AALBORG UNIVERSITY
COPENHAGEN

Semester: 4

Title:
Security and Privacy in IoT Architectures

Project Period:
Autumn 2017

Semester Theme:
Master Thesis

Supervisor(s):
Henning Olesen

Project group no.:
4SER 4.7

Members
(do not write CPR.nr.):
Magnus Nebel Sohn

Pages: 71
Finished: 30-10-2017

Aalborg University Copenhagen
A.C. Meyers Vænge 15
2450 København SV

Semester Coordinator: Henning Olesen

Secretary: Maiken Keller

Abstract:
In the past decade the idea of Internet of Things (IoT), where everything from sensor devices to TV's are connected to the internet, has emerged.
With a great amount of different vendors and software solutions, there is a need to address how to mitigate security in the connected devices, and ensure the consumer's privacy.
This thesis presents a design for an IoT security gateway that can raise the level of security in IoT devices, without demanding the devices to be designed in a specific way. The thesis is based on research about current IoT security solutions and consumer products on the market. This in combination has provided the relevant information to be able to create gateway, that by the help of meta data from the transmssion of IoT devices is able to notify user's about possible security breaches, and possibly block transmissions if desired.

Security and Privacy in IoT Architectures

Magnus Nebel Sohn

Master Thesis

Innovative Communication Tehcnologies and Entrepreneuship

AAU - Copenhagen

1. Introduction

Throughout the past decade the term Internet of Things (IoT), has been more frequently within the world of ICT. The term IoT covers, as the name suggests, things connected to the internet by various technologies. This could be anything from a regular WiFi connection and Bluetooth Low Energy to RFID technologies. The “things” connected vary from TV’s and coffee machines, to embedded sensors in industrial manufacturing machines. Common to all of the devices are, that they are able to connect to other devices and servers, transmit and receive data, and to some extend, act upon this autonomously. Within the recent years the number of IoT devices has exploded in both the industrial, and consumer industry. This means that the number of connected devices have extended several billions. With so many devices and a vast amount of manufacturers, there is a great need for some kind of standardization and regulation within the area. This both in regard to protocols and technologies, both especially in terms of security and privacy of both the devices, and the data they handle.
Almost every month, new examples of IoT devices being compromised appears in the media, this could be anything perpetrators taking over a number of connected devices and include them in a botnet, as to a connected “nanny-cam” sending a live footage through an insecure connection for everyone to watch.

1.2 Motivation

As stated in the introduction of this thesis there is a lack of privacy and security controls within the world of IoT. Therefore there could be huge benefits for both consumers and manufacturers to investigate the security and privacy measures in these devices, which also is the motivation for this thesis. However there is a big difference between devices made for the industry, which often are designed for a very specific purpose, and running in an internal network, and devices for private consumers, that are mass produced at a low cost, and not taking the individual consumer’s knowledge, technological skills, and setup into account.
Therefore the main focus of this thesis will be on the security measurements and privacy objectives in consumer-oriented IoT products.
The aim is to investigate how the security can be increased in the huge amount of IoT devices consumers are offered today. This in such a way that the consumer should not need any deep technical skills to increase the security in his or hers devices, and should not have to be aware of which manufacturers his or hers devices origin from. Furthermore a goal is to let the owner of the devices have control over his or hers own data, and to whom it is shared.

By this, the research question of this thesis will be as follows:

How can security measures and privacy in IoT consumer products be raised with the help of a combined hardware and software gateway, that puts the user in charge?

With the following sub-questions:
Without changing the design and implementation of the devices?
Without taking the type and make of the device into account?

The outcome of the research question will be a thorough analysis of how to secure, and ensure privacy in IoT devices, based on scientific research and real life products. This will lead to a proposal of an architecture of how to design a gateway that can handle various security risks in the end devices, and at the same time help managing the user's data, and thus his or hers privacy.
Furthermore a prototype will be developed from the outcome of the proposed architecture, in order to demonstrate how such a setup could work in a real life environment. The prototype should be able to handle and manage security risks towards IoT devices with the help of relevant frameworks and policies.
To get a better understanding of the outcome, of the thesis, a very early overview of the system will be presented in the following figure:

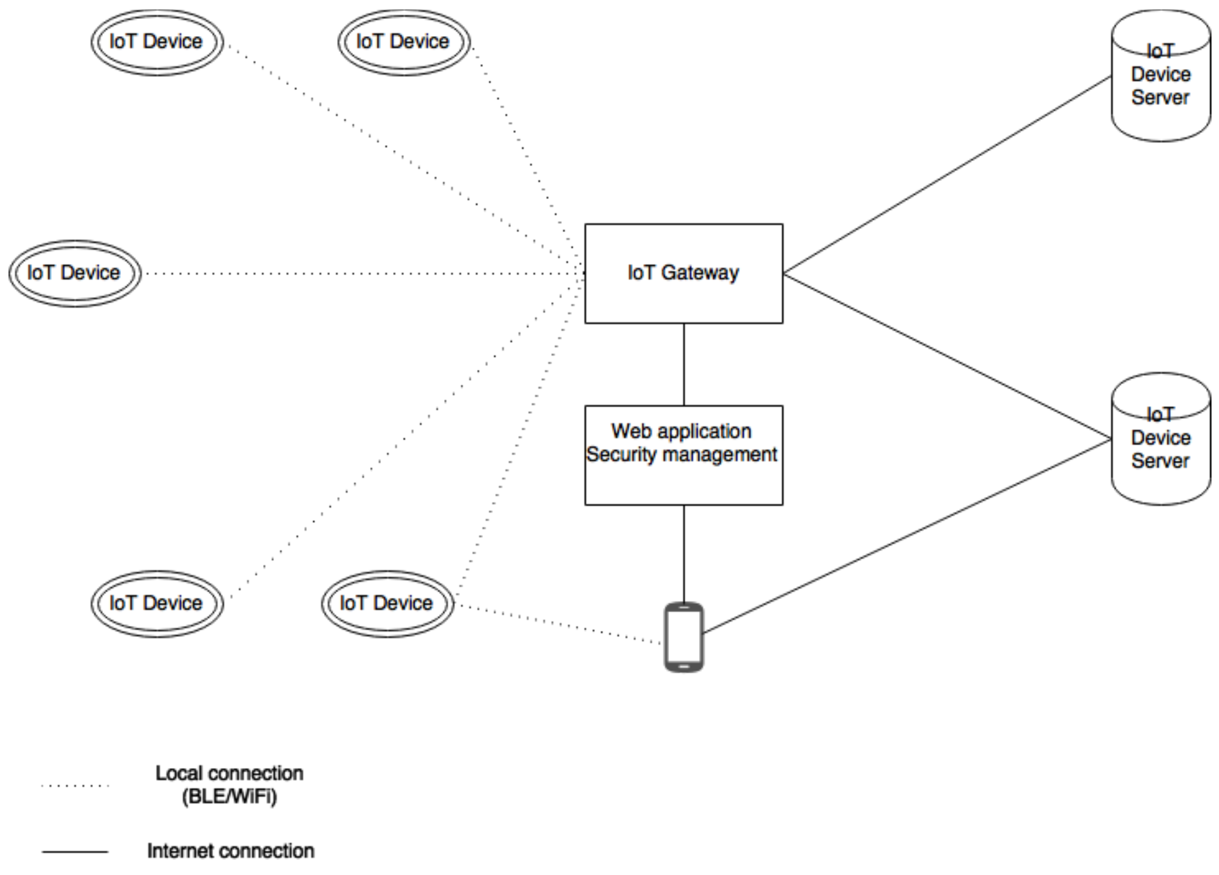


Figure 1: Initial overview of system architecture

Figure 1 above shows a proposal of how an IoT setup could be. Every device is connected to the gateway, which will handle the security measurements and the policies for the device, as well as the connection to a possible server outside the perimeter of the local network. Furthermore the IoT Gateway also connects to a web application, which is responsible for handling the policies and to communicate with the owner of the IoT devices, letting him or her take action upon possible incidents, which subsequently is communicated back to the IoT Gateway, which enforces the rules.
Besides this, an IoT device can also be connected directly to a smartphone, depending on type of the device. Regardless of how the devices connect, the gateway will always be the entity handling the security and privacy rules for all devices within the network.
This kind of setup will in that way, be able to handle to whom the user's data should be presented, and monitor the IoT devices traffic for malicious activity.

1.3 Delimitations

This thesis will primarily focus on raising the level of security in IoT devices by the means of a gateway, however will be areas within this field that will not be the focus of the thesis, these areas are:

- User Interface
- Bootstrapping of the devices
- User authentication and authorization

It is clear that these parts are needed in order to create a full implementation of the proposed system, however it is considered out of scope for this thesis and therefore will not be analyzed and implemented. The last chapters of the thesis discussing future work might lightly touch some of the areas, but they are not seen as an essential part of this project.

2. Methodology

The methodology used in this thesis cannot be described as a straight road to, create a solution to the problem stated in the early beginning of the thesis. If any it is better described as a bumpy gravel road from the beginning to the end.
As many projects, this started with an idea of a problem one has noticed in during one's everyday life, in this case the rising of IoT devices, and the problems that lies within connecting all kinds of products to the internet. Also as for many other ideas, the solution to a problem is imagined right away, without concerns about whether the problem already has been solved, or for that matter how to solve it, which was also the case for this project, with the envisioned solution lightly described in the introduction. Needles to say, the case is seldom that easy to solve.
Therefore a structural approach to solve the problem was needed. The first thing to do was to take a look into what kind of real devices existing on the market, to get a grasp of the functionalities and technical setup of these devices. This has helped provide an idea of the subjects that had to be protected and as such how the system should be able to solve this. Naturally it is of interest to look into what have been done before this project, both to know how researchers propose to solve this problem in general, and what tools can be used to solve the problem. Last but not least, it was interesting to look into known attacks towards IoT devices.
All of this information has been used as background information for solving the problem. It has been the basis for the analysis in the thesis, and thus the basis for the system as a whole.
Yet the approach has not been a direct research, analyze, develop approach. During a process like this, new information will be uncovered, forcing one to revisit both all different sections of the project, all the way through the process. New ideas emerge, making initial findings seem obsolete and therefore needs to be changed. This has indeed also been the case for this project.
The process and method of this project, can in that way said to be very iterative, where every part and chapter has been revisited with new findings, whenever they would arise.
In the same way the research and analysis part have not been totally separated from the developing part, as the thoughts about how to develop the solution has been in the background throughout all phases of the project, and that development and testing solutions and technologies has been a parallel track to the research and writing. It could be called an agile process, where one hasn't decided on a single solution in the beginning and sticks to that until the end. In reality it is the only feasible solution to do this, as the risk of a lot of work would be wasted by basing everything on a waterfall model, just going straight from one end to another.

3. Background

The following chapter is going to cover the relevant background research for this thesis. The aim is to create a basic overview of IT security, both general security principles, and how they relate to IoT systems. Additionally the chapter will reveal information about attacks, vulnerabilities, and how they can be mitigated. This primarily by going through research within the area, to uncover already known problems and how they have been mitigated. Lastly the chapter will present relevant work trying to mitigate threats in IoT services, that will help to understand what to focus on, when designing a system that should be able to protect IoT home systems.

3.1 The CIA Triad

When working with IT security, often three core concepts comes into play, namely "Confidentiality, Integrity, and Availability". These three concepts are also commonly referred to as the CIA Triad. As described in [1] these concepts covers most aspects of IT security, described as the following:

"

Confidentiality: *Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.*

Integrity: *Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.*

Availability: *Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.*
"[2].

Even though the CIA triad provides a good high level overview of different aspects of security, it is very generic, and does not specify relevant functions of security and how to implement these.
Regardless of this, it has been used for several years, and is still used today. Therefore it will also be a part of the analysis in this thesis, as the model is easy to manage, and thus can be used as a guideline, when analyzing the problems opposed to.

3.2 Related Research

The literature about IoT is vast, since the term was first mentioned by Kevin Ashton in 1999[3], the research conducted within the area has been tremendous. Everything from very simple applications with RFID chips getting scanned, to IoT networks covering whole cities is a part of this area of research. However in this section the focus will primarily be on “newer” IoT applications where the devices is interconnected both to each other, and to the internet, either directly, through a gateway, smartphone, or similar. As the aim of this thesis is to create a solution for devices that are constantly connected, and not just when scanned by a RFID or NFC reader.

