree on a IoT security guideline or best practice.

Sammendrag

IoT enheter har flere ganger blitt bevist å mangle skikkelige sikkerhetsmekanismer. Forbrukere er ofte ikke klar over hvordan farene slike enheter påvirker deres privatliv og sikkerhet. Denne oppgaven utfordrer problemet med å sikre IoT enheter som er rettet mot forbrukermarkedet gjennom fire forskningsspørsmål

RQ1 Hva er dagens status innen forbukerrettede IoT sikkerhet?
RQ2 Hvilke sikkerhetstiltak brukes innen forbukerrettede IoT enheter?
RQ3 Hvordan varierer typen og mengden implementerte sikkerhetstiltak mellom forskjellige forbrukerrettede IoT enheter?
RQ4 Kan en forbruker påvirke sikkerheten i IoT enheter gjennom oppdateringer og konfigurasjonendringer?

Gjennom et literaturstudie av dagens status innen IoT sikkerhet blir to forskjellige sett av beste praksis presentert. Disse baseres på nåværende utfordringer som forbrukerrettede IoT enheter har. Settene av beste praksis er både forbukerrelaterte og produsentrelatert.

Beste praksis settes på prøve gjennom eksperimenter med fire forskjellige IoT enheter, alle fra forskjellige produsenter. Enhetene er enten elektriske ovner eller "baby call". Ved å kombinere de enkelte resultatene viser det seg at "baby caller" har flere implementerte sikkerhetskrav enn de elektriske ovnene. Kameraene mangler egenskaper for å støtte kontinuerlig operasjon, mens ovnene i større grad mangler krav til oppdatering og passord kompleksitet. Generelt er det mangel på sikkerhetstiltak på alle enhetene, og ingen følger helt beste praksis. Sluttbrukere kan forbedre sikkerheten på enheter gjennom oppdatering og konfigurasjon, men dette krever stort sett kunnskap og interesse for teknologi og IoT.

Mangelen på en standard, og enighet rundt en beste praksis, er muligens en av grunnene for at forbrukerrettede IoT enheter mangler sikkerhetstiltak. Sluttbrukere kan ikke regnes å ha kjennskap til hvordan å sikkert konfigurere sine enheter, noe som setter et press på

at produsentene må bli enige om en felles IoT sikkerhetsstandard og beste praksis.

Preface

This master's thesis sums up 20 weeks of work and completes a five-year journey on Norwegian University of Science and Technology. It is submitted to the Department of Information Security and Communication Technology as the final stage of fulfilling a Master of Science in Communications Technology.

Firstly, I would like to thank my professor, Lillian Røstad, for allowing me to tackle my own problem description and for pairing me up with my supervisor. Lillian has provided me with support and guidance throughout the thesis.

Second, I would like to sincerely thank my supervisor, Eirik Thormodsrud, from Sopra Steria for his knowledge and experience in the field, as well as almost instant feedback along the way. I would not have been able to complete the thesis without you.

Third, I would like to thank my fellow master student Jon-Anders Kabbe for providing valuable feedback. I would also like to thank Andrew Riege for providing an American-English proofreading.

Finally, I would like to thank you, the reader. If you have made it this far, you have at least read one page of my thesis. Thank you.

Contents

List of Figures

List of Code

List of Tables

Chapter 1
Introduction

1.1 Motivation

During the last ten years, the idea of connecting already computing devices gave way for the concept of "connected things", also known as Internet of Things (IoT)[1]. The term was first created by Kevin Ashton in 1999[2], but has since then evolved into covering a wide spectrum of devices. Enabling communication between a variety of devices is quite intriguing as the possibilities of new ways to combine technology become endless[3]. Whether devices are interacting with each other, the environment, or people, these devices are on the technological frontier - helping people, research, and industries.

The nature of how IoT are used, requires them to have a small form factor so that they can be included in everyday items. The size also substantially limits available resources within a single device[4]. Since IoT devices are often mass produced, the cost of each device is expected to be low. Low cost combined with limited resources make up two of the challenges that contribute to the difficult process of securing IoT. Turning devices off is also proving a challenge, as more and more devices are what connect the user to the Internet[5].

Lately, within the last few years, wearables and connected devices have proceeded to enter the consumer market. IoT include everything from TVs, clocks, refrigerators, air conditioning units and light bulbs. Home automation is a trend that connects the home to the Internet, making it more accessible and more vulnerable than before.

Commercial interests in IoT are rising as new parties can take advantage of

the increased use of home automation and monitoring of their customers. Law enforcement can make sure that people drive under the speed limit, electrical suppliers can better monitor electrical consumption for its customers, and insurance companies can tailor policies for certain patterns of behavior[6].

While there are several stakeholders within the IoT domain, there does not seem to be a real consensus between them on what to expect from IoT devices. IoT lack laws, regulations and a common techonology standard[7]. New IoT devices are created all the time, and ordinary household appliances are turned from "dumb" devices to devices that feel their surroundings and interact with each other and the environment around them.

End-users are becoming dependent on IoT devices. This dependence increases the need for secure devices and a minimum standard of security within IoT eco-systems. Manufacturers must find and follow a security basis for devices to minimize the risk of unnecessary exposure of end-users privacy and information[8], regardless of device type.

1.2 Problem Description

Although IoT security testing is frequently seen at security conventions and in research, there has been little comparison across different genres of IoT devices and the industry itself[9]. Penetration testing results often conclude the testing through the use of known vulnerabilities, or lack of security implementations, without considering how implementations stray from a security best practice.

This thesis focuses on the cybersecurity of commercially available end-user IoT devices used at home. The goal is to make a brief comparison across different devices and manufacturers to figure out how device security mechanisms vary across the industry. Achieving this requires a literary study to research and find a proposed best practice within the IoT state-of-the-art. Finally, comparison of the selected devices and proposed best practice will be used to see if end-users can take measures to improve security of IoT devices in the home.

The goal of the thesis can be compressed into four research questions:

RQ1 What is the current state-of-the-art within consumer-grade IoT security?
RQ2 What security controls are used in consumer-grade IoT devices?

RQ3 How does the type and amount of implemented security controls differ between consumer-grade IoT enabled devices?

RQ4 Can a consumer affect the security of their IoT device through updates or configuration changes?

1.3 Limitations

Although this thesis contains security testing of several devices, it is not meant to be a purchase guide for IoT devices. Results from the testing are neither intended to be a comprehensive security test or to test a specific part of the IoT industry. Security testing is done to provide results based on the proposed best practice and compare these results across different manufacturers within the IoT industry.

IoT devices within a consumer, or end-user, context is the primary focus of this thesis. Devices studied were, at the time of writing, available for purchase in either physical or online stores. The purpose of choosing such devices was to explore a selection of devices that are easily obtainable, and that does not require installation by an electrician or certified personnel. Devices were chosen at random and from different manufacturers to get the broader sense regarding how devices are created from a manufacturer's point of view.

A device's eco-system may be comprised of a web portal, smartphone app and the device itself. As software is a major component of IoT devices, the term will also include firmware and related software such as apps and web portals.

Security testing of IoT devices connected to a standard 802.11 WLAN will not include security testing the network setup itself, only devices connected to it and associated with the IoT device in question.

Contents of this thesis will assume users to have some minimum knowledge of how computer networks work, as network setup is completed in the experiments throughout the thesis. Knowledge about IoT devices and cybersecurity issues related to communicating devices will reduce the complexity of the contents of this thesis but are not a prerequisite.

1.4 Outline

This thesis consists of seven chapters, including this introduction. Chapter 2 presents background information about IoT and explains relevant terms. Secondly, a literary study is done on the state-of-the-art within IoT security, resulting in a set of proposed best practices for the industry and consumers. Third, the methodology for the thesis is presented with an introduction of software tools, experiment setup, and experiment procedure. The methodology is followed by the security testing of four devices and comparing the test outcome with the proposed best practice. Chapter 6 will discuss, based on consumer best practice presented in chapter 3, the steps consumers can take to secure their devices and homes further. Finally, a conclusion sums up the findings regarding cybersecurity at home based on research and experiments completed through the thesis. The conclusion also proposes topics for further work.

Chapter 2
Background

To better understand the upcoming chapters, this chapter will introduce terms and background material that is considered necessary to better comprehend the thesis.

2.1 Defining the Internet of Things (IoT)

Internet of Things is as vaguely defined as the term itself is written. There is not one particular definition of IoT as there has yet to be a consensus on a single definition. This lack of consensus makes defining the term IoT a bit more complex. Although several definitions exist[10, 11, 12], the main idea of IoT as a concept is the ability to connect devices and enable them to interact with each other, the environment, as well as more complex and legacy computing devices[1]. Oxford dictionary[13] defines IoT as the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data. However, IoT devices does not necessarily need to communicate over the Internet but some communication, wired or wireless, is required. IoT devices can be considered an interface between the physical and digital world[14].

2.2 Definitions related to cybersecurity

Three main objectives, often referred to as the *CIA triad* are at the heart of computer security, which is also valid for other computer systems[15]:

Confidentiality

This term covers two related concepts:

Data confidentiality Assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Integrity

This term covers two related concepts:

Data integrity Assures that information and programs are changed only in a specified and authorized manner.

System integrity Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Availability

Assures that systems work promptly and service is not denied to authorized users.

2.3 What is IoT security?

Like any Internet-connected device, IoT devices deal with the same issues regarding security and privacy. However, IoT devices are usually much more resource constrained, because of their size, than conventional end-user devices like laptops, smartphones, and desktop computers. For this reason IoT devices are not always capable of using the same communication and encryption schemes as other end-user devices. The term IoT security sums up both security and privacy issues within the IoT-domain.