In a blogpost on eSecurity Planet, Sekhar Sarukkai states that a Hewlett Packard study from 2014 reveals that up to 70 percent of IoT devices are vulnerable to an attack, and another study from Data Corporation estimates that by this year 90 percent of organizations will have a breach related to IoT.[4]
Furthermore a white paper released by Canonical in January this year elaborates on the threats to IoT devices and why they exits. The paper lists some of the vulnerabilities to IoT devices, by the security researcher Brian Krebs[5]:

“

- *Hard-coded passwords*
- *Fundamentally weak security at both the software and hardware levels*
- *Lack of software updates*
- *The size of the opportunity*

“

The reasons listed above, are obvious and probably on the first page in the book of security researchers and advisors, nevertheless it still poses a problem for the industry. As this thesis does not aim to explain why IoT devices are so vulnerable compared to other ICT systems there will not be thorough analysis of why these vulnerabilities still exits in products, whose very nature are connectivity and data transmission. However it is interesting to look into the flaws of IoT security and privacy, and what researchers and professionals have done to mitigate some of these vulnerabilities.

In an analysis conducted by Gaona-garcía, Paulo et. al. they divide the IoT into three main layers, similar to the OSI model, as the following[6]:

- *The lower level, is the perception layer used mainly to capture, gather, distinguish and identify object information. The layer includes RFID tags and literacy devices, cameras, GPS, sensors, laser scanner, and so on.*
- *The second level is the network layer, which is used to transmit and process information obtained by the layer of perception and provides such information to the application layer, with the support of reliable communication.*

- *The upper level is the application layer, used to process data intelligently, and aggregation of data from various sources with different types. The layer implements control and information management, making use of cloud computing, data mining etc.*

As the study suggests, this model is one way to help design a reliable and secure IoT network.
Looking at the models three layers, the lower level, which can be compared to the physical layer in the OSI model, takes care of the data capturing done by the IoT devices. For a third party it is hard to change anything in this level, as it is build into the hardware.
The second and upper level on the other hand deals with the network and application layer. In these levels of the infrastructure it is possible to for at third party, which is not the manufacturer of the product, to work to improve the security, both in terms of analyzing the traffic and mitigating attacks.

From this model, it makes sense to take a look into the relevant layers, how they work and their security and privacy measurements. The same study, has depicted a model that identifies the security areas in IoT(see figure 2)

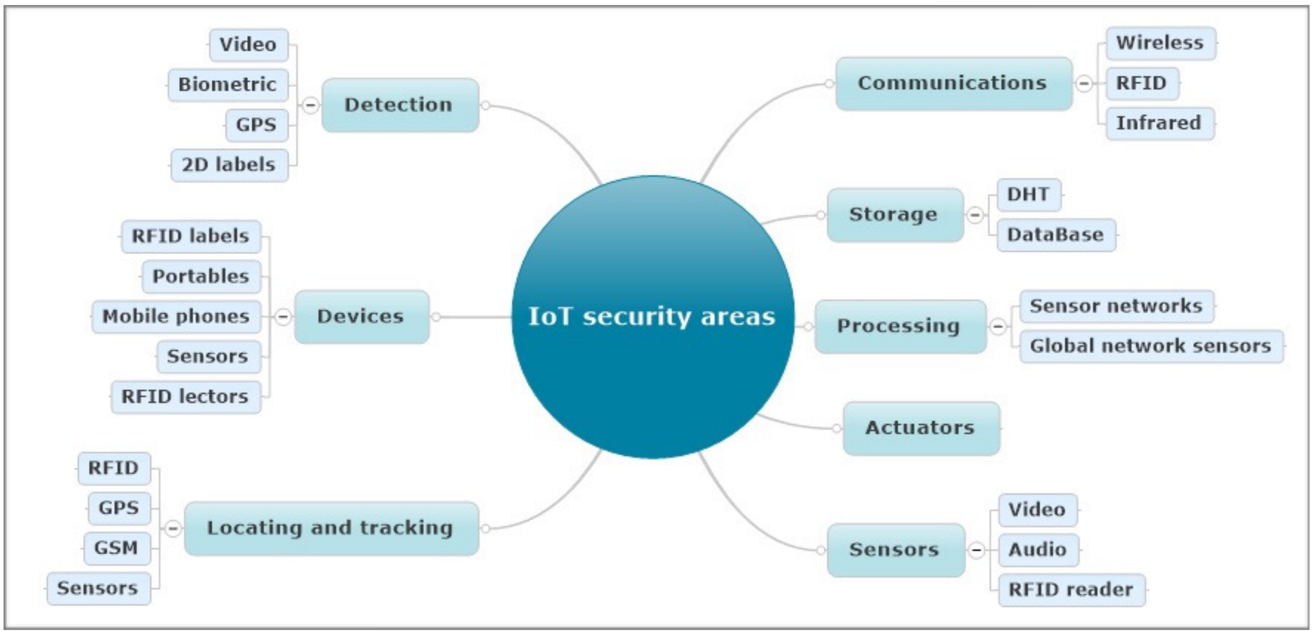


Figure 2: Security areas in IoT[6]

From this model especially three areas are of interest: “Communications”, “Actuators” and “Devices”.

Communications: The type of communication technology and protocol, the type of security implemented, if any.

Devices: How are they secured, and how can they be infected e.g. with malware that are capable of DDOS attacks or transmitting private data.

Actuators: Who are the actuators, what can they do with the devices and what kind of data do they get access to?

Some of the conclusions from the study, shows what vulnerabilities IoT devices face, such as:
“If communication channel is not adequately protected by encrypting data, it can be easy for an attacker to carry out attacks.”[6]. and “Another common feature characteristic to a large quantity of IOT devices, is that they use cloud services. In this case these applications have other potential risk; for instance; if there are deficiencies in the management or update the platforms; intruder would be able to access the information store and even take control of the IOT device.’[6]
Which leads to another risk, which have been debated a lot lately, namely the following:
“There is a specific need for research into the availability of communication due to DDoS and service provided by IP. In addition, the integrity of the devices must ensure their freedom from malware such as spyware or rootkits, seeing the need for more research. Finally, almost all areas lack mechanisms applicable in the privacy of Internet of Things’[7]

Despite the fact, that there are great risks associated with IoT and the devices, frameworks trying to handle some of the problems with IoT have been developed, or is under development. In a research paper by Mohammad Irshad from the Singtel Cyber Security Institute[8] 4 different frameworks are analyzed in order to get an overview of how the security threats are handled according to ISO 27001. Even though these frameworks are able to handle some of the security issues in IoT, they are not able to take care of the issues if the connected devices themselves does not implement the means from the frameworks. Therefore, until the frameworks suggested are implemented in the devices, there will be no assurance that the devices are able tot handle the security risks they are faced with.

3.2.1. Risks and Threats

In order to be able to mitigate risks and threats in the world of IoT, an overview of these are needed. As there exists several threats and risks for regular computer systems and networks, so does for the IoT. Several studies emerge around this issue. In this subsection, some of the findings from the research about this topic will be revealed, in order to gain knowledge about the focus areas of the subject. The section will be a summarization of the relevant findings, meaning that not all risks and threats will be covered, merely those found relevant to the specific problem stated earlier.

In the study “Threat-based Security Analysis for the Internet of Things” by Ahmad W. Atamali and Andrew Martin[9] three IoT use cases, their threats and impacts are reviewed.

In the study three sources of threats are discovered, namely: Malicious User, Bad Manufacturer and External Adversary[9]. From this different security properties and privacy issues, in regards to IoT are deducted, whereas the most relevant to this project are:

"

- *An access control mechanisms is needed in each device in order to prevent unauthorized access from compromising the entire system…*
- *Identification and authorization mechanisms should be employed…*
- *The data exchanged between a user and the IoT devices should be protected so that an attacker eavesdropping on the communication can't infer information about the user…*
- *The messages exchanged between IoT devices must not reveal Personal Information Identity (PII) of the user.*
- *Signals from a device must be sent in a privacy preserving manner so as not to reveal the device's function since this can reveal information about the user.*
- *The IoT devices should keep a record of personal user information only when absolutely necessary, and in such a case it should be for a limited time only.*
- *Only data that doesn't reveal the personal information of the user can be collected such as aggregated data, e.g keeping a record of the number of people in a building, but not data relating to their identity like name, ID, and visual image.*
- *The user should be made aware what and when data is being captured.*
- *The user must be able to securely erase all private data from a device, e.g. if the device is to be resold.*

"

Depending on the device and their services, the above listed properties might not be desirable nor necessary. Regardless, they provide a good overview of what to take care in terms of privacy and security in IoT devices.

Another study, "Security and privacy in the Internet of Things: Current status and open issues" from the same year, also assesses the status of security and privacy in IoT[10]. Also here it is found that some of the threats to IoT is Denial-of-Service attacks as well as attacks on privacy. Where some of the attacks on privacy are described as: Eavesdropping and passive monitoring, Traffic analysis and Data mining[10].
Besides this the study finds that some of the challenges within IoT consist of User Privacy and Data Protection, Authentication and Data Management, Trust Management and Policy Integration, and Authorization and Access Control.[10]

A third study from 2012 by Kozlov et. al.[11] maps threats in IoT Architectures in different operational areas such as health care and home electronics. The conclusion from the study reveals the important threats are as follows: eavesdropping, man-in-the-middle, data confidentiality and getting control of components by malicious and unauthorized persons.[12]

In a study by Barbar et. al. [13] a proposed hardware and software design method is presented. As the study focuses on the full stack of IoT devices including the hardware design, not every of the findings are relevant, however some of the proposals are of interest to this thesis:
“
User identification: It refers to the process of validating users before allowing them to use the system.

Secure network access: This provides a network connection or service access only if the device is authorized.

Secure data communication: It includes authenticating communicating peers, ensuring confidentiality and integrity of communicated data, preventing repudiation of a communication transaction, and protecting the identity of communicating entities.

Identity Management: It is broad administrative area that deals with identifying individuals/ things in a system and controlling their access to resources within that system by associating user rights and restrictions with the established identity.
“

Another more recent study by Pasha et. al.[14] describes a security architecture divided into groups, namely the physical part which constitutes the following: Local IoT devices, Local Networks, Physical access to local devices and networks.
The network part: Back end cloud services, communication between cloud and IoT devices
The application part: Communication between mobile applications and IoT devices, communication between the user’s application and cloud services.
The study proposes several ways to mitigate the threats, such as making sure to change default passwords, making sure the cloud is protected and to check security measurements before buying a product. Regardless of these suggestions, it does not help to protect an unsecured or vulnerable device, which is also discussed in the paper, suggesting that IoT devices should be secure by design.
Furthermore the study lists nine important attack vectors relevant to IoT systems. These vectors lists as the following[14]:
“
1) Weak authentication or authorization.
2) Weak security on network services.
3) Weak security configuration.
4) Lack of encryption in transportation.
5) Weak security on cloud interface.
6) Weak security on mobile interface.
7) Weak Firmware security.
8) Weak physical security.“

By this, it has been established, that there indeed exists security and privacy threats within the world of IoT that needs to be addressed. Even though many of the threats looks similar to the ones known from regular computing and networking, they way to mitigate these threats calls for different solutions than the ones already known. Primarily because most IoT devices are very limited in terms of interfaces and computing power, compared to regular computers. This means that the user will have another way of interacting with the devices compared to a "regular" computer. Therefore the user will not necessarily know when an IoT device is attacked. Furthermore, due to the limited resources of these devices, normal countermeasures such as antivirus and similar are not applicable, making the devices harder to protect.

3.2.2. IoT Security Solutions

When looking into security in IoT devices, one have to consider the differences between these kind of devices and regular computers and security measurements related to those. Seen from a high level perspective the objectives one wants accomplish are the same as known for computer security in decades, namely: Confidentiality, Integrity and Availability, naturally depending on the application of the device. However, as the devices used for IoT applications often differ from regular computers, the means to accomplish these goals can be very different as those known from regular computer or IT security. Which has also been described in the preceding section.
Several reports have been analyzing how security frameworks for IoT devices could be designed in the future. This section aims to elaborate some of that work, in order to help identifying how a system to enhance IoT home devices security could be designed.