Chapter 3
State-of-the-art

This chapter is a literary study to find best practices within state-of-the-art IoT security. Terms regarding IoT security has already been defined and explained in chapter 2. Firstly, this chapter will focus on the current challenges of IoT. Subsequently, this chapter is summed up with two proposed industry best practices to combat those current problems.

Chapter 3 seeks to answer research question: *What is the current state-of-the-art within consumer-grade IoT security?*

3.1 Challenges within IoT security

Challenges within IoT are divided into two parts - device manufacturer and end-user. This section will try to enlighten challenges for both parties. Security is often not a priority for the manufacturer[9] and devices lack built-in mechanisms to receive automatic updates. Only about 10% of organizations are confident that their connected devices are secure[16]. On the other hand, users forget about their devices and are unaware of when, or if, a device is being used for malicious purposes[17].

3.1.1 Unsecure commmunication

IoT devices communicate. Communication is fundamental for IoT but may not necessarily be wireless. Devices can communicate with other IoT devices such as smartphones, computers over the local network or through the Internet, or even to back-end servers. Resource constrained devices, like smartwatches and smart-heaters, can not always easily implement standard protocols and methods

for secure communication. Using standard protocols might not always be feasible due to the risk of being too time-consuming on some devices[18]. Lightweight cryptographic algorithms and protocols become important to tackle IoTs lack of resources[7]. Where possible, symmetric and asymmetric encryption algorithms, like RSA and blowfish, are ideal for IoT due to their low power consumption[19].

Securing communication for IoT has not been a priority[7], which has lead to several attacks[20, 21]. Improper authentication and encryption make devices vulnerable to attacks, including man-in-the-middle attack (MITM). Bad authentication can lead to MITM which enables a third-party to intercept traffic, e.g. between a device and its back-end server, change it or even reply with custom responses. Lack of encryption(plain-text communication) will allow anyone with access to the communication stream to observe and extract data transmitted. This would directly make it an issue related to data leaks(section 3.1.3) and breaches of privacy(section 3.1.5).

3.1.2 Security vulnerabilities

Newly purchased devices must never be assumed to be secure. The manufacturer may not adequately secure the software supply chain[22, 23] and vulnerabilities may be discovered further along the lifespan of the device. Cloud technology, which many IoT devices use for data processing or storage, can easily be compromised by e.g. an insider[19]. Malware can potentially be injected during production, distribution and even during the operation phase of a device.

UPnP may also cause problems. In 2013 security firm Rapid7 released a report[24] stating the security flaws of UPnP and how easy they are to exploit. UPnP offers “plug and play” within the networking domain to allow easy communication between networked devices. It is also enabled by default on millions of systems, including Microsoft Windows, Mac OS X, and many Linux distributions. Insecure router configuration may expose UPnP on the Internet, making the devices vulnerable and exposed. In 2012 Rapid7 discovered 81 million unique IP addresses that responded to UPnP discovery requests. Subsequently, UPnP simplifies network configuration but may also simplify exploitation, thus making any user an easier target for adversaries.

IoT devices are often specialized. Services that are not required to support core functions are thus unnecessarily exposed[16]. An example is having Telnet enabled[9]. These services can expose devices to additional vulnerabilities.

Devices may also be made with publicly known vulnerabilities which render them vulnerable to attacks from the time of installation[25]. In 2016, 6,435 vulnerabilities were added to the Common Vulnerabilities and Exposures (CVE) database. The CVE database contains registered vulnerabilities, categorized by vendors, products, and type. CVEs are not restricted to a particular device, and may thus also affect IoT devices. Symantec stated in the Internet Security Threat Report (ISTR) report of April 2017[17] that it only takes 2 minutes for an IoT device to be attacked. Conclusively, insecure software greatly puts the user, and their devices, at risk.

3.1.3 Data leaks

As sensors are introduced to the home, private user data is gathered and stored by such devices. It can be assumed that the vast variety of devices, from heart monitors to electrical heaters, in the home make stored data more personal and dynamic[26]. As a consequence of a device's limited resources, data is likely to be stored elsewhere e.g. "the cloud". Data leaks from the cloud are not unique to the IoT context, but must not be forgotten. Poor encryption and authentication of data transfers has shown to be a common weakness for IoT devices[20, 21]. This weakness could again lead to the security breaches that could potentially affect millions of users[27].

3.1.4 Disruption

Loss of connection is a problem that can occur with high frequency within a IoT ecosystem. A Internet Service Provider (ISP) may experience technical difficulties, an area can be affected by a power outage, and homeowners may decide to restart their router and modem at will. Whenever a device uses back-end servers, these may also experience similar connection issues. Accounting for loss of network connection, both locally within a Local Area Network (LAN) and to the outside world, is essential for any IoT system.

IoT devices can be used for home automation. Examples are devices that work as thermostats and control A/C, heating or devices that control lighting. A loss of network connection can render apps and online web portals useless for controlling such systems, which in turn can result in a disruption of service.

Connectivity is a major part of the IoT devices and operation should not be disrupted when a connection fails.

3.1.5 Breach of privacy

In 2015 Hewlett Packard Enterprise (HPE) released their ISTR report stating that

> 90 percent of devices collected at least one piece of personal information via the device, the cloud, or its mobile application[28].

Considering a regular desktop computer, information gathered about a user is limited. With IoT, information collection is not limited to only the Internet browsing behavior of users[29]. Privacy breach can potentially reveal a stunning amount of information about a user. E.g., when a user is at home, what other devices are in use or even how the user interacts with other devices on the network. Also, a privacy breach can be utilized as a stepping stone to gain further unauthorized access[25] or create device failures.

IoT devices have frequently been misused, repurposed or infected in the last few years. In April 2016 a malware was found in security cameras sold by Amazon[23]. In fact, "big names" are increasingly targeted because consumers trust them. Just imagine the consequence of having an infected thermostat during winter - frozen and burst water pipes - or even having homes heated or cooled without the owner's consent. Not all IoT devices are equipped with enough resources to use available authentication methods[26], which poses an obstacle when it comes to privacy preservation. Security and privacy problems with IoT devices can ultimately constrain the future growth of the IoT sector[22].

3.1.6 Software updates

Even if a device's software is considered secure and bug-free when shipped and sold, vulnerabilities and bug are likely to be discovered through the device's life cycle. There is no such thing as bug-free software[29], and new vulnerabilities are discovered on a regular basis. In 2015, more than 5,500 new vulnerabilities were discovered along with 54 zero-day vulnerabilities[30].

Software updates are not only made to fix bugs or fix security issues but can also improve performance and functionality. In other words, software updates may extend the life of a device.

Patching security issues is not straightforward. According to Schneier[5], even if a patch is available it is rarely applied. Users may be forced to manually download and install the relevant updates, which they may not be interested in or even able to do. An end-user rarely has the expertise to administer devices manually. The increasing amount of devices introduced through IoT, which might not even have a display or Graphic User Interface (GUI) to assist users, makes manual administration of devices a major obstacle when it comes to software updates.

Software updates are an important part of the IoT lifecycle, and should be a required capability[16]. Even if enabling updates can create new vulnerabilities, the drawbacks of not updating are far too high.

3.2 Best practice

Best practice within a field is usually established as either a written agreement or a consensus. This section will mainly present agreements on best practice or best practices submitted by internationally recognized security organizations. Best practices may not be all-inclusive but will act as an answer to the challenges of IoT from section 3.1.

There is a consensus within the IoT industry that security is the most challenging part of IoT[31]. The industry is lacking a common best practice. However, research and reports share most views on what is considered best practice[8, 22, 28, 17]. These views are mainly covered by two viewpoints in the security industry

End-users best practice includes steps that can be taken by consumers to minimize the risk of unnecessary use of unsafe or poorly made IoT devices.

Device manufacturers best practice tries to set a standard of what a producer should include as standard, and as a minimum, in their devices.

This section will sum up what the security industry consider as the best practice for securing IoT, one viewpoint at a time.

3.2.1 End-user best practice

Symantec[17] put together a list of steps that a user should do ahead of buying and while using any IoT device. Their list is targeted towards end-users, but for the most part also cover topics from section 3.1.

Prior to purchase

0. Perform an audit of IoT devices already on the network.
1. Ensure that the Wi-Fi network uses strong encryption (WPA2).
2. Research the new device's capabilities and security features.
3. Ensure that a hardware outage, like from a piece of network equipment, does not result in an insecure state of the device.

Installation

4. Use complex and unique passwords for all device accounts and Wi-Fi networks. Change the default credentials on new devices.
5. Disable unnecessary services and features.
6. Use SSH instead of Telnet where possible.
7. Modify security and privacy settings according to the user's requirements.
8. Disable remote access to the device whenever it is not needed.
9. Where possible, use wired connections instead of wireless.

Operation

10. Regularly check the manufacturer's website for firmware updates. If available, enable auto update of devices.

3.2.2 Device manufacturer best practice

An end-user should not be expected to know anything about cyber security or how to secure their IoT devices. This burden should lay on the device manufacturers. Broadband Internet Technical Advisory Group (BITAG) published a report[22] on IoT security and privacy recommendations in November 2016. The report is established as a "Uniform Agreement Report", where all BITAG members agree on the contents of the report. Members include CISCO, Sandvine, ADTRAN, AT&T, Comcast, and T-mobile. The list below has a basis in the BITAG report[22], but is not exclusively true to the report.

Software best practices

Software best practices include the whole development and operational phase of a device. This section is divided into two parts. Firstly, best practices for development is listed, followed by best practices for the operational phase.

Development phase

1. Software should not contain severe known vulnerabilities.
2. Best practices for software development should be followed.
3. Strong authentication by default. Default passwords are still the biggest security weakness for IoT devices[17, 16].

Operational phase

4. Software updates should be automated and forced

Secure different configurations

Configuration changes can improve security[9]. A device that allows for configuration changes should be expected to use different configurations for different users, based on their needs. Securing different configurations should thus be addressed.

5. Wherever users are allowed to change configurations, various configuration settings should be tested for vulnerabilities ahead of distribution. It should be assumed that users may not use the default settings.

Security and Cryptography

Proper security and the use of cryptography ensures secure communication and transfer of data as well as a measure for hardening the device itself.

6. All communication must be authenticated
7. Communication must be encrypted where possible
8. Devices that communicate with each other or with a back-end must require mutual authentication and authorization
9. Wherever IoT devices are connected, network isolation should be implemented. This isolation is recommended for any network with IoT devices, but cannot be assumed to exist due to end-users lack of knowledge.

Continuous operation

Continuous operation ensures that devices provide basic functionality without relying on other systems or networks. It is important to note here that a device should only be required to supply basic features when the network is inaccessible. Such features would include heating and air conditioning in an automated heating system.

10. Devices should provide continuous operation if the network or the Internet is inaccessible.
  - Manual controls where needed, either physical buttons or digital on the device itself.

11. Access to the Internet should not be a requirement for a device to function properly.

12. Devices should function properly without a connection to back-end servers

Privacy policy

As IoT units collect and share data about its users, and their usage of the devices, a plainly and understandably written privacy policy should be readily available to end-users. It should state what information is collected, shared and stored.

13. An easily coherent privacy policy should be accessible by the user.

3.3 Summary

Everyday items are steadily turning into connected and communicating devices. As these systems become an integrated part of our daily lives, so are any drawbacks that come with them as well[32]. IoT faces many challenges. Devices for end-users must function properly without requiring any prior knowledge of such devices. Besides, devices cannot assume that the communication systems they are connected to are properly configured for security. IoT devices are on the border of the digital and physical world. Proper security and privacy should thus be considered a requirement for the device itself[33]. Development of security practices is crucial to secure any IoT eco-system[3]. The lack of an industry-wide minimum security standard of IoT is likely to have caused a big rift in how different manufacturers implement security in their devices, if they implement any at all.

The research question *What is the current state-of-the-art within consumer-grade IoT security?* is thus concluded.

Chapter 4
Methodology and materials

This chapter contains an introduction to tools used for experiments in chapter 5, followed by an explanation of experiment setup and procedure.

4.1 Software tools

Experiments completed in chapter 5 require a setup. Mainly, the setup includes software related to the creation of a WLAN, but also the inspection of the packet flow that is captured through each experiment. This section will address such software.

Experiments require a controlled WiFi AP to monitor connections and ensure that all packets are captured. Correct routing and connectivity to the Internet is also crucial for some of the tests that will be done in chapter 5. The latter is done through dnsmasq while packet analysis is done mainly through the use of Wireshark. Exposed services are detected through the use of nmap. This section will give a more in-depth explanation of the software used.

4.1.1 hostapd v2.3

Hostapd is a daemon for creating and managing APs and authentication servers. The daemon supports IEEE 802.11 access point management, RADIUS authentication servers, and IEEE 802.1x/WPA/WPA2/EAP authenticators, and is meant to be a daemon program that functions as a back-end component controlling authentication[34]. Configuration used in the experiment is attached in Appendix A.1.

4.1.2 dnsmasq v2.72

Dnsmasq was created to be used in small embedded systems, so unneeded functions are not included. Its purpose is, simply put, to provide DHCP and DNS services to a LAN[35]. During this experiments both DNS and DHCP will be implemented through dnsmasq. DNS queries are served either locally by the dnsmasq software or forwarded to a real, recursive, DNS server. DHCP is served locally, assigning addresses to devices connected to the AP. Configuration for dnsmasq, used in the experiments, are attached in Appendix A.2

4.1.3 Wireshark v2.2.6 / Tshark 1.12.1

Wireshark is an open source GUI network packet analyzer. According to the Wireshark manual[36], it is designed to capture packets and display them in a detailed manner. It is not an intrusion detection system or a packet manipulator as it only captures packets on selected interfaces and enables detailed inspection of them. Tshark is the Command-line Interface (CLI) version of Wireshark.

During experiments, Tshark will capture packets while Wireshark will be used to analyze the packet streams. An example of Wireshark usage is inspecting packets to check for encrypted communication.

4.1.4 Nmap v6.47

Nmap is an open source security audit and network discovery tool[37]. It was designed to quickly scan large networks revealing available services and Operating System (OS) on hosts in the network.

In this thesis nmap will be used to scan all ports on the IoT devices to look for unnecessary services and vulnerable versions of such services. The command used to discover remote services and their version numbers is found in listing 4.1.

```
1 nmap -O -A -p 1-65535 <ip>
```

Listing 4.1: Nmap command to detect remote services, version numbers and information about the operating system.

4.2 Setup and procedure

Experiment setup, procedures, and control questions are explained and defined in this section.

4.2.1 Experiment setup

The setup consists of a VMware vSphere 6.5 with an individual Virtual Machine (VM), installed with a Debian 8 "Jessie", for each experiment. The VMware server has a TP-Link TL-WN722N USB wireless device connected, which is forwarded to the individual active VM. Each VM has a script(attached in Appendix A) to easily deploy and start the experiment setup. In combination, the script and the two software create an AP for IoT devices, enabling them to reach the Internet. During experiments, IoT devices are connected to the Internet through the VM so that packet capturing and inspecting can be performed with Wireshark. Nmap completes the software list by discovering remote services on the devices. An illustration of the experiment setup can be seen in figure 4.1.

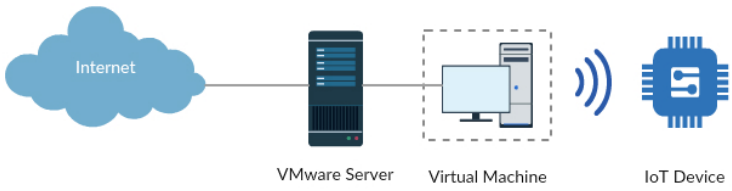

Figure 4.1: Experiment setup. IoT devices are connected wirelessly to a virtual machine run on VMware vSphere server that is connected to the Internet.

4.2.2 Experiment procedure

This subsection defines how experiments are completed throughout the thesis. Part two of this section contains control questions used to compare devices to the best practice(section 3.2.1).

Experiment procedure consists of several steps to ensure that data is captured correctly on the VM, as well as ensure that the device behaves like it would in a regular user environment.

1. Boot up a new pre-configured VM for the experiment
2. Ensure that the network is broadcasting and that packet capturing has started
3. Unbox device and read install instruction.
4. Configure device according to installation instructions.
5. Install any apps that may accompany the device. Phones can at this point be attached to the wireless network as the device.
6. Try out basic functionality.
7. Inspect configuration, if possible.
8. Run nmap to discover remote services
9. Shutdown device.
10. Shutdown packet capturing.
11. Save package dump for further analysis(Wireshark).

Best practice control questions are extracted from the best practice section (section 3.2.2) in chapter 3. These questions are to be considered as a control sequence to establish if a device is following the current best practice or not. It is not feasible to test all aspects of the best practice through only the device itself(e.g. software development practices). The list below may thus not completely reflect the best practice in section 3.2.2.

- Authentication
  - Is there a unique device first-time password?
  - Does the device require a change of password on setup?
  - Are there requirements for password complexity?
  - Can passwords be disabled?
- Automatic updates
  - Does the device allow updates?
  - Are automatic updates enabled by default?
  - Does the device automatically force updates?
  - Can automatic updates be turned off?
- Communication
  - Is communication encrypted?

  - Can configuration be updated to allow non-encrypted communication?
  - Are unnecessary services disabled?
  - What algotithms are used for encryption?
- Continuous operation
  - Does the device provide its services locally when an Internet connection is disabled?
  - Can the device function properly without any network connection?
- Privacy Policy
  - Is there a privacy policy attached to the device manual?
  - Is there a privacy policy available online?
  - Is the privacy policy easily readable for a non-technical end-user?

Chapter 5
Experiments

Chapter 5 will answer two research questions

RQ2 *What security controls are used in consumer-grade IoT devices?*
RQ3 *How does the type and amount of implemented security controls differ between consumer-grade IoT enabled devices?*

Experiments with four IoT devices will be used to answer both RQ2 and RQ3. This chapter is divided into two parts. Experiments are conducted on all four devices, before findings are summed up with a comparison of devices.

This chapter will contain experiments completed according to the procedure defined in 4.2.2. Control questions listed in section 3.2.2 will be used to answer both RQ2 and RQ3.

5.1 Philips SCD860

5.1.1 Device description

Philips Avent SCD860 is a WiFi enabled baby monitor. Essentially it is a standalone web camera that allows users to live stream video, and take snapshots, directly to a smartphone remotely. The device comes with built-in environmental monitoring and speakers. The Avent connects to an existing wireless network and is compatible with multiple APs and will try to stay attached to the best one at all times.