In the research paper "Security Requirements Analysis for the IoT" by Se-Ra Oh and Young-Gab Kim[15] findings about security requirements in IoT is presented, whereas some of the most important are the heterogeneity in the field of IoT, where it is argued that *"The biggest problem of heterogeneity is absence of common security service"*, and *"For providing common security service, unified IoT security standard has to be established'*[15]. It is also argued that due to the lack of performance in many IoT devices, it is not possible to integrate known standards such as TLS and AES in the devices, therefore new algorithms taking this problem into consideration should be designed. Furthermore the work mentions different attacks related to IoT, most of them are known from regular IT security such as: buffer overflow, sniffing, man-in-the-middle and spoofing. However it is stated that namely because of the heterogeneity in IoT devices, it is hard to overcome such attacks, and because of the lack of performance in IoT devices, it is also increasingly difficult to overcome the problems with e.g. A DOS attack.
Similar to the paper by Se-Ra Oh and Young-Gab, Jarkko Kuusijärvi et al. has presented a research paper "Mitigating IoT Security Threats with a Trusted Network Element" where a question from the SANS institute is presented, as the following: "What do you think the greatest threat to Internet of Things will be over the next 5 years?"[16] 31% of the replies

was “Difficulty patching Things, leaving them vulnerable”[16]. This answer can to some extent be seen in relation to the findings in the paper by Se-Ra Oh et. al. and the heterogeneous nature of IoT devices. However Jarkko Kuusijärvi et al. introduces a more specific solution to the problem namely what they call a “Trusted Network Element”.

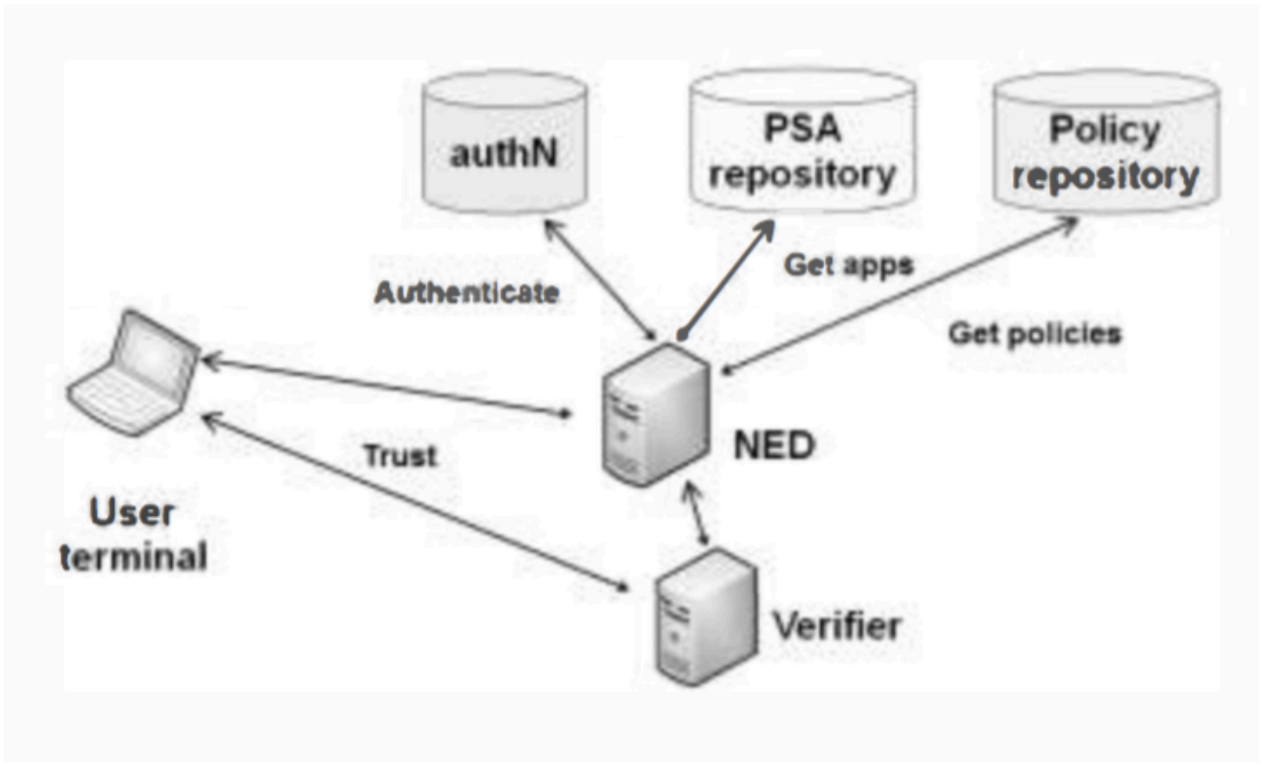


Figure 3: SECURED logical architecture[17]

The figure (3) above, depicts an overview of how the solution with a trusted network element works. As described in the paper[18] the idea is to let all user terminals, that be a regular computer or an IoT device, be connected to the Network Edge Device (NED) which then handles the security for the user terminal. This is done by the help of different repositories in example the PSA repository is described as follows: “*The user's security policies are enforced by Personal Security Application(s) (PSAs) running in the NED. The PSAs are the security controls required for enforcing confidentiality, integrity and availability. The PSAs are fetched by the NED from a PSA repository.'*[19]. Furthermore the NED adds another layer of policy handling, as it also provides a “Policy repository” where policies in different levels can be defined, depending on the type of user and organization the user terminal belongs to, in example a specific company or organization[20].
The general ideas of these policies are that the can be enforced by the NED, depending on the user terminal (in this case an IoT device) connected to the NED. The NED enforces different policies making it possible to raise the level of security for the user terminal connected through the NED. As examples specific for IoT devices, can be mentioned policies such as bandwidth control, malware detection, logging and several other functionalities (for a full list of the functionalities, see appendix A).

The NED also features other functionalities such as verification and authentication services, which are highly relevant as well, however this section will not go into details with that, as the most relevant parts are the offloading of security services to the NED instead of letting the user terminal/IoT device handle the security.
This approach to IoT security aligns very well with some research from Gartner Inc. In an analysis from November 2016, "Predicts 2017: Security Solutions"[21] some of the key findings and recommendations are as follows:

"*Expect a shift in spend from endpoint-based IoT security to gateway-based security features."*

"As a result of current network firewalls' limited ability to understand, share and collaborate on internal network threats, firewall providers will adapt existing firewall policies to incorporate historical threat detection into access control policy decisions."

"The adoption of public cloud computing will drive demand for MSSPs to provide security monitoring capabilities. This will create competitive differentiation opportunities for the few providers with security monitoring service capabilities for public cloud environments."

"The specific nature of many IoT devices and sensors makes many traditional security controls inadequate; therefore, develop new security delivery models, such as IoT security gateways and embedded security features at the semiconductor level".

From the preceding chapters it becomes apparent that the risks and threats for IoT solutions indeed exists, and these problems in many ways are the same as known from general IT security. However it is also shown that the trying to solve these problems cannot necessarily be handled in the same way as it has been done with regular computers.
The findings clearly shows a tendency for IoT security handling to be expanded beyond the actual IoT end devices, to an end point similar to the suggestions presented in the paper "Mitigating IoT Security Threats with a Trusted Network Element".

3.2.3 Frameworks

In order to have a basis on how to analyze and solve some of the security problems in IoT consumer devices, it makes sense to have a described framework to define objectives of what should be accomplished. As it has already been described earlier, the world of IoT is very heterogeneous, therefore the frameworks to be used should not be to specific to a certain area, but more generic, in order to use them as a guideline for the analysis.

A good example of such a framework is the "Framework for Improving Critical Infrastructure Cybersecurity" by NIST[22]. The framework is not specific to IoT devices, but helps with a good high level overview of how to address IT security, whether to be used in organizations or in this case to analyze security in IoT devices.

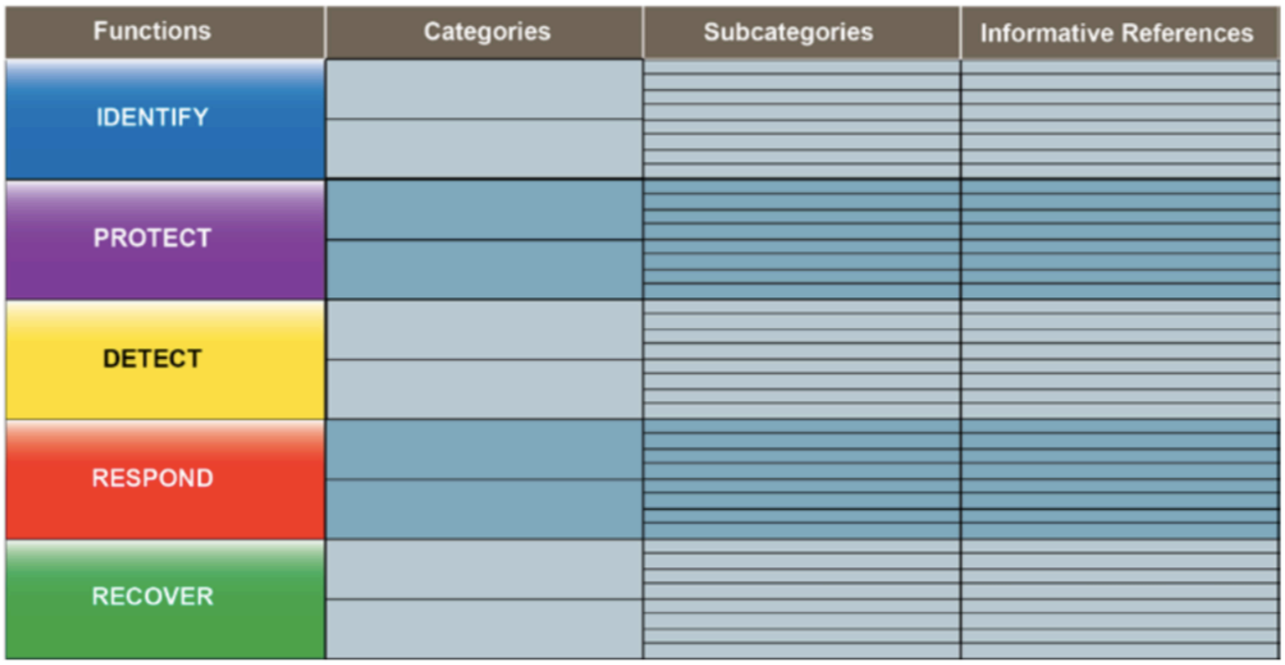


Figure 4: Framework Core Structure[23]

Figure 4 above shows a high level overview of the framework, where five different functions are defined. By the publication, it is described that the functions aids in expressing the management of cybersecurity risks, addressing threats, and improving and learning from previous activities[24].
The categories and subcategories of each function is used to split each function into lower level activities such as access control and detection processes.
According to the publication each function can be described by the following:

- "Identify – Develop the organizational understanding to manage cybersecurity risk to systems, assets, data, and capabilities.
- Protect – Develop and implement the appropriate safeguards to ensure delivery of critical infrastructure services.
- Detect – Develop and implement the appropriate activities to identify the occurrence of a cybersecurity event.
- Respond – Develop and implement the appropriate activities to take action regarding a detected cybersecurity event.
- Recover – Develop and implement the appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity event."[25]

As the framework is not directly related to IoT security, it is not necessarily all functions that can be implemented as described, and some might not even be applicable to the concept, regardless of this, it is believed to be a good guideline when analyzing the proposed solution.

Another paper by Dong Hee Kim et. al.[26] identifies some of the key requirements of security in IoT for service providers. The paper focuses on ranking the most important security requirements in an IoT services lifecycle for service providers of IoT. This means that some of the requirements requires access to the software of the device in example to ensure the use of secure protocols, therefore not all requirements of this model will be applicable to this project, as the aim is to raise security without having the possibility to change the implementation of different IoT devices, regardless, the model gives a good overview of important requirements for IoT security.