5.1.2 Authentication

The configuration of the baby monitor required installing an app called “uGrow Smart Monitor”. At the time of the experiment, the app version was v1.6.2. Usage of the app requires the creation of an account, in which users also accept the privacy policy when using the monitor. Privacy policy is addressed later in section 5.1.6. Upon registering the account, a password had to be specified. The complexity requirements include

- At least 8 characters
- Contains at least two of the following:
  - letters (a-z or A-Z)
  - numbers (0-9)
  - special characters (_.@$)

Password complexity requirements prevent end-users from choosing weak passwords. As there are no other passwords used in combination with the Avent, unique non-standard passwords are forced.

Connecting the Avent to the WiFi network is done through the App. A QR code is generated which contains information about the wifi-network. Decoding the QR code reveals the following content

```
1 WIFI:T:;S:Thesis-Experiment;P:picky combat;;17:12:;TOKEN
  :1879707591941830;##+
```

which translates to

T Type:WIFI
S SSID
Network type unspecified
P pass phrase
Token Most likely an identifier to connect the camera to a specified account

Contents of the QR code are not encrypted. However the QR code is only used once, not sent anywhere, and just temporarily available to the user.

5.1.3 Software updates

Software updates were enabled by default. App automatically started to download and apply firmware update for the monitor upon connecting to the Internet. Whenever a firmware update was available, a notification was shown on screen forcing users to apply firmware updates. After the update firmware version of the Avent was v171.100.171.7. The notification can be seen in figure 5.1. The figure indicates that auto-update is forced.

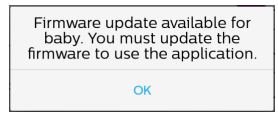

Figure 5.1: In-app notification from the uGrow app, forcing firmware updates

5.1.4 Communication

An observation made when using the included Android App is that the camera only sends video stream when the app is connected and is viewing or monitoring the device. Otherwise, it only transmits "keep-alive" messages to the video server. All video traffic goes through Philips' video servers. A loss of Internet connection prevents the camera from successfully communicating with the server, and most services are terminated until the Internet connection is reestablished. In addition to being a camera, the Avent also has a nightlight and lullaby features. These are the only two functions that remain enabled whenever the video server is unreachable.

```
Starting Nmap 6.47 ( http://nmap.org ) at 2017-06-10 22:35 CEST
WARNING: RST from 192.168.0.10 port 80 -- is this port really open?
WARNING: RST from 192.168.0.10 port 80 -- is this port really open?
WARNING: RST from 192.168.0.10 port 80 -- is this port really open?
WARNING: RST from 192.168.0.10 port 80 -- is this port really open?
WARNING: RST from 192.168.0.10 port 80 -- is this port really open?
WARNING: RST from 192.168.0.10 port 80 -- is this port really open?
Nmap scan report for 192.168.0.10
Host is up (0.0031s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE    VERSION
80/tcp    open  http?
56789/tcp open  tcpwrapped
MAC Address: 1C:5A:6B:AE:8B:2F (Philips Electronics Nederland BV)
Device type: general purpose
Running: Linux 2.4.X
OS CPE: cpe:/o:linux:linux_kernel:2.4.21
OS details: Linux 2.4.21
```

Figure 5.2: Nmap scan of the Philips Avent. Results show two pontentially open ports, applications using each ports, and information about the operating system.

Completing an nmap scan on the Avent revealed two potentially open ports. The scan results can be seen in figure 5.2. Results included

80/tcp http?
56789/tcp tcpwrapped

but did not provide much information. The application *tcpwrapped* is often associated with a firewall or wrapper that prohibits access to the underlying application. As the scan results reflect, port 80 *http?* is possibly not actually open. Browsing the ports resulted in a timeout. According to nmap, the Avent uses Linux kernel 2.4.21. This version is vulnerable to privilege escalation through CVE-2009-2692[38].

Communication between the Avent, Philips back-end servers and the phone app is done through encrypted communication. Both TLS 1.2 and TLS 1.0 are used with `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` and `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA` respectively. The latter is used when the device is registering with the Philips back-end, while TLS 1.2 and `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` is used for establishing connections with servers and apps for STUN/TURN services, XMPP/SIP, and when directly streaming video. STUN/-TURN servers are used for NAT traversal, while XMPP/SIP are two protocols for real-time communication.According to RFC7525[39], `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384` is considered one out of four recommended cipher suites for TLS 1.2.

Encrypted communication to Philips' and related servers use one out of two certificate chains

```
Verisign Class 3 Public Primary Certificati (SHA1 with RSA)
└─Symantec Class 3 Secure Server CA - G4 (SHA256 with RSA)
  ├─www.ecdinterface.philips.com - HSDP Device Cloud (SHA256 with
  │ RSA)
  ├─dcp.cpp.philips.com - HSDP Device Cloud (SHA256 with RSA)
  └─smartbabymonitor.ugrow.philips.com - Consumer Lifestyle (SHA256
    with RSA)


Starfield Class 2 Certification Au (SHA1 with RSA)
└─Starfield Root Certificate Authority (SHA256 with RSA)
  └─Starfield Secure Certificate Authority G2 (SHA256 with RSA)
    └─logs-01.loggy.com (SHA256 with RSA)
```

The first certification chain, with a base in Verisign Certification Authority, is used for all communication with the Philips back-end. The latter is only used with *loggy.com*, which is a cloud logging service. In both chains, SHA1 with RSA is used for to sign the root certificate. SHA1 is deprecated and should, according to Symantec[1], be replaced by SHA-2 on all certificates that expire after December 31, 2015. A transition has thus not been made for Starfield or Verisign root certificates.

In addition to communcation with Philips back-end, the Avent broadcasts UPnP messages. UPnP messages are sent through the use of Simple Service Discovery Protocol (SSDP). One such message can be seen in figure 5.3 where a broadcast message is sent to the broadcast IP 239.255.255.250.

Figure 5.3: Screenshot from Wireshark showing an UPnP packet that has been broadcasted by the Avent, located at 192.168.0.10.

5.1.5 Continous operation

The Avent has an RGB status Light Emitting Diode (LED) which indicates the connectivity of the camera. In normal operation it stays turned on with a green color, symbolizing that everything works fine. When there is no wireless network, the LED turns into an orange color, stating that the camera is not connected. Disconnecting the Avent from the Internet prevents the camera from receiving connection information from the Philips back-end servers, thus preventing a connection being established between the Avent and the Smartphone app. In turn, the lack of an Internet connection prevents the transfer of video from the Avent to the app. Conclusively, the Avent does not have any modes for continuous operation when it comes to baby monitoring, without network and Internet connection.

[1]https://www.symantec.com/theme/sha2-transition - Transition from SHA-1 to SHA-2 SSL

5.1.6 Privacy policy

The Avent comes with a manual containing general information about the camera. A separate section of the manual is dedicated to privacy. It states that Philips strongly believes in protecting the privacy of the personally identifiable information that the user shares through the app, and that information about the use of such data can be found in the privacy notice in the app. Additionally, the privacy policy can be accessed through the Philips website[40]. The policy states that video and audio streams are not stored, but are processed through the cloud. Personally identifiable information, like name, email, date of birth, and gender, are saved as long as the account is active.

5.1.7 Summary

Through testing the Avent, Philips has proven to create a product that more or less complies with the best practice control questions. The failed tests of continuous operation are partially understandable as a baby monitor that is unable to transmit to a receiver is not fulfilling its intended task. The Philips Avent should, however, work without an Internet connection. Enabled unnecessary services found through nmap should also be disabled. Turning off, or adjusting provided services are not possible through device configuration. Test results are summed up in table 5.1.

Table 5.1: Best practice control questions reflecting Philips Avents compliance with the best practice presented in chapter 3

Best practice control questions		**Philips Avent SCD860**
Authentication	Unique first-time-password?	-
	Require change of password on setup?	✓
	Requirements for password complexity?	✓
Software updates	Does the device allow updates?	✓
	Are automatic updates enabled by default?	✓
	Does the device force automatic updates?	✓
	Can automatic updates be turned off?	✓
Communication	Is communication encrypted?	✓
	Does the configuration allow unencrypted communication?	✗
	Are unnecessary services disabled?	✗
	What algorithms are used for encryption?	AES256
Continuous operation	Does the device work without an Internet connection?	✗
	Does the device work properly without any network connection?	✗
Privacy Policy	Included with the device?	✗
	Available online?	✓
	Easily readable?	✓

5.2 Motorola MBP845Connect

5.2.1 Device description

The Motorola Connect is a baby monitor that shares several features with the Avent from the previous experiment. One major difference is that the Connect comes with a "parent unit" that offers a control device to control the baby monitor locally. The Connect also provides "Internet Viewing" through a downloadable app known as Hubble.

A quick start guide is included with the Connect. It contains simple instructions on how to setup the system, including how to perform basic operations through the use of the physical buttons on the device. Connecting the device to the Internet requires the use of the Hubble app.

5.2.2 Authentication

Turning the parent unit and baby monitor on resulted in an instant connection between the two. No setup or button pushes necessary. Connecting the baby monitor to the app, and thereby the Internet, required a little more effort but was simple enough as the discovery of nearby devices was integrated into the app. The Connect created a WLAN that the app connected to, in which discovery of nearby WLANs make selecting and finishing the connection work smoothly.

Using "Hubble" requires the creation of an account. There are several easily understandable requirements for selection of username, email address, and password. The password complexity is quite extensive as it includes requirements of

- at least one number
- at least one uppercase letter
- at least eight characters
- at most thirty characters
- cannot be the same as the username

Creating a user account also requires the acceptance of Terms of Service, which include a reference to the privacy policy of the Hubble. A great password complexity ensures that users are more likely to choose a good password.

5.2.3 Software updates

In the current app version v4.5.6(474), there were not many settings to configure in the app, or on the parent unit for that matter. Firmware version of the Connect was listed in the app as a "button" for updating the firmware. Current firmware version is v01.19.64. The configuration of monitor name and a button to check for firmware updates were the only options other than viewing information about the system.

Through accessing hidden HTML pages on the Motorola, a firmware update page becomes available. Uploading new firmware requires a manual download of new firmware and manually upload it to the camera through a page called `http://<ip>:8080/fwupgrade.html`. Uploading new firmware requires HTTP, without any encryption.

5.2.4 Communication

Whenever the Motorola Camera communicates to the Motorola backend API server or Hubble resources, this is done through the use of TLS 1.2 or TLS 1.1 and the `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` or `TLS_RSA_WITH_AES_256_CBC_SHA` cipher suites respectively. The latter is, according to RFC7525[39], deprecated as negotiation of cipher suites based on RSA does not support forward secrecy. The former, `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256` cipher suit, is, however, one out of four recommended cipher suites in the RFC7525.

Communication is done with one out of two certification chains

```
Go Daddy Class 2 Certification Aut (SHA1 with RSA)
└─ Go Daddy Root Certificate Authority - G2 (SHA256 with RSA)
   └─ Go Daddy Secure Certificate Authority - G2 (SHA256 with RSA)
      └─ *.hubble.in (SHA256 with RSA)
```

```
Baltimore CyberTrust Root (SHA256 with RSA)
└─ DigiCert Baltimore CA-2 G2 (SHA256 with RSA)
   └─ *.s3.amazonaws.com (SHA256 with RSA)
```

where one covers all communication with `*.hubble.in` domains, and the other handles communication with Amazon AWS servers. Go Daddy root certificate is

signed with an SHA1 based algorithm, which should have been replaced by an SHA2 based certificate.

Video transferred between the camera and the app, and between the camera, back-end servers and camera(remote viewing) is encrypted in all phases of transfer. Video streaming from the camera to the app on the local network is encrypted with a session key that is requested and sent in plaintext. HTTP request and response containing the plaintext session key request and actual session key can be seen in Appendix B.1.1 and B.1.2.

Just like when experimenting with the Avent, it can be observed that the Connect only sends data through the network when requested by the app. Transfer to the parent unit is not included in this as intercepting packets sent on the proprietary connection, between the parent unit and the monitor, is not part of the scope of this thesis.

Nmap scan of the Connect revealed six open ports. Scan can be seen in figure 5.4. Results from the scan included

80/tcp nuvoton
: Returns a *404 Not Found* page.

6667/tcp GM Streaming Server httpd
: Grain Media Streaming server - IP camera streaming software

8080/tcp Busybox https 1.13
: A Linux tool that provides several UNIX tools within a single executable.

Researching each of the applications revealed that Busybox has a total of seven CVE registered[41]. Version 1.13 is vulnerable to at least five out of the listed seven, ranging from a severity score of 2.5 to 7.5. Four out of the CVEs are scored at 5 or greater severity score. Vulnerabilities include code execution, Denial of Service (DOS), overflow and bypassing. The other applications found through the nmap scan did not appear to have any registered CVEs.

According to the nmap scan, the camera uses a Linux 2.6 kernel.

Trying to open an Real Time Streaming Protocol (RTSP) stream to port 6667 through a video player required a password and username. RTSP is a protocol for controlling video streaming servers. Similar Motorola baby monitors have been hacked[42] and it has been discovered that

Username: user
Password: pass

has been known to be the default username and password. Trying the default username and password resulted in a successful connection to the Motorola MBP845 video stream. It was not possible to capture and decode the RTSP video stream.

```
Starting Nmap 6.47 ( http://nmap.org ) at 2017-06-11 12:49 CEST
Nmap scan report for 192.168.0.52
Host is up (0.0061s latency).
Not shown: 65529 closed ports
PORT      STATE SERVICE    VERSION
80/tcp    open  tcpwrapped
|_http-methods: No Allow or Public header in OPTIONS response (status code 400)
|_http-title: Site doesn't have a title (text/plain).
6667/tcp  open  http       GM Streaming Server httpd
|_http-methods: No Allow or Public header in OPTIONS response (status code 501)
|_http-title: Site doesn't have a title (text/html).
|_irc-info: Unable to open connection
8080/tcp  open  http       BusyBox httpd 1.13
|_http-methods: No Allow or Public header in OPTIONS response (status code 501)
|_http-title: 404 Not Found
51108/tcp open  tcpwrapped
60000/tcp open  unknown
60001/tcp open  unknown

...

MAC Address: 00:0A:E2:4D:4A:E6 (Binatone Electronics International)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.16 - 2.6.35 (embedded)
Network Distance: 1 hop
Service Info: OS: Linux; Device: webcam; CPE: cpe:/o:linux:linux_kernel
```

Figure 5.4: Nmap scan of the Motorola Connect. Results show six open ports, the applications, the application version using the ports, in addition to showing OS related information. For easy readability, information regarding submitting fingerprint for an unrecoginzed service/version has been removed and replaced by '...'

In addition to the ports mentioned above, the nmap scan also found three other open ports

51108/tcp tcpwrapped
60000/tcp unknown
60001/tcp unknown

None of the ports have a description of service attached, and it is thus not possible to identify services that use the three ports. Browsing the ports resulted in a timeout.

The Motorola's settings can be adjusted through non-disclosed HTTP pages on the device itself. Several groups have tried to hack and exploit Motorola's ip cameras[42][2,3] and found that the firmware used is quite similar. Accessing the page `http://<ip>/cgi/jpg/image.cgi` returns a still image from the Motorola camera. Both the request and the returned image are sent unencrypted through HTTP. The image, extracted from the captured packets, can be seen in figure 5.5.

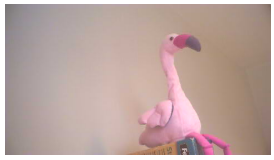

Figure 5.5: Captured still image from the Motorola MBP845. The image was downloaded from Wireshark and was captured with packet capturing.

The Motorola does not send UPnP packets.

5.2.5 Continuous operation

Transferring data from the monitor to the parent unit is done through a point to point wireless connection, while the connection between the app and the monitor is made directly between the two. This enables the monitor to function as intended without any network. However, remote viewing is not possible with the lack of network connection or Internet connection.

5.2.6 Privacy policy

Creating a user requires the acceptance of Terms of Service, which include a reference to the privacy policy of the Hubble. Even though the privacy policy is mentioned, links are not clickable, making reading the policy a hassle. Following

[2]http://blok.tiyun.de/2015/view-your-hubble-camera-stream-whereever-you-want - Instructions on how to view camerastream without Hubble
[3]http://atom0s.com/forums/viewtopic.php?t=45 - Hacking of Motorola FOCUS66 (Forum)

the link redirects customers to another web page that does not include the policy, but the policy is reachable through another set of links. The wording of the policy is simple enough, but the effort it takes to reach it makes it qualify as not easily available.

5.2.7 Summary

Motorola has, through the creation of the MBP845 Connect, created a device that only partially complies with the best practice control questions. Encrypted communication is used, but the occurrence of a standard password and plaintext HTTP traffic makes the MBP845 vulnerable. The lack of a simple firmware update method enabled unnecessary services and hard to find privacy policy sets this device quite far apart from the Philips Avent. The only feature which Motorola brings to the table is the ability for the device to work without a network connection, due to its parent control unit. Complete test results are summed up in table 5.2.

Table 5.2: Best practice control questions reflecting Motorola MBP845's compliance with the best practice presented in chapter 3

Best practice control questions		**Motorola MBP845**
Authentication	Unique first-time-password?	✗
	Require change of password on setup?	✓
	Requirements for password complexity?	✓
Software updates	Does the device allow updates?	✓
	Are automatic updates enabled by default?	✗
	Does the device force automatic updates?	✗
	Can automatic updates be turned off?	-
Communication	Is communication encrypted?	(✓)
	Does the configuration allow unencrypted communication?	✓
	Are unnecessary services disabled?	✗
	What algorithms are used for encryption?	AES128/256
Continuous operation	Does the device work without an Internet connection?	✓
	Does the device work properly without any network connection?	✓
Privacy Policy	Included with the device?	✗
	Available online?	(✓)
	Easily readable?	✗

5.3 Adax NP08WIFIWH

5.3.1 Device description

Adax Neo is an electric radiator with a programmable heating schedule. The radiator offers WiFi to enable user-specified heating programs. Basically, the Neo is a radiator with extensive features that are further enhanced by the WiFi capability. The manual is structured as a regular electric radiator manual would be, with little information about how to use the smart features. WiFi is only required to sync user specified heating programs, synchronize time and remote control. All other features can be operated by physical buttons located on the device itself.

The heater comes with a simple two-paged manual explaining the buttons and features of the apparatus. “Smart Configuration settings” is only briefly mentioned. To enable smart features on the Neo, the accompanying app must be installed on a smartphone. At the time of the experiment, the app was at version v1.35.

5.3.2 Authentication

The app requires users to sign up. Password complexity is limited and only requires the password to contain at least six characters. There are no character combination requirements. Passwords like “123456” are defined as a valid and pass the password complexity requirement. The app does not allow for in-app password change.

Connecting to the heater required a physical reset of the Neo as well as connecting to the heater’s own hotspot to transfer necessary information about the local WiFi network. The procedure to accomplish this is simple enough, pressing three buttons simultaneously for 5 seconds, but required several retries to complete successfully. The Neo’s hotspot WiFi password was a standard password, “magicpass123”, which is most likely hard-coded into all Neo radiator.