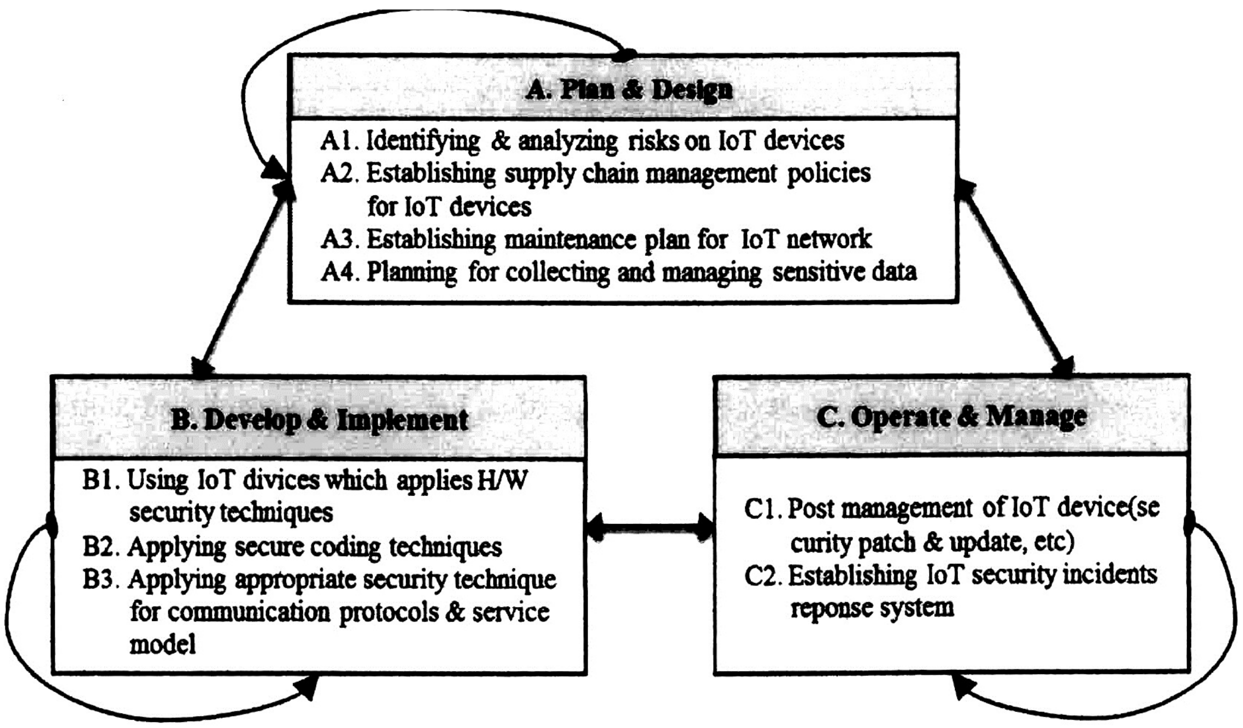

A. Plan & Design
A1. Identifying & analyzing risks on IoT devices
A2. Establishing supply chain management policies for IoT devices
A3. Establishing maintenance plan for IoT network
A4. Planning for collecting and managing sensitive data
B. Develop & Implement
B1. Using IoT divices which applies H/W security techniques
B2. Applying secure coding techniques
B3. Applying appropriate security technique for communication protocols & service model
C. Operate & Manage
C1. Post management of IoT device(se curity patch & update, etc)
C2. Establishing IoT security incidents reponse system

Figure 5: ANP Network model for IoT security requirements[27]

Especially Phase A and C as shown in figure 5 above are of interest, noticing that objective A1 aligns very well with the NIST framework described in the preceding section, and phase C; operate and manage are of high interest, as the proposed solution will have a high focus on managing the security of IoT devices connected to a consumer's home network.

These two frameworks will be a basis of the analysis in order to be able to design a system that raises the security level of IoT home devices. Therefore these will be referred to throughout th analysis among with the other findings of the background chapter.

3.3. State of The Art

In the following chapter State of The Art relevant real world attacks, products and technologies for this thesis will be elaborated. The chapter will cover a broad number of subjects. The first part of the chapter is going to focus on known attacks at different IoT services to show that these problems exists outside the academic literature.
Furthermore some of the IoT consumer devices on the market today will be described, this description will lead to create more generic scenarios of how different IoT setups are designed, to help analyze setups and data flow of such devices.
Lastly some of the technologies that will be used to design the prototype, that should be the outcome of this thesis will be described, this is done create an outline of the capabilities of the technologies used in the design of the system.

3.3.1. Known Attacks

One thing is how researchers and companies within the world of IoT analyses threats and tries to design IoT systems more persistent to threats than those known today. Another thing is how IoT systems weaknesses are actually exploited in todays world. This subsection will investigate some of the known attacks to IoT devices, in order to uncover what problems exists in the world of IoT today. It will not be possible to uncover all attacks, but it will provide and overview of what the world has seen in the recent past.

Voice Recordings Held for Ransom
A news story from February 2017[28] tells the story about a kids product, a stuffed teddy bear able to record voice conversations, has been attacked. Two million voice recordings were held ransom. The product is able to record voices and conversation, and by the help of an application installed on the user's smartphone store and transmit those recordings to other users'. The attack happened due to an insecure MongoDB at the company's backend. The insecure database let the attackers harvest all the information, delete the original information, and thus demand a ransom to let the users get access to their information again. Furthermore the general login information was very weak, making it easy to access user information such as email addresses and login data.

Worm Infecting Light Bulbs
This attack is developed by scientists that explore the Philips Hue light Bulbs. In a story from The Register[29] It is explained that researchers have been able to create a worm that exploits hardcoded encryption keys, and thus is able to jump from bulb to bulb just via the direct Zigbee connection between bulbs. The worm is described as the following by the researchers:

"The worm spreads by jumping directly from one lamp to its neighbors, using only their built-in ZigBee wireless connectivity and their physical proximity. The attack can start by

plugging in a single infected bulb anywhere in the city, and then catastrophically spread everywhere within minutes, enabling the attacker to turn all the city lights on or off, permanently brick them, or exploit them in a massive DDOS attack.'[29]

The Mirai Attack
The Mirai attack happened in October 2016, where a massive DDOS attack was aimed at Dyn (an internet infrastructure company), around the world. The attack primarily came from insecure IoT devices. As Brian Krebs describes on his blog KrebsonSecurity[30], the attacks was primarily performed by hacked IoT devices. The Mirai searches the internet for unprotected devices with the aim of overtaking them for a DDOS attack. The attack is mainly possible due to badly protected IoT devices with weak usernames and passwords. An example of this is provided by the Computerworld[31] which is telling the story that a Chinese manufacturer with the name of Hangzhou Xiongmai Technology who produces DVR's and internet connected cameras, became a part of the attack. Due to weak default passwords, not changed by the buyer of the product, the Mirai Malware spread to more than 500.000 of their products, helping the attackers to create a major bonnet able to perform a massive DDOS attack.¨

3.3.2. Smart home devices

Philips Hue
The Philips Hue, is possibly one of the most known commercial IoT systems for the home. The Philips Hue System is a connected set of light bulbs which can be managed via a smartphone application or similar. The system consists of four main components[32]

Apps: This can be anything from the official smartphone app from Philips to home composed apps created by the helps of the Hue API.
Bridge: The bridge handles the connection between the bulbs and the apps through API's Where most of them requires a direct connection to the bridge.
Portal: A web portal provided by Phillips that connects the bridge, and thereby the bulbs to the internet.
Lights: This is the actual LED light bulbs. The lights are able to create a mesh network between each other, and naturally connect to the bridge, all of this communication is handled through the Zigbee technology[33].

The Philips Hue system let developers create their own applications by the help of their RESTful API, however it is important to note that it is only possible to use the API if the application is running on the same network as the bridge[32], whereas Philips' proprietary smartphone application lets the user control the lights from anywhere as long the device is connected to the internet[33].

Google Nest
The products from Nest, are mainly based on home automation and connectivity, they started building their first prototype of a thermostat in the end of 2010 [34]. Now the company offers several products such cameras for surveillance and connected smoke alarms[35].

Directly connected devices
Another type of products within the world of home IoT devices, are the ones that are directly connected to the home's internet gateway through WiFi of Ethernet. These kind of products, does not come with a manufacturer controlled bridge like the ones presented previously, but are connected directly to the internet like at smartphone or home computer.

An example of a directly connected device is the coffee machine Smarter Coffee by the manufacturer Smarter[36]. The manufacturer does not tell much about the architecture of the coffee machine, besides that it comes with WiFi connection and app from where you can control the coffee machine. However looking at the manuals for the machine on their web page[37], it shows that connecting the machine to the internet, is very similar to connecting any other WiFi device, namely choosing a network to connect to and enter the required credentials. As the product does not offer any APIs or web interface, the only means of control for the user is through the proprietary app provided by the company. This also means that user does not have any control of how the device connects to the internet and what kinds of data it transmits to the company's server.

Local connected devices
A third way of connecting devices, is through a local connection directly to a computer or smartphone. An example of this is the Nespresso device Prodigio [38]. This machine has a Bluetooth connection, that connects directly to a smartphone and by the means of the Nespresso app, it is possible to brew a cup of coffee, check status of the machine, etc. In this case the device's connection to the outside world is handled by the user's smartphone and the app provided by the manufacturer. This means that the device is can only communicate when connected to a smartphone via bluetooth. By This, it is solely the smartphone and the belonging application that decides what kind of data to transmit and when to do it. This makes the user unable to control what he or she shares with the manufacturer (and others), and the user has to rely solely on the user agreement provided by the manufacturer.

3.3.3 Scenarios

From the preceding section, it has been shown that IoT devices can work and be connected in several different ways. The four examples of IoT products in State of The Art, does far from cover every application or scenario, but it provides an overview of some of the functionalities IoT home/home automation devices comes with. The devices described

all have different functionalities, however the actual functionality is not the most important discovery. If the device provides a way to control a home's lights, temperature or coffee making, is not of interest. The important note is the architecture of the devices and how they connect.
The products described in the preceding section is not analyzed completely both in regards to functionality and architecture, this is mainly due to two things; the lack of information provided by the manufacturer, even though some of the devices presents API's for customers and others to use, and because even though a full review of the devices was possible it would not change the fact that all of the devices are proprietary and is meant to work "out of the box". Therefore is it not possible to change how their hardware nor software actually works.
Instead the assessment of these products will be used to create an overview of the different ways IoT devices can connect to each other and the internet, in order to be able to analyze how the security and privacy for the user can raised.
In the coming paragraphs, different setups for IoT devices will be presented in a generic way, based on the reviews in the section 3.3.2.

Setup A:
The first setup is deducted from the functionality and architecture of the Phillips Hue system and the Nest system. This setup is based on the actual "things" in where the functionality lies. These things have a communication channel, in the case of Philips it is Zigbee, but it could also be Bluetooth LE or similar. The main thing to notice is, that the things themselves are not able to connect to the internet, a smartphone or computer, they need an intermediary, which in the case of the Philips Hue system is a bridge that connects the devices to the home router, from where it is possible to communicate with other devices. However the things can create a mesh network where they are able to communicate in-between, so not all of them has to have direct connection to the bridge, information from the bridge can travel to a "thing" far away by the mean of other "things" closer to the bridge.
To get a better overview of how such a system would look like a generic setup of such a system is illustrated in figure 6 below.