5.3.3 Software updates

Upon establishing a successful connection between the Neo and the Internet, the app showed a notification stating that a new firmware update was available. The notification can be seen in figure 5.6 There was no automatic or forced update, but the button to initiate the update was readily available in the app. After the firmware update, the version of the Neo was v1.0.0.23. The app sends a firmware update command through Hypertext Transfer Protocol (HTTP) unencrypted, as seen in listing B.2.1, but this is the only real request that is not encrypted.

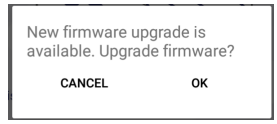

Figure 5.6: In-app notification from the ADAX WiFi app, informing users about an available firmware update.

5.3.4 Communication

Adax back-end uses TLS 1.2 with either `TLS_RSA_WITH_AES_256_CBC_SHA256` or `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA` to communicate with the heater and app, respectively. `TLS_RSA_WITH_AES_256_CBC_SHA256` is deprecated due to lack of forward secrecy while the `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA` is one of four recommended ciper suits in RFC7525[39]. Packets are sent using HTTP over TLS.

During communication, one out of two certificates are used

```
Baltimore CyberTrust Root (SHA256 with RSA)
└─ Microsoft IT SSL SHA2 (SHA256 with RSA)
   └─ *.azurewebsites.net (SHA256 with RSA)
```

```
sheater.adax.lt (SHA1 with RSA)
```

Strangely both certificate trees are used by the same host `sheater.adax.lt`. The domain resolves to azure websites and a service called `cloudapp.net`, which is a domain used to expose services through Azure. The last certificate is a self-signed certificate using the deprecated SHA1. The two certificates are consistently used, where the first is used between the heater and back-end and the latter between the app and the back-end.

In addition to communication to a back-end, the heater and app also communicate directly with each other. This communication is encrypted but appears to be using a symmetrical cipher with a shared secret as there are no TLS handshake or plaintext communication.

```
Starting Nmap 6.47 ( http://nmap.org ) at 2017-06-11 16:39 CEST
Nmap scan report for 192.168.0.77
Host is up (0.0021s latency).
Not shown: 65455 closed ports, 79 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http?
MAC Address: 5C:CF:7F:EA:6F:BA (Unknown)
No exact OS matches for host (If you know what OS is running on it, see http://nmap.org/submit/ ).
```

Figure 5.7: Nmap scan of the Adax Neo. Results show one open port, but nmap is unable to detect the application using the open port.

Performing an nmap scan on the heater reveals only port 80 as open. However, nmap was unable to recognize the application using the port. Results of the nmap scan can be seen in figure 5.7. Browsing the port resulted in a timeout.

The Adax Neo heater does not send UPnP packets.

5.3.5 Continuous operation

When disconnecting the heater from the Internet, the heater continued its operation. Heater remained reachable through the app, but with only limited functionality. Test of connectivity, making the Neo's led blink, worked like normal without the Internet. However, adjusting the target temperature and heating programs were not possible through local network alone(no Internet).

5.3.6 Privacy policy

There is no reference to a privacy policy in the manual, in the app or on the device itself.

5.3.7 Summary

Experiments completed with the Adax Neo has proven that it only partially comply the proposed best practice defined in chapter 3. Lack of privacy policy, automatic updates, and password complexity are among the most prominent issues.

Good routines and a password generator would in effect reduce the vulnerabilities that these issues can cause.

Table 5.3: Best practice control questions reflecting Adax NP08WIFIWH compliance with the best practice presented in chapter 3

Best practice control questions		**Adax Neo**
Authentication	Unique first-time-password?	✗
	Require change of password on setup?	✓
	Requirements for password complexity?	✗
Software updates	Does the device allow updates?	✓
	Are automatic updates enabled by default?	✗
	Does the device force automatic updates?	✗
	Can automatic updates be turned off?	-
Communication	Is communication encrypted?	✓
	Does the configuration allow unencrypted communication?	✗
	Are unnecessary services disabled?	✗
	What algorithms are used for encryption?	AES256
Continuous operation	Does the device work without an Internet connection?	✓
	Does the device work properly without any network connection?	✓
Privacy Policy	Included with the device?	✗
	Available online?	✗
	Easily readable?	-

5.4 Mill AV600WIFI

5.4.1 Device description

Mill Glass WiFi is another electric radiator with WiFi capabilities. As with the Neo, the Glass radiator is mainly an "offline" heater where its WiFi capability only extends its features. The electric heater comes with installation instructions and a user manual. It explains how to operate the heater with the physical buttons and include a section where the use of the WiFi capability is explained.

5.4.2 Authentication

An app allows users to control the heater's extensive features remotely. At the time of the experiment, the app version was v2.0.9. Creating an account to use the app requires that users accept the privacy policy and terms. To create an account, the email must be verified before the account can be created. This is done through a verification pin code that is sent to the email. Upon entering the pin, account creation also requires a password to be specified. As with the Neo, "123456" is considered a valid password, which emphasizes that the system lack password complexity requirement besides a minimum length of 6 characters.

5.4.3 Software updates

There is no option to update software in the app.

5.4.4 Communication

Running an nmap scan on the Mill heater revealed one open port. The result from the port scan can be seen in figure 5.8. The nmap scan returned `25001/tcp`

```
Starting Nmap 6.47 ( http://nmap.org ) at 2017-06-12 20:53 CEST
Nmap scan report for 192.168.0.57
Host is up (0.0031s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE       VERSION
25001/tcp open  icl-twobase2?
MAC Address: 00:0E:C6:08:14:04 (Asix Electronics)
No exact OS matches for host (If you know what OS is running on it, see http:
```

Figure 5.8: Nmap scan of the Mill Glass. Results show one open port, and an unrecognizable operating system.

as open. The service registered by IANA[4] is the same as found by nmap. Neither IANA or the nmap can determine any specific service that uses the port. As a result, no unnecessary services are found present on the device.

Analyzing captured packets reveals that communication between the "Millheat"-app and the "Millheat-cloud" is sent as plaintext, without any encryption. Packets are sent through `eurouter.ablecloud.cn`, which belongs to a Chinese IoT service provider. The IP resolves to Germany. Messages intercepted include the heaters assigned name, room name, and current temperature. Sample messages sent between the "Millheat-cloud" and the app can be seen in Appendix B.3.

The Mill heater does not send UPnP packets.

5.4.5 Continuous operation

Enabling WiFi on the radiator disables the manual controls, but WiFi can easily be turned on or off with a single physical button located on the heaters control panel.

Whenever the heater loses connection to "Millheat-cloud" all WiFi features are deactivated and only manual controls are available. This goes for either loss of an Internet connection and loss of network connection in general.

5.4.6 Privacy policy

The electric heater comes with installation instructions and a user manual. There is no privacy policy in the manual, only references to the manufacturer's webpage to obtain more information about the product. To create a user account in the app, accepting the privacy policy is required.

5.4.7 Summary

Testing the Mill electric radiator has shown several similarities with IoT challenges mentioned in chapter 3. Password complexity requirement, automatic updates, and encryption has proven to be the major deviation, as shown in table 5.5, from the best practice control questions. Reducing risks can only be achieved

[4]IANA Service Name and Transport Protocol Port Number Registry - https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?s=&page=126

by disabling WiFi, which in turn would disable any features and characteristics associated with IoT devices. Complete test results are summed up in table 5.4

Table 5.4: Best practice control questions reflecting Mill AV600WIFI compliance with the best practice presented in chapter 3

Best practice control questions		**Mill Heater**
Authentication	Unique first-time-password?	-
	Require change of password on setup?	✓
	Requirements for password complexity?	✗
Software updates	Does the device allow updates?	✗
	Are automatic updates enabled by default?	✗
	Does the device force automatic updates?	✗
	Can automatic updates be turned off?	-
Communication	Is communication encrypted?	✗
	Does the configuration allow unencrypted communication?	✓
	Are unnecessary services disabled?	✓
	What algorithms are used for encryption?	-
Continuous operation	Does the device work without an Internet connection?	✓
	Does the device work properly without any network connection?	✓
Privacy Policy	Included with the device?	✗
	Available online?	✓
	Easily readable?	✗

5.5 Experiment results and device comparison

This chapter has covered experiment on four devices, and compared them to the industry best practice presented in section 3.2.2 through a set of control questions found in section 4.2.2. Discoveries from experiments has been put into table 5.5 where all devices have been added to provide a side-by-side comparison. Implemented security controls, as defined in the best practice from chapter 3, has been put to the test through experiments and the outcome of each experiment has been concluded by a checkmark(✓) or xmark(✗) in table 5.5 to indicate compliance with the best practice.

When it comes to security controls used in IoT, experiments showed a variety of implemented controls. Users are for the most part forced to change or create a new password on device setup. This ensures that end-users do not keep using default passwords. However, password complexity is lacking in half of the tested devices. Fixing software bugs and vulnerabilities are covered by the possibility to update a devices software, and encryption of communication is often implemented as a standard. This satisfies RQ2, thus making the research question *What security controls are used in consumer-grade IoT devices?* answered.

Side-by-side comparison made available through table 5.5 also satisfies RQ3 as it clearly indicates differences between the devices. Within each group of devices, there can be seen a variation of implemented security controls. The table indicates that the baby monitors have a higher security level, but somewhat lack the ability to provide continuous operation.

Electrical heaters clearly lack security controls. Mill and Adax are less experienced in producing IoT devices than e.g. Philips, which is the best device among those tested. Lack of encryption, missing privacy policies and lack of password complexity makes the electrical heaters poorly secured.

Philips and Motorola are both manufacturers of IoT devices, and distribute their electronics world-wide. There is still a big rift between the two, where the Motorola baby monitor is worse off than the Philips. Default passwords, only partially encrypted communication, and a poor privacy policy sets the two devices apart. Old and reused firmware in the Motorola baby monitor is another drawback that is not covered by the best practice control questions.

In general, neither heaters nor baby monitors provide easily accessible and

readable privacy policies as none of the manufacturers included it with the device. Unnecessary services are exposed through open ports and updates are for the most part a manual operation if it is available at all.

Table 5.5 shows that none of the tested devices completely comply with the best practice. In sum, this concludes RQ3: *How does the type and amount of implemented security controls differ between consumer-grade IoT enabled devices?*

Table 5.5: Best practice control questions - Comparison of tested devices

Best practice control questions		**Philips Avent SCD860**	**Motorola MBP845**	**Adax Heater NP08WIFIWH**	**Mill Heater AV600WIFI**
Authentication	Unique first-time-password?	-	✗	✗	-
	Require change of password on setup?	✓	✓	✓	✓
	Requirements for password complexity?	✓	✓	✗	✗
Software updates	Does the device allow updates?	✓	✓	✓	✗
	Are automatic updates enabled by default?	✓	✗	✗	✗
	Does the device force automatic updates?	✓	✗	✗	✗
	Can automatic updates be turned off?	✓	-	-	-
Communication	Is communication encrypted?	✓	(✓)	✓	✗
	Does the configuration allow unencrypted communication?	✗	✓	✗	✓
	Are unnecessary services disabled?	✗	✗	✗	✓
	What algorithms are used for encryption?	AES256	AES128/256	AES256	-
Continuous operation	Does the device work without an Internet connection?	✗	✓	✓	✓
	Does the device work properly without any network connection?	✗	✓	✓	✓
Privacy Policy	Included with the device?	✗	✗	✗	✗
	Available online?	✓	(✓)	✗	✓
	Easily readable?	✓	✗	-	✗

Chapter 6
Securing Internet of Things

The previous chapter, chapter 5, put the devices themselves to the test. This chapter will concern itself by answering RQ4: *Can a consumer affect the security of their IoT device through updates or configuration changes?*, which is consumer/end-user oriented opposed to the previous manufacturer oriented research questions.

Chapter 3 defined two proposed best practices. One of them was concerning end-users effort to secure their devices. This proposed best practice will be relevant to this chapter.

6.1 Following the best practice

Answering RQ4 can be split into three parts, which complies with the best practice from section 3.2.1,

Configuration Securing device through configuration
Network Ensure a secure network setup
Knowledge Technical expertise about a system or device

As chapter 3 states, one cannot assume that a user has any technical knowledge about relevant subjects. This renders the last part useless for the purpose of answering RQ4. Creating a secure network setup, analyzing devices, and performing device audit also falls into this category.

Any steps to secure a device during installation not explicitly explained in the manual will be natural to exclude from expected steps for end-users to complete.

The configuration of devices, including unique and complex passwords, is the simplest steps a user can take towards improving device security. If used in combination with frequent software updates, these measures will for the most part cover configuration changes that can be implemented by an inexperienced end-user.

6.2 Summary

As the previous section briefly explained, users without technical knowledge are unlikely to secure their devices further than what is explained in a device manual. A consumer can usually affect the security of an IoT device, but due to the complexity, it is unlikely. RQ4: *Can a consumer affect the security of their IoT device through updates or configuration changes?* is thus concluded.

Chapter 7
Conclusion and Further Work

This thesis has presented challenges that the IoT industry face today, with an emphasis on consumer related devices. Through a literary study, the lack of a universal and industry-wide standard for IoT security has become apparent. Four consumer IoT devices have been tested against a proposed best practice.

Experiments conducted in this thesis reveal that devices lack different basic security measures. These include lack of encryption, use of default passwords, enabled and exposed unnecessary services, and lack of privacy policy. The tested devices can be divided into two categories, based on their features: baby monitors and electric heaters. Studying the results grouped by category shows that baby monitors, in general, provide a higher security level, but somewhat lack the support for continuous operation and disabling of unnecessary services. Electric heaters, on the other hand, lack password complexity. Both category has devices that are considerably worse than others. Neither device type, or individual device, get a full score. In conclusion, neither of the two categories get a good score in the experiments due to lack of necessary security features.

Summing up all the previous chapters leaves us with two options

1. Demand and expect more from manufacturers of IoT devices
2. Educate consumers

The first option can be achieved through standardization of IoT requirements, regardless of what industry it comes from(i.e. baby monitors vs. electric heaters). BITAG has done this for all its members, but this only covers a tiny portion of po-

tential IoT enabled device manufacturers. Establishing global device certifications can be a way to increase the likeliness, and quality, of manufacturers implementing security controls on their devices by default. This would also simplify the user's process of selecting secure devices over insecure(and thus uncertified) devices. A system for device certification is not covered by this thesis and must thus be considered a suggestion for further work on this subject.

Option 2. involves changing how users understand their devices and the device's capabilities. This is unlikely to happen as devices become more intertwined with everyday objects, each other, and the Internet itself. Complexity will increase and raises the knowledge barrier further. Education of consumers is by itself, not a valid approach for closing the IoT security gap.

Further work should focus on the manufacturers perspective of securing IoT devices. Potential cost reduction and good reputation are likely to make companies prioritize cyber security concerns on the same line as functionality concerns.

References

[1] D. Giusto, I. Antonio, G. Morabito, and L. Atzori, eds., *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer, 2010. ISBN: 978-1-4419-1673-0.

[2] K. Ashton, "That 'internet of things' thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009. accessed on 05-may-2017.

[3] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, and A. Bouabdallah, "A systemic approach for iot security," in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, pp. 351–355, IEEE, 2013. accessed on 04-may-2017.

[4] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the ip-based internet of things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, 2011. accessed on 14-may-2017.

[5] B. Schneier, "The Internet of Things Is Wildly Insecure—And Often Unpatchable." URL: https://www.schneier.com/essays/archives/2014/01/the_internet_of_thin.html, January 2014. accessed on 03-may-2017.

[6] D. Osada and E. Gambart de Lignières, "Home automation and connected cars: a source of services and tools for insurers - Blog Sopra Steria." URL: http://blog.soprasteria.com/insurance-home-automation/, February 2017. accessed on 12-feb-2017.

[7] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: a review," in *Computer Science and Electronics Engineering (ICCSEE), 2012 international conference on*, vol. 3, pp. 648–651, IEEE, 2012. accessed on 15-jun-2017.

[8] AT&T, "The CEO's Guide to Data Security." URL: https://www.business.att.com/cybersecurity/docs/vol5-datasecurity.pdf, 2017. accessed on 02-june-2017.

[9] E. Skoudis, "Internet of things: It's not just the devices! | rsa conference." URL: https://www.rsaconference.com/blogs/internet-of-things-its-not-just-the-devices, October 2016. accessed on 26-may-2017.

[10] H. Kopetz, "Internet of things," in *Real-time systems*, pp. 307–323, Springer, 2011. ISBN: 978-1-4419-8236-0.

[11] A. Meola, "What is the Internet of Things? Definition, Industries & Companies." URL: http://www.businessinsider.com/what-is-the-internet-of-things-definition-2016-8?IR=T, December 2016. accessed on 26-may-2017.

[12] F. Xia, L. T. Yang, L. Wang, and A. Vinel, "Internet of things," *International Journal of Communication Systems*, vol. 25, no. 9, p. 1101, 2012. accessed on 26-may-2017.

[13] Oxford Dictionaries, "Definition of internet of things in english." URL: https://en.oxforddictionaries.com/definition/internet_of_things, 2017. accessed on 22-may-2017.

[14] D. Miessler, "IoT Attack Surface Mapping." URL: https://www.owasp.org/images/3/36/IoTTestingMethodology.pdf at DEFCON 23, 2015. accessed on 22-may-2017.

[15] W. Stallings, *Cryptography and Network Security: principles and practices, 5th ed.* Pearson Education, Inc., 2006. ISBN: 978-0-13-609704-4.

[16] AT&T, "The CEO's Guide to Securing the Internet of Things." URL: https://www.business.att.com/cybersecurity/docs/exploringiotsecurity.pdf, 2016. accessed on 02-june-2017.

[17] Symantec, "ISTR - Internet Security Threat Report - Volume 22," April 2017. accessed 02-may-2017.

[18] A. Ardiri, "Is it possible to secure micro-controllers used within IoT?." URL: http://evothings.com/is-it-possible-to-secure-micro-controllers-used-within-iot/, August 2014. accessed on 04-may-2017.

[19] M. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of internet of things (iot)," *International Journal of Computer Applications*, vol. 111, no. 7, 2015. accessed on 06-may-2017.

[20] L. Franceschi-Bicchierai, "Internet-Connected Fisher Price Teddy Bear Left Kids' Identities Exposed." URL: https://motherboard.vice.com/en_us/article/internet-connected-fisher-price-teddy-bear-left-kids-identities-exposed, February 2016. accessed on 03-may-2017.

[21] S. Gibbs, "Hackers can hijack Wi-Fi Hello Barbie to spy on your children." URL: https://www.theguardian.com/technology/2015/nov/26/hackers-can-hijack-wi-fi-hello-barbie-to-spy-on-your-children?CMP=twt_a-technology_b-gdntech, November 2015. accessed on 03-may-2017.