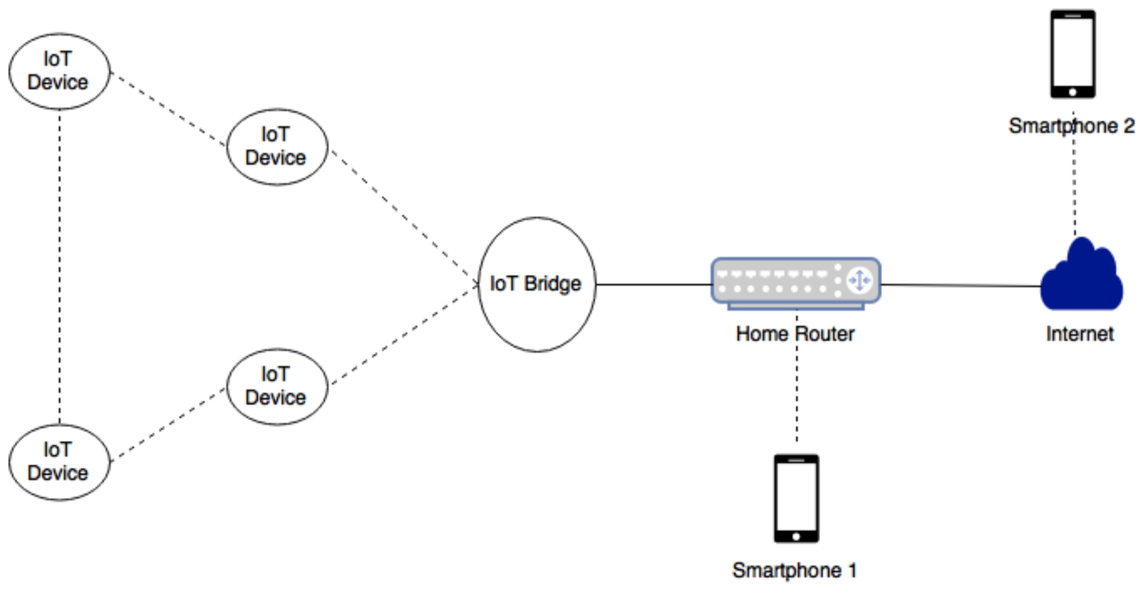


Figure 6: Generic overview of setup A

As visualized in figure 6, the setup consist of 4 IoT devices connected to a bridge, either directly or through a mesh network, letting the devices furthest away connect to the bridge through other devices. The bridge mitigates the connection to the home router. The home router then takes care of the connection to the user agent, in this case visualized by a smartphone. This can be done in two ways: To a user agent connected directly to the home router, or through the internet with the possibility of going through several servers provided by the manufacturer of the IoT device.

Setup B:
The second scenario, describes IoT devices that are directly connected to the home router and thereby the internet. These kind of devices has their own WiFi chip, and are thereby able to connect without an intermediary, just like a computer or smartphone that is WiFi enabled.
In this case the user agent, it be a smartphone or computer, will connect directly to the IoT device through an interface, either provided on a manufacturer controlled web server, or by connecting directly to the device which runs it own interface.

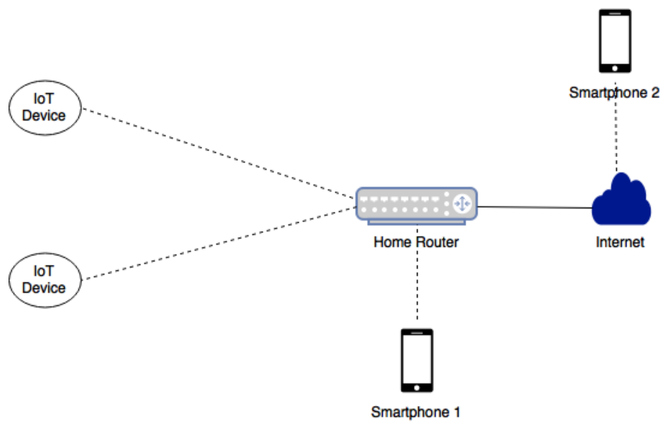


Figure 7: Generic overview of setup B1 with no manufacturer controlled web server

Figure 7 shows how a setup with the IoT device directly connected to the home router and no manufacturer controlled web server used to control and setup the devices. In this setup the user agent connects directly to the IoT device either within the local network, or through the internet, only with the home router as an intermediary letting the IoT device connect to internet.

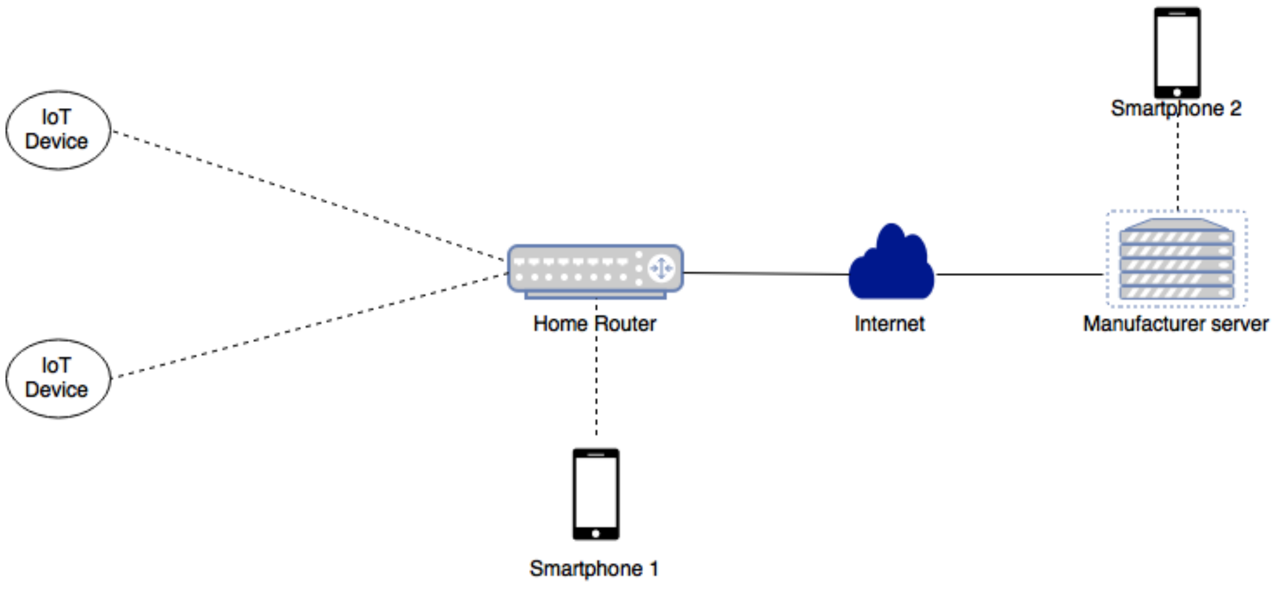


Figure 8: Generic overview of setup B2 with a manufacturer controlled web server

The above figure shows a setup of scenario B, again where the devices are directly connected to the internet through the home router. However in this case a manufacturer controlled server on the internet handles the communication between the device and the user agent, and thus the server provides the interface, instead of the actual IoT device. The two setups differ in the way the interfaces to the devices are handled, as one has a manufacturer server provided, and the other has the interface directly build into the device. However they go into the same group, as the commonality is their direct connection to the internet, through the home router. This in reality entails that the only security measurement between the devices and the great internet is the home router, possibly with a build in firewall.

Setup C:
The last setup differs most from the two others, in this case the IoT device does not have the possibility to connect to the internet without an intermediary nearby. The communication has to go through a smartphone with a manufacturer developed app installed.

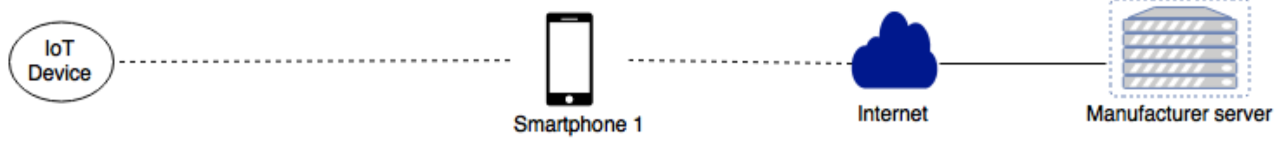


Figure 9: Setup C: Generic overview of local connected IoT devices

In figure 9, an overview of how local connected IoT devices works. These kind of devices needs a connection to a smartphone in order to be able to communicate with the outside world. This connection goes through an app provided by the manufacturer, and thus it is solely the manufacturer that controls the functionality of the device through the functionalities in the app. Furthermore the device's connection is restricted to when a smartphone with the manufacturer provided app is in proximity of the device. In other terms the devices can only be controlled when the user is inside or very close to his or hers home.

The three scenarios described in this section, are elicited by the IoT products described in section 3.3.2. The idea behind the scenarios is to present an overview of how different IoT setups looks like in a generic way. Not taking the specific service or product into account. With the help of these three setups the coming analysis will be able to uncover how a system should be setup in order to solve the problem definition claimed in the beginning of this resport, and thus deduct a requirement specification for such a system.

3.3.4. Technologies

In the following section some of the relevant technologies to be used in the project will be described to create an overview of what they are capable of. The description will merely serve the purpose of an overview, more detailed explanations can be given throughout the report

Raspberry Pi
The Raspberry Pi[39] is small computer with limited capabilities, running on an ARM system, making it perfect for running a Linux distribution. The Raspberry Pi employees an ethernet port on all model and for some also a WiFi chip. As the machine is physical small, low cost and low power it makes a good case for running the IoT gateway in the proposed setup.

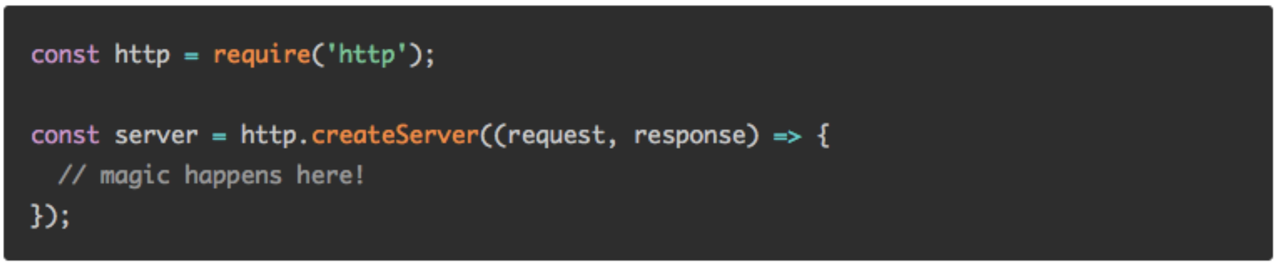
Python
Python is an object oriented programming language[40] which runs of a variety of Unix platforms, Mac OS and Windows. The programming is supported with a huge number of libraries, in example over 1000 packages for network programming exists for Python[41].

REST
REST is an architecture that is defined by being stateless and uses HTTP methods. It was originally described by Roy Felding in 2000[42].
A RESTful setup provides the possibility to communicate with a web services through well known http methods such as: POST, GET, PUT and DELETE. It provides the possibility to communicate with a web service and request and update resources only through these methods. Furthermore it transfers both XML and JSON formats which can be useful when communicating between the devices.

Mongo DB
Mongo DB is a document oriented database, which stores data in JSON like formats[43]. This also means that it is not relational like in example MySQL databases. Naturally this have both it benefits and drawbacks, however for this system the MongoDB seems like a good choice, this is for several reasons:

- The database is supported by Python and is easy to connect to through Python
- The JSON like format makes it easy to transfer objects to and from the database by a RESTful service without having interpret the data transmitted.
- As the database is not relational it is easy to update with new information without having to update the structure of the database. This can be an advantage in a project like this very additional discoveries might occur both during the design, but also when the system is running.

NodeJS
For building a server or web application it makes sense to use NodeJS, possible in collaboration with other tools. NodeJS is by themselves defined as “Node is designed to build scalable network applications.”[44].
Node is build in Javascript and by the help of HTTP, this also makes sense in relation to the RESTful methods thought to be used throughout this project.
In example to create a server object is quite simple, see figure 10 below:



```
const http = require('http');

const server = http.createServer((request, response) => {
  // magic happens here!
});
```

Figure 10: Create server object in NodeJS[45]

The server is called whenever a HTTP request goes into it, which makes it perfect for a distributed setup where HTTP is thought to be the backbone of communication.

Furthermore MongoDB is directly supported by NodeJS and can be installed relatively easy, this also argues for NodeJS as a good choice for running the server.

Digital Ocean

Last but not least a host for the server is needed. There exists several services providing web hosting, but for this project Digital Ocean has been chosen. This because it offers a service called droplets, where a virtual server can be created with a few clicks, and is automatically setup with the desired operating system[46]. Furthermore it exposes a public IP address, so it can be called from anywhere without dodging around with different hacks and difficult setups.
The choice of Digital ocean for running the web services has not been undergoing and analysis and decided upon that, it was chosen only because of the easy setup and manage. It might not be the best choice if undergoing an extensive analysis, but for this project it fits its purpose.

By this the choice of technologies in this project will be concluded. The relevant technologies in use has been described in general. More details about how they have been implemented and where they have been used will be described in the implementation section of this report.

4. Analysis

In this chapter the focus will be on analyzing how a system as imagined in the first chapter, could be designed. The aim is to be able to create a system that can help raise the security level for IoT home devices, without influencing or modifying the hardware and software of the device. The analysis will be based on the findings in the Background and State of The Art chapter. More specifically the scenarios of IoT setups build from real IoT home products. The outcome of the analysis will be a requirements specification for the IoT security system, which will be used to create the system design and thus the implementation of the system.

4.1 Preface

Before going into depth with the solutions of the IoT security system, it is important to set a basis for the analysis, in order to make sure to focus on the right and technical feasible solutions. This section will try sort out the sheeps from the goats, in that way the remainder of the analysis will only focus of the relevant solutions for such a system.

4.1.1 Deducting Scenarios

As a beginning we will take a look at the three scenarios described in chapter 3, State of the Art. Two of the scenarios, namely A and B are constructed in away where their connection will be, either directly or indirectly connected to a home router and through that to an internet connection. Thus making it safe to assume that the way these devices connect would be rather static, always connected to the same router and the same internet connection. This also makes it possible to setup an intermediary or gateway that can help protect the IoT devices setup in the home, and thus there will be something to work with. On the other hand, looking at scenario C (as depicted in figure 11 below), which interacts through a smartphone and a dedicated app is harder to cope with. As a smartphone has many other functionalities than acting as a gateway for an IoT device, there will be a lot of connections to and from the phone, furthermore the phone works in a more dynamic environment, where it will connect through different means, such as the cellular connection and WiFi connections to different access points.

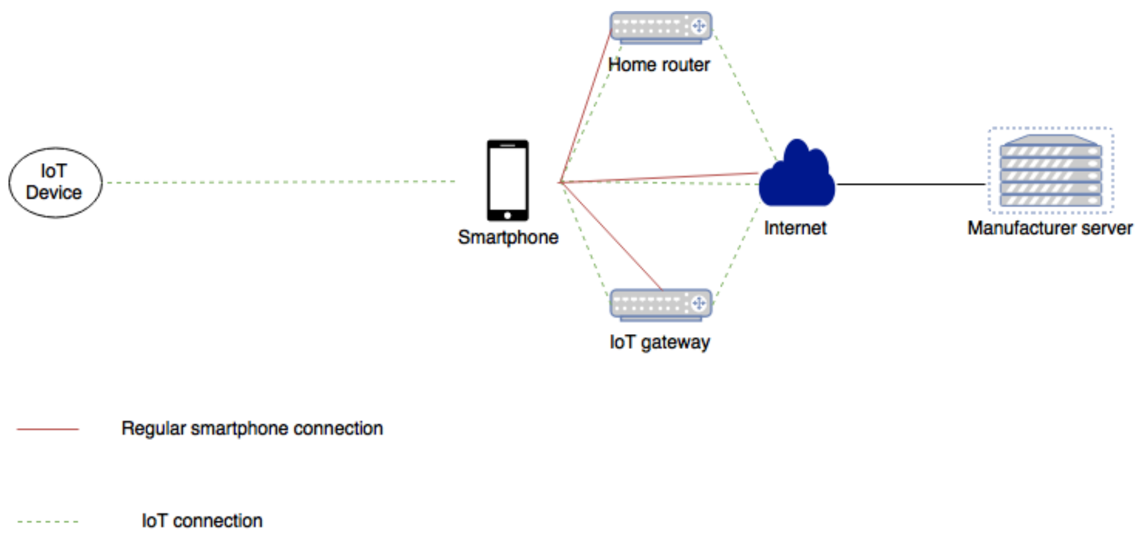


Figure 11: Scenario C with different connection types and routes

Looking at figure 11, some different type of connections for at smartphone is visualized. To be able to raise the security for IoT devices with the help of an intermediary, in this picture called IoT gateway, it would require the phone only to connect through that gateway, and not the home router or directly to the cellular network.
This is believed not to be a feasible solution, as it will limit the phone owner's possibilities to use the phone, furthermore many phones connects to the internet via the best available connection, meaning that a bad connection through the IoT gateway could lead the phone to connect directly through the cellular network or another WiFi connection.
Even though the phone would be setup to connect only through the IoT gateway, a smartphone has a huge amount of connections, as it is not only used to control IoT devices but anything from web browsing to gaming.
In figure 11 the IoT device's data stream is depicted as a green dashed line an all other connections with a red line. To let an IoT gateway be able to distinguish between these two types of connection will be almost impossible. Just trying to do it, will possibly lead to a huge amount of false indications of security breaches, and thus annoy the user rather than help him/her.
By the above arguments, scenario C, where the connection from the IoT devices goes through a smartphone to connect to the internet, will not be a part of the solution, as it is believed to do very little good, and there neither be relevant in the rest of the analysis.

4.1.2 Protocols and frameworks

Another important consideration in the design of this solution is the idea of implementing it, without access to the manufacturers hardware, software design, and ways of implementation. Going through the literature theres exists several protocols and framework proposals that should be able to raise the security level within IoT networks, however, all of these requires the devices to have implemented such protocols, or at least access to implement them, which is not the focus of this thesis. Furthermore, as stated in the related research, heterogeneity of IoT devices and networks is still a huge issue, meaning that even though some devices might have implemented security features, which of course would be a good thing, the chance is that many other devices in a user's home might not. Therefore implementation of different security protocols or frameworks directly to the IoT devices will not be a part of the proposed solution and therefore not the analysis as such.

4.2 Overview

Having established what to focus on, and what not, in the analysis of the proposed system, an overview is presented, in order to ease the understanding of how the system could be designed and what relevant entities such the system consists of.
Figure 12 below, shows a high level overview of how a system to raise the level of IoT security devices in a home environment.

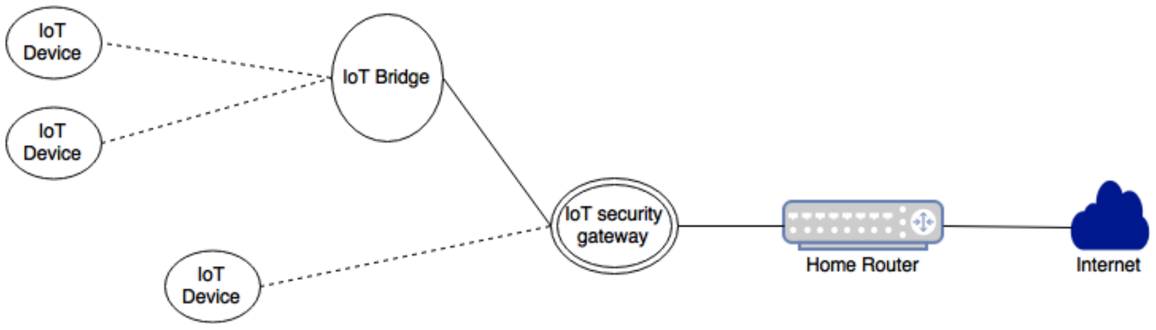


Figure 12: High-level overview of an IoT security system

The relevant scenarios from State of the Art, have been combined and an intermediary called IoT security gateway has been introduced between the home router and IoT devices/IoT bridges.

The reasoning behind the IoT security gateway comes with different arguments, one of them is from 3.3.2, where it has been shown how a network edge device, can be introduced to help manage the security risks for IoT devices, however the setup shown by Kuusijärvi et. al. was heavy and not as such appropriate for much simpler home setups. Another argument is that this setup should be able to raise the security without having to modify software and hardware in the actual IoT devices, therefore, this cannot be done without placing an intermediary as shown in figure 12.

By the placement of the IoT security gateway, the network traffic both to and from the IoT devices will go through this device and thus it will be possible analyze some of the data going through the network, and by implementing the right mechanisms raise the level of security.
In figure 13 below, an overview of the data flow between entities in the system, and the function of the IoT security gateway.

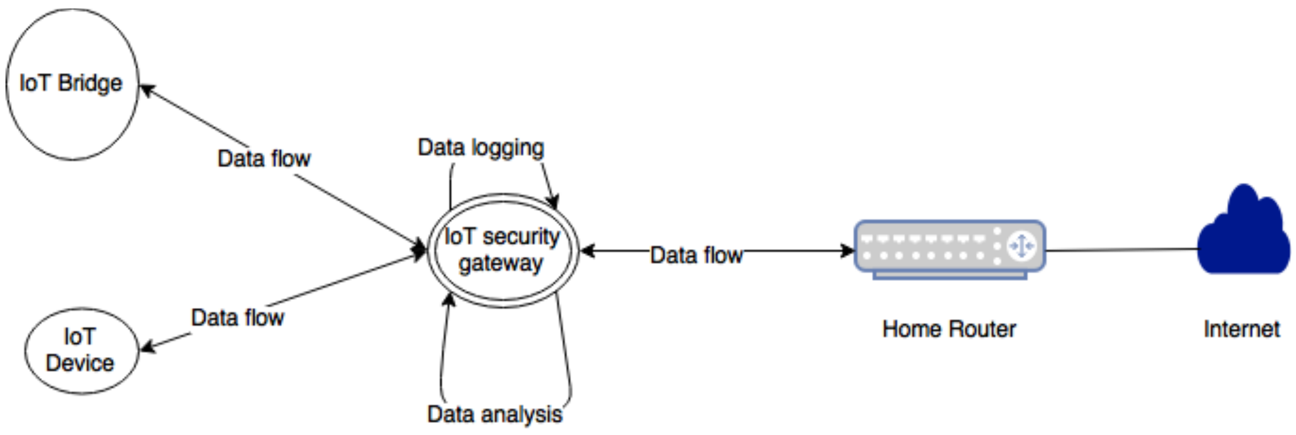


Figure 13: Data flow of different entities in the system and function of the gateway

As it is depicted the IoT security gateway should have the functionality to log the data flow and analyze the logged data with the goal of discover and act upon possible security breaches or misuse. As the security gateway should be able to work, no matter what kind of IoT device connected to it, it is not feasible to let it analyze the actual payload of the data packets transmitted, as this will differ from device to device, might be in proprietary formats, encrypted or just impossible to understand for others than the designers of the given device.
By this the only possibility for the gateway is to log data that makes sense to analyze, in this case it will be the metadata from the data transferred in the network, and subsequently try to use this data to interpret what is going on in the network.

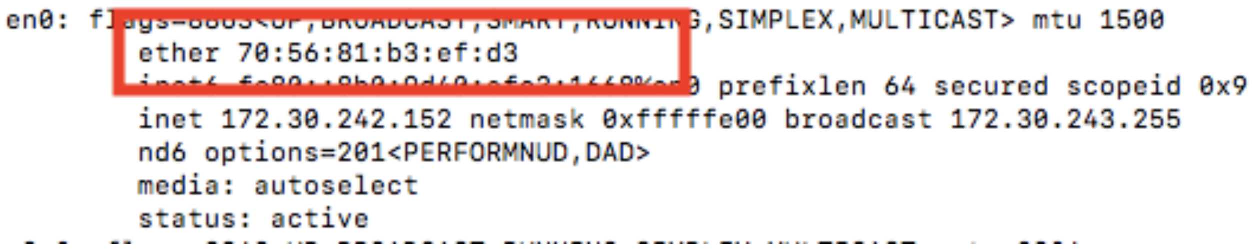
4.3 Meta Data

As mentioned above meta data is going to play a very important role in this system, as this is the only transmitted data that is in a standardized format, and therefore easily can be interpreted.
By definition metadata is: "*A set of data that describes and gives information about other data.*"[47]. So instead of interpreting the actual payload of the transmission, the aim is to interpret data about the payload(data about the data). The aim of this section is too look into the kind of metadata that is possible and relevant to extract from the network communication flowing through the IoT security gateway, as all communication to and from the IoT devices goes through the gateway.

In the preceding section an overview of the system is depicted in a high level, from this overview it is clear that the IoT devices in the system in one or another way will communicate through the internet. This means that the hardware whether it is the IoT device itself or a bridge mediating the communication must have some networking hardware able to connect by the means of regular network protocols such as ethernet or WiFi. Furthermore the communication is assumed to go through the Internet Protocol[48] which will be discussed later, however it is important to note, that the Internet Protocol (IP) in its current use exists in a version 4 and version 6 or IPv4 and IPv6 respectively. Regardless IPv6 is the newer and the most powerful of the two, the protocol is still not adopted widely around the world. By Google IPv6 statistics[49] it is shown that in Denmark the penetration of IPv6 is only 2,24%, therefore in this analysis only IPv4 will be a target, as this is still the most widely used technology.