[22] BITAG - Broadband Internet Technical Advisory Group, "Internet of Things (IoT) Security and Privacy Recommendations." URL: https://www.bitag.org/documents/BITAG_Report_-_Internet_of_Things_(IoT)_Security_and_Privacy_Recommendations.pdf, November 2016. accessed: 01-may-2017.

[23] K. Townsend, "Malware Found in IoT Cameras Sold by Amazon." URL: http://www.securityweek.com/malware-found-iot-cameras-sold-amazon, April 2016. accessed on 04-may-2017.

[24] M. HD, "Security Flaws in Universal Plug and Play - Unplay. Don't Play.," January 2013. accessed 16-may-2017.

[25] D. Palmer, "The first big internet of things security breach is just around the corner." URL: http://www.zdnet.com/article/the-first-big-internet-of-things-security-breach-is-just-around-the-corner/, July 2016. accessed on 02-jun-2017.

[26] Z.-K. Zhang, M. C. Y. Cho, and S. Shieh, "Emerging security threats and countermeasures in iot," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pp. 1–6, ACM, 2015. accessed on 04-may-2017.

[27] L. Franceschi-Bicchierai, "Hacked Toymaker VTech Admits Breach Actually Hit 6.3 Million Children." URL: https://motherboard.vice.com/en_us/article/hacked-toymaker-vtech-admits-breach-actually-hit-63-million-children, December 2015. accessed on 03-may-2017.

[28] Hewlett Packard Enterprise, "Internet of things research study." URL: http://h20195.www2.hpe.com/V4/getpdf.aspx/4aa5-4759enw, November 2015. accessed: 02-may-2017.

[29] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "Iot security: ongoing challenges and research opportunities," in *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pp. 230–234, IEEE, 2014. accessed on 28-may-2017.

[30] Symantec, "ISTR - Internet Security Threat Report - Volume 21," April 2016. accessed 03-may-2017.

[31] T. Xu, J. B. Wendt, and M. Potkonjak, "Security of iot systems: Design challenges and opportunities," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, pp. 417–423, IEEE Press, 2014. accessed on 03-jun-2017.

[32] H. C. Chen, M. A. A. Faruque, and P. H. Chou, "Security and privacy challenges in iot-based machine-to-machine collaborative scenarios," in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, p. 30, ACM, 2016. accessed on 14-may-2017.

[33] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012. accessed on 29-may-2017.

[34] J. Malinen, "hostapd: IEEE 802.11 AP, IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator." URL: https://w1.fi/hostapd/, January 2013. accessed on 24-may-2017.

[35] S. Kelley, "Man page of DNSMASQ." URL: http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html, May 2016. accessed on 24-may-2017.

[36] U. Lampig, R. Sharpe, and E. Warnicke, *Wireshark User's Guide*, November 2014. accessed on 28-may-2017.

[37] G. Lyon, "Nmap reference guide." URL: https://nmap.org/book/man.html, December 2016. accessed on 30-may-2017.

[38] MITRE, "CVE-2009-2692." URL: https://www.cvedetails.com/cve-details.php?t=1&cve_id=CVE-2009-2692, Aug 2009. accessed on 11-jun-2017.

[39] Y. Sheffer, Intuit, R. Holz, NICTA, and P. Saint-Andre, "RFC 7525 - Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)." URL: https://tools.ietf.org/html/rfc7525, May 2015. accessed on 11-june-2017.

[40] Philips, "Philips privacy notice." URL: http://www.philips.com/a-w/privacy-notice.html, March 2017. accessed on 09-may-2017.

[41] MITRE, "Busybox - list of security vulnerabilities." URL: https://www.cvedetails.com/vulnerability-list/vendor_id-4282/product_id-7452/Busybox-Busybox.html, May 2017. accessed on 30-may-2017.

[42] C. Labs, "Hacking my baby cam (motorola mbp853) – careless labs." https://carelesslabs.wordpress.com/2017/01/12/hacking-my-baby-cam-motorola-mbp853/, January 2017. accessed on 12-jun-2017.

Appendix A

Access Point Configuration Files

A.1 hostapd configuration

```
interface             = wlan0
driver                = nl80211
ssid                  = Thesis-Experiment
channel               = 1
wpa                   = 1
wpa_passphrase        = picky combat
logger_syslog         = -1
logger_syslog_level = 2
```

Listing A.1: hostapd configuration used in the experiments. The configuration file is for creating a WPA passphrase protected WLAN named *Thesis-Experiment*. The configuration ensures that hostapd logging is done correctly.

A.2 dnsmasq configuration

```
interface=wlan0
dhcp-range=192.168.0.10,192.168.0.100,8h
dhcp-option=3,192.168.0.1
dhcp-option=6,192.168.0.1
server=8.8.8.8
log-queries
log-dhcp
```

Listing A.2: dnsmasq configuration used in the experiments. The configuration specifies remote DNS server and local DHCP configuration settings.

A.3 custom AP script

```
#!/bin/bash

WIRELESS_DEVICE=wlan0
ETH_DEVICE=eth0

trap ctrl_c INT
function ctrl_c(){
    echo Stopping processes
    killall dnsmasq
    killall hostapd
}

ifconfig $WIRELESS_DEVICE 192.168.0.1/24 up
dnsmasq -C dnsmasq.conf -H dns_entries
sysctl -w net.ipv4.ip_forward=1
iptables -P FORWARD ACCEPT
iptables --table nat -A POSTROUTING -o $ETH_DEVICE -j MASQUERADE
hostapd ./hostapd.conf -B
tshark -i $WIRELESS_DEVICE -w output.pcap -P
```

Listing A.3: Bash script to simplify AP setup, including starting and stopping of hostapd and dnsmasq services. The script ensures that IPv4 forwarding is enabled and configures iptables to forward packets. In addition, the script also starts a program to dump packets to a file.

Appendix B
Experiment data

B.1 Motorola MBP845

B.1.1 Get Session Key Request

```
GET /?action=command&command=get_session_key&mode=local&port1=59052&
    ip=192.168.0.56&streamname=60E3ACF41A19_3603 HTTP/1.1
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; LGUS375 Build/
    MRA58K)
Host: 192.168.0.52
Connection: Keep-Alive
Accept-Encoding: gzip
```

Listing B.1: Plaintext HTTP GET request captured from the Motorola Connect App. The request is using API to make the camera send the session key for videostreaming.

B.1.2 Get Session Key Response

```
HTTP/1.1 200 OK
Proxy-Connection: Keep-Alive
Connection: Close
Server: nuvoton
Cache-Control: no-store, no-cache, must-revalidate, pre-check=0,
    post-check=0, max-age=0
Pragma: no-cache
Expires: 0
Content-type: text/plain

get_session_key: error=200,port1=59454&ip=192.168.0.52&key
    =474944723670472c335d5c4246747058&upnp=0
```

Listing B.2: Plaintext HTTP response to a session key request made by the Motorola Connect App. Motorola camera acknowledges the request and responds with the session key.

B.2 Adax NP08WIFIWH

B.2.1 Firmware update request

```
POST /upgrade?command=reset&value=9262&sign=magicpass123 HTTP/1.1
Connection: close
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; H60-L04 Build/HDH60
    -L04)
Host: 192.168.0.77
Accept-Encoding: gzip
Content-Length: 0
```

Listing B.3: Plaintext HTTP POST request captured from the Adax heater. The request is using API to make heater do a firmware upgrade.

B.3 Mill AV600WIFI

B.3.1 HTTP request

```
POST /millService/v1/selectDevicebyRoom HTTP/1.1
Content-Type: application/x-zc-object
X-Zc-Content-Length: 52
X-Zc-Major-Domain: seanywell
X-Zc-Sub-Domain: milltype
X-Zc-Timestamp: 1495192740
X-Zc-Timeout: 300
X-Zc-Nonce: 0sWhFWhju9ckrPjs
X-Zc-User-Id: 7096
X-Zc-User-Signature: cf2a5c1c956ad95da83fb8f2df1d65345ba7a067
X-Zc-Phone-Id: 865520023090120
X-Zc-Device-Os: android
X-Zc-Operation-Type: app
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; H60-L04 Build/HDH60
    -L04)
Host: eurouter.ablecloud.cn:5000
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Length: 52

{
    "timeZoneNum":"+02:00",
    "roomId":201705191118460000
}
```

Listing B.4: Plaintext HTTP POST request from the Millheat app requesting information about a device(heater) given a room ID. Request is generated by the Millheat-app and forwarded to the oven in plaintext.

B.3.2 HTTP response

```
HTTP/1.1 200 OK
Content-Length: 336
Content-Type: application/x-zc-object
Server: Jetty(9.1.5.v20140505)
X-Zc-Msg-Name: X-Zc-Ack
X-Zc-Trace-Id: 35a5c3384031ed5880101343bdee8387
Date: Fri, 19 May 2017 11:19:00 GMT

{
    "roomProgramId":201705191118430016,
    "comfortTemp":22,
    "roomProgram":"Standard Program",
    "awayTemp":10,
    "avgTemp":25.0,
    "roomId":201705191118460000,
    "roomName":"Mari",
    "deviceInfo":[{
        "currentTemp":25,
        "deviceId":9676,
        "deviceName":"Mill",
        "deviceStatus":0
    }],
    "currentMode":0,
    "offLineDeviceNum":0,
    "sleepTemp":17,
    "onlineDeviceNum":1,
    "programMode":3
}
```

Listing B.5: Plaintext HTTP response to a Millheat app request containing heater settings and sensor data, including device name, room name and current temperature.