4.3.1 Hardware

Looking at the hardware devices connected to the IoT security gateway, it will use either the ethernet of WiFi standard, which both are of the IEEE 802 family. This means that both of the technologies have implemented a MAC address, which is a unique hardware address assigned to the network interface of the device. From IEEE it is described: "*All MAC protocol data units contain addressing information. The addressing information consists of two fields: the destination MAC address and the source MAC address.*"[50]. The Mac address will always be a part of the transmission inside the local area network (LAN), to which the device is connected. This provides useful information as it will always be possible to know the physical source and destination of a packet transmitted inside the network.



```
en0: fl[illegible]G,SIMPLEX,MULTICAST> mtu 1500
        ether 70:56:81:b3:ef:d3
        [illegible]0 prefixlen 64 secured scopeid 0x9
        inet 172.30.242.152 netmask 0xfffffe00 broadcast 172.30.243.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

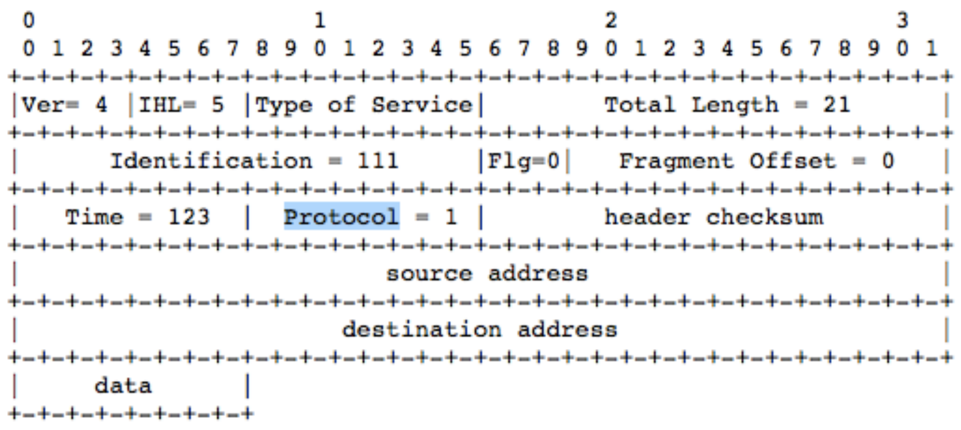
Figure 14: Example of a MAC address for the WiFi controller in Mac OS X

As the MAC address is fixed and bound to the actual physical device, differing from the IP address, which change depending on the network connected to, the MAC address is a way to uniquely identify which entities are communicating which each other, and thus very helpful analyze the communication within the network

4.3.2 Internet Protocol

Another important way extract meta data, is by the help of the internet protocol, as the IP protocol specifies the format of how data is transported through the internet from the source to the destination. From the standard of IPv4[51] it is described that: " *The internet protocol implements two basic functions: addressing and fragmentation.*" Which means

the protocol is able to address the destination of a datagram send from any given source, furthermore it is able to fragment data packets into smaller peaces to transfer them through the internet and reassemble the packets at its destination.
Figure 15 depicts and example of how an internet datagram looks like by the IEEE standard. As it has been established, this system is not aiming at analyzing the data, or payload in the transmitted packets, therefore the interesting parts of the datagram is the information elicited from some the fields not containing the data, also specified as the header[52].



```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |Ver= 4 |IHL= 5 |Type of Service|        Total Length = 21      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      Identification = 111     |Flg=0|   Fragment Offset = 0   |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Time = 123  |  Protocol = 1 |        header checksum        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         source address                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                      destination address                      |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     data      |
 +-+-+-+-+-+-+-+-+
```

Figure 15: Example Internet Datagram[51]

From the header it is possible to elicit different types of information that is of interest to IoT security gateway, not all of them will be used, but some might be interesting in order to analyze what is going on in the network, and thus trying to interpret if some of the IoT devices are working in a malicious way. Therefore it makes sense to go through the fields to investigate which of them carrying relevant information.

- Ver = Version: As it has already been established this will focus on version 4 of the IP, therefore it is also assumed that all packets uses version 4
- IHL = Internet Header Length: Informs about the length of the header to be able to see where the payload starts. As this is merely for the receiver to know where the actual payload starts it it not relevant to the internet security gateway.
- Type of service: Indicates what quality of service the sender desires through the network to the destination. This is used throughout the network if supported to decide the route to the destination. This is not of interest to the gateway as it does not give any information about the what kind of data that might be transmitted or to whom.
- Total Length: The total length of the datagram about to be transmitted. This might be of interest to the IoT security gateway as the length of the datagram can be relevant information if it is combined with knowledge of the type of service the device transmitting the datagram is providing.
- Identification: Used to help assemble fragmented datagrams. This is of no relevance to the IoT security gateway.

- Flags: Also used to help assembling the datagram, therefore of no interest to the gateway
- Fragment offset: Help to assemble fragments of the datagram, of no interest to the gateway
- Time to Live: Decides how long the datagram lives in the network, from source to destination, of no interest to the gateway
- Protocol: Tells what protocol is used in the next level of the network model (eg. TCP/UDP), this can be of interest if combined with information about the services the sending devices provides.
- Header checksum: Checksum to be computed by each hop in the network, therefore not relevant to the gateway.
- Source Address: The source IP address of the data packet, this is highly relevant as it with the aid of other services can be determined whether the sender of the packet is legitimate
- Destination address: As with the source address the destination address can be also help to decide the legitimacy of the destination.

The points shows what can be gathered from the IP header, and what of the data that could be relevant to the design of the IoT security gateway, the relevant data will be elaborated further in the coming chapters

4.3.3 Various Meta data

Besides the predefined meta data from different protocols, it is also possible to gather other types of data, that could be relevant to the gateway and analysis of the data transmitted through it, however this is metadata is not standardized from protocols or similar, and therefore requires work to be identified, these types of data will be discussed below.

Timestamps
It can be of high relevance to know when devices transmits and receives data, this knowledge combined with the type of device can help to identify if the transmitted data is legitimate or not. Take an example of IoT light bulbs, that can be controlled from a smartphone app or similar. A usual pattern of controlling light bulbs in one's home would be when one is actually home, and the sun is set, usually not in broad daylight. Naturally there can be situations, where this is not the case, if there is a poorly lit room in the house, or one wants it to look like there is somebody home to discourage burglars. However the usual pattern can be assumed to be during night time, or early mornings. If such a device suddenly starts to transmit or receive a lot of data outside the timeframe it is supposed to be used, it could be an indication of a problem or a malicious takeover. Thus it makes sense to know when a devices is transmitting or receiving data, and timestamps is way to

mediate this, simply by letting the IoT security gateway log the time for every packet send or received through the gateway.

Location

The location of the data packets transmitted and received can also be a way to discover if something is wrong. As already described it is highly relevant to know the destination and source IP addresses of the datagrams sent, these IP addresses, can amongst other things be used to locate where a communicating party is located. Naturally the location of the IoT device will always be known as it is in the owner's position, however this is not the case for the parties the device is communicating with.
Assume an American made product by Google, it will likely communicate with Googles servers, maybe in America or with a Google data center it Europe. However if the ´product suddenly starts to transmit data to, or receive data from a server in China or Russia there might be something wrong. Of course it is not certain, Google might have decided to reroute the traffic to that location, but it should definitely raise a suspicion if everything is right. By the help of the IP address in the packets transmitted, and an IP location service such as keycdn[53] it is possible to connect an IP address to a location, and thus detect whether the location of a communicating party could be suspicious.

Amount of data

One thing is the data transmitted to and from a device, another thing the amount of data transmitted. Depending on the service of the IoT device, the amount of data transmitted can be helpful to determine if the device is only acting as supposed to. If a device, only designed transmit relatively small amount of data this could be commands such as "turn on", "turn off", "set value to x" etc., suddenly starts to transmit a huge amount of data this can be an indication of something not right, thus it makes sense to log the amount of data a device transmits or receives.

All of the metadata presented in this section, in combination with the type of IoT device and the device's services, can be used to help analyze if there is a security threat to the devices, or if a device is misbehaving and pose a security threat to others. Therefore the data is highly relevant to log to the IoT security gateway in order to help analyze the network traffic.

4.4. Gateway - Logging

By the help of meta data it becomes possible to gather information about transmissions in the network without analyzing the actual payload transmitted. By this it becomes clear that the IoT security gateway should be able to gather the data described in the preceding section. Thus a high level setup of the gateway would look like depicted in figure 16

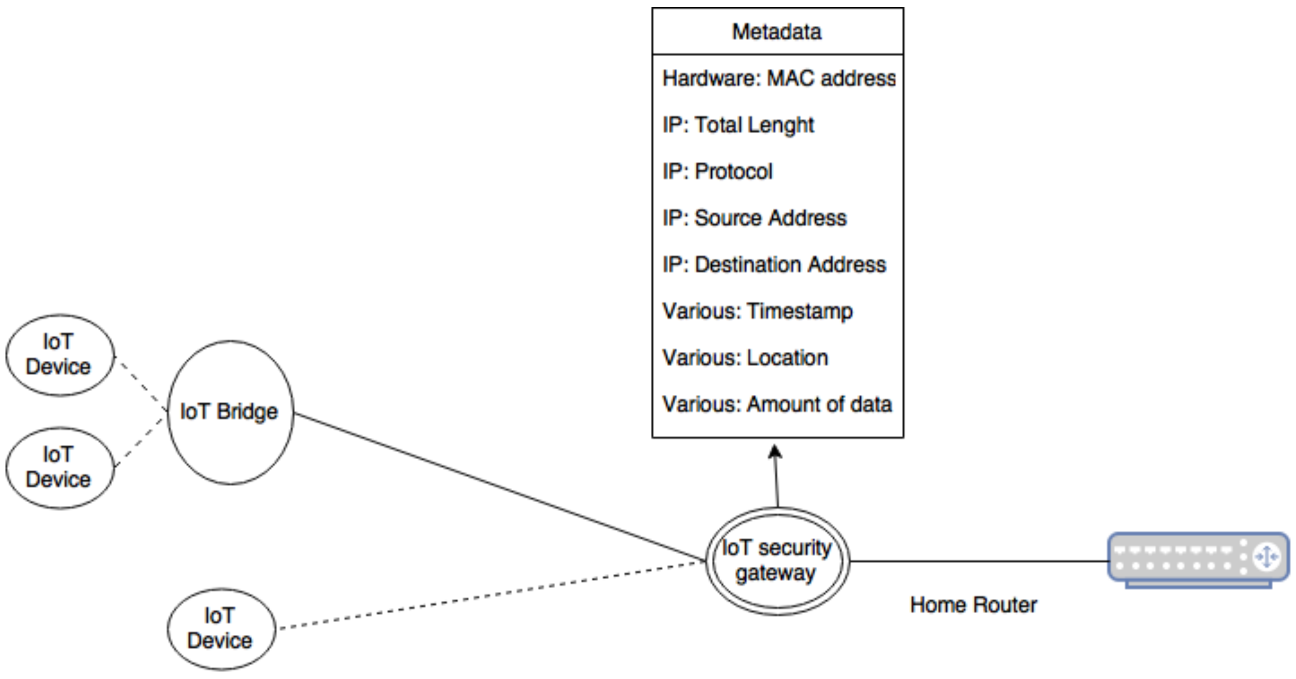


Figure 16: Overview of data logged by the IoT security gateway.

The gateway simply has to log all of the data depicted in the figure. The logging must be from both sides of the network, meaning it most log both the data transmitted from IoT devices and the data received from the home router. The logged data should then be analyzed in order to decide whether an IoT device is acting malicious or is under attack from an outside perpetrator. Therefore the gateway needs not only to log the data, but also interpret the logged data. The interpretation of the data can be done either by the gateway itself or by a third party connected to the gateway, only letting the gateway be responsible for the logging and the transmission of the data.
A setup with a third party could have several advantages, such as computational power, the possibility to share information with other gateways, if in example a malicious IP has been discovered. The more detailed setup will be discussed in the chapter System design.

4.5 Security Mechanisms

In order to be able to used the logged metadata to anything it is evident to look at the security mechanisms in such a system. This chapter is going to look into different aspects of security in order to create the right mechanisms to design a setup where the IoT security gateway will be able to raise the level of security for the devices connected to it. In order to do this, different risks and threats will be analyzed, and security objectives for the system will be identified. This will be done in accordance to data available to the system and the limitations of the system such as the lack of possibilities to implement better protocols or change the design of the IoT devices. The goal of the chapter is to end with a set of objectives that can be implemented in the IoT security gateway, to aid the requirements specification and design of the IoT security gateway.

4.5.1. Risks and Threats

The first phase of this chapter is to identify some of the risks and threats IoT devices are exposed to. To do this the background chapter will be revisited, in order to get an overview of the known problems within the world of IoT. From chapter 3.2 it is shown that for this project three parts are highly relevant to look into, namely; communications, devices and actuators. In the communication it is relevant to look into if the channel used by the devices is secured, For the devices, it is interesting to look into the capability of the IoT device, and thus the risks and threats it exposes. Lastly the actuators are interesting, who are they and what are their possibilities exploit the IoT devices.
From the background, different risks and threats have been described by the help of different research papers and known attacks. In the following these findings will be grouped into where they belong; devices, communications and actuators. It is important to note that not all findings will be described as some of them are overlapping, regardless it will provide and overview of the IoT devices vulnerabilities.

Risk/Threat	**Device**	**Communication**	**Actuator**
Weak device security	X		
Lack of software updates	X		X
Cloud service storing data			X
Malware	X		X
Confidentiality	X	X	X
Integrity	X	X	X
Availability	X		X
Privacy	X	X	X
Identity Management	X		X

Table 1: Risk/Threats for IoT devices

In table 1 above, the most outstanding risks and threats to IoT devices, has been listed according to the group it belongs to, which provides a good overview, however to get the understanding of the plotting the arguments will be listed below:

Weak device security: Lack of security in the physical IoT device, such as hardcoded passwords and lack of ability to encrypt data.

Lack of software updates: Leaving the IoT device exposed to attacks if security vulnerabilities in the software are not updated. The same could happen on the actuator side, e.g., lack of ability to update server side software.

Cloud services storing data: Lack of the ability to protect data on the actuators side could lead to loss of data and privacy.

Malware: If not protected correctly the IoT device can be exposed to malware.

Confidentiality: Lack of encryption in either hardware/actuator side or in the communication channel can lead to loss of confidentiality.

Integrity: Lack of possibilities to sign data in the device or lack of security in the communication channel or actuators side of the network can lead to lack of data integrity.

Availability: If not secured correctly, the IoT device can be made unavailable by e.g. a DOS attack. Furthermore the device itself can be infected, and be a part of a DDOS (distributed denial of service) attack, either inside the local area network connected the device is connected to, or to outside services.

Privacy: If not protected at both the device, in the communication channel and at the actuator, data can be stolen and lead to a loss of privacy

Identity Management: If not handled on both the device and at the actuator, personal information might be shared with unauthorized persons.

With the above overview of risks and threats and what entities they belong to in an IoT system, there is need to structure the elements in a reasonable way to be able to identify how the IoT security gateway should be able to mitigate some of the risks and threats IoT systems are exposed to.
In order to do this, the NIST Framework presented in chapter 3.2.3. will be used, in order to create a better understanding of the objectives that has to be accomplished in order to raise the security by the help of an IoT security gateway.
The framework will be used in the following way: The already identified threats and risk will be broken down to sub categories, that defines the implications of the each risk or threat. From the breakdown different protection and detection methods will be defined, in combination with at proper respond, if there is a need to do a recovery of the system or some of the devices, the appropriate action will be stated.
Furthermore the functions “Protect” and “Detect” might be interchanged in some cases, as situations where detection is needed before it is possible to protect the device.

Identify	
Categories	**Sub categories**
Weak device security	Lack of encryption Low complexity/hardcoded passwords Altering of data - loss of integrity
Lack of software updates	Exploration of security vulnerabilities on old software
Cloud service storing data	Lack of encryption Altering of data - loss of integrity Stealing data - loss of privacy
Malware	Stealing data - loss of privacy Altering data - loss of integrity Botnet - used to DoS attacks Ransomware - encrypting data Attack other devices in the local area network
Confidentiality	Lack of encryption Stealing data - loss of privacy
Integrity	Altering data Ransomware - Encrypting data
Availability	DOS attack towards to device DDOS attack from the device controlled by a botnet
Privacy	Lack of encryption - stealing data
Identity Management	Unauthorized access to the device

Table 2: Identification of risks and threats

By the help of the breakdown presented in table 2, it is possible to take a look into how the different categories can be protected and detected. This will be done with respect to the possibilities the proposed IoT security gateways possess. Meaning that only the meta data it collects can be used. Therefore mechanisms such as implementing encryption on the devices, making cloud storage more secure and other things demanding a change in the design of the IoT devices and respective communication parties will not be proposed.

Protect	
Categories	**Sub categories**
Lack of encryption	Block transmissions through unsecured protocols
Low complexity passwords	Urge the user to change password if possible
Altering data	N/A
Lack of software updates	N/A
Stealing data	Block transmissions through unsecured protocols Block transmission to unknown hosts
Botnet	Block transmissions to/from suspicious IP addresses
Ransomware	N/A
Attack other devices in the local area network	Block communication with other devices in the network
DDoS/DoS to from the device	Block transmissions to/from suspicious IP addresses
Unauthorized access to device	Block transmissions to/from suspicious IP addresses

Table 3: Protection of IoT devices

In table 3 presented above, the methods to protect an IoT by the means of the IoT security gateway is presented, the sub categories marked with N/A means that the vulnerability cannot be mediated by the possibilities the proposed gateway implements. From the table the protection by just blocking the communication that constitutes potential risks seem trivial. However that is far from the case, as one has to known what is a potential risk and what not. Of course there might be certain cases where it is obvious, but besides these cases, it requires a well considered logic to handle. Therefore the detection of possible of security incidents plays as an important role as the protection.

The following table will present the detection mechanisms of the system and how to make a logic of what the system should analyze and thus be able not only to detect incidents, but also help to protect the IoT devices.

Detect	
Categories	**Sub categories**
Lack of encryption	Log transmission protocol
Low complexity passwords	N/A
Altering data	Log transmission IP source and destination Log timestamp
Lack of software updates	N/A
Stealing data	Log transmission protocol Log transmission IP source and destination Log timestamp
Botnet	Log transmission IP source and destination Log timestamp
Ransomware	N/A
Attack other devices in the local area network	Log transmission MAC address
DDoS/DoS to from the device	Log transmission IP source and destination Log transmission total length Log amount of data Log timestamp
Unauthorized access to device	Log transmission IP source and destination Logt timestamp

Table 4: Detection mechanisms of IoT security gateway

From table 4 above, different methods to detect security methods are shown, categories and sub categories are defined by the knowledge from different attack types, and the meta data the IoT security gateway should be able to log.

4.5.2 Logic

To be able to create a meaningful detection, and thus protection of the IoT devices connected to the security gateway, there is a need for a logic helping the gateway to be able to detect whether a transmission is desired or not. The following section will focus on how this logic should work with respect to the objectives defined in the preceding section. The goal is to be able to combine the metadata provided by the gateway with knowledge of the IoT device's services and usage, and let the output help the gateway to detect and protect against security incidents.

In order to build this logic, it is needed to look into some different factors, namely what is known, what is expected and what falls outside either the known or expected rules for the given IoT device. Depending on the service the IoT device offers known and expected factors might differ. In example an IoT-light bulb should transmit less data than a surveillance camera offering a video feed. Therefore it is not possible to create one true logic that can be applied to all connected devices, and different rules will apply depending on the of interest.

The known factors naturally will be defined by the metadata extracted by the IoT gateway, whereas the expected factors can be defined by knowledge of the devices and informing the gateway about expectations, in other ways, training the gateway, either by the help of the user or by previous knowledge from similar devices.

In order to design such a logic, it makes sense to take a real life case, as an example the Philips Hue light bulb, described in State of the Art. This device can be described as a "command and control" device, where the primary function of the connectivity build into the device is to be able to control it, and let the device respond to the control commands send to it. This means that the amount and the size of the data packets transmitted is expected to be rather low. As the devices of interest are light bulbs, it can also be expected that they are only communicating in certain times during the day; when it is dark outside and people are awake, depending on location and time of the year, this timespan of usage might differ, however, it is possible to define a span in which the bulbs are expected to communicate. Also the setup might be communicating through a manufacturer controlled web service, making it possible to expect a specific destination for the commutation, or if not a location to where the communication is going (e.g., Denmark, Europe, U.S. etc).

This kind of interpretation of the metadata that is logged can help to build a logic at the gateway, to let it be able to decide whether the communication going through it is expected/allowed or if it is unexpected and might pose a security risk.

Type of device: Light bulb		
Meta data	**Expected**	**Unexpected**
IP: Total lenght	Short	Long
Various: Amount of data	Low	High
IP: Source address received packet	Known address	Unknown address
IP: Destination address transmitted packet	Known	Unknown
Various: Timestamp	[07:00-10:00] - [17:00-23:00]	Outside expected
Various: Location	Nothern Europe	Anywhere else

Table 5: Overview of interpretation of metadata in the IoT security gateway

Table 5 above, presents an overview of how an interpretation of the communication transmitted by an IoT light bulb/light bulb bridge could be set up. Naturally for a computer system to understand this, the rules needs to be specific, e.g. the amount of data needs to be a fixed number, or set to an exact threshold, however this relies of some real life testing, and therefore is not possible to define in this part of the thesis.

The interpretation of such a rule set can be set to different levels of acceptance. So the user gets a higher level of control, depending on his/hers preferences. A proposal of such settings could be:

- Meta data: All - Unexpected behavior: Notify user:
- Meta data: Location: Unexpected behavior: Block communication until further notice

This could also be interpreted as the respond to occurred security incidents, as seen in the NIST framework. Depending on the type of device and the occurrence of the unexpected event, different responses can be invoked. In all cases it makes sense to notify the user of the IoT devices, so he/she has a possibility to react on the event. However depending on the service of the device and the importance of the service to the user, an actual action, such as blocking for communication must depend on the user's preferences as there can be differences in valued the service is to the user, thus the user might want to risk the possibility of a security breach to keep the service available. Therefore the responds to security incidents has to be a distinguished by both the user and the type of service the IoT devices offers.

4.5.3 Policies

To be able to build the logic discussed in the preceding section, a set of rules is needed, better described as policies by Kuusijärvi et.al.[54]. These policies will the backbone of the system in order to protect the system and detect security incidents, thus they have be considered very carefully. To describe these policies a top down approach will be taken, using the information already known from the Background, State of the Art and preceding chapters of the analysis.

It is clear that IoT devices in a home can take different forms and offers very different services, by this it makes sense that each device, or at least each type of device should invoke its own policy, meaning that the same set of policies does not necessarily apply to all devices connected in the network, as a light bulb or thermostat offers a very different service than a webcam. Therefore there is a need to identify each device, which can be done by the unique MAC address all devices are equipped with. Even though all devices needs a unique set of policies, some devices in the network might be the same or be very similar in its services, in example "command and control" devices, where the functionality is similar. Consider two set of light bulbs where one set is placed somewhere in the household only used in the evening, and the other set placed somewhere used both early morning and in the evening. From table 5, a lot of the expected behavior can be identical, however the timespan of expected use might differ. In the light of such a setup it makes sense not only to define individual policies, but also a group policy where the basic settings are the same, but with individual preferences depending on the IoT device.

The actual policy of course needs be setup with respect to the gathered meta data described in section 4.3, but there are differences in the way it makes sense act on different metadata. In example the IP protocol metadata is determined by the design of the given IoT device, if it transmits through UDP or TCP. This cannot be changed by the security gateway, therefore it does not makes sense to create a policy that blocks a transmissions through a given protocol, yet it is considered to be important to let the user know whether a devices transfers data through an insecure protocol, even though it cannot be changed, it gives the user the possibility to decide if, and to what he/she wants to use the connected device.
On the other hand there will be situations where it makes sense not only to notify a user, but also block transmissions to and from device, this could be in the case of transmissions from an unknown source located in a place known for problems with hacking, or if the amount of transmitted data rises to unusual amount. Therefore the policy should have the possibility both to act not only by notifying a user, but also to block connections in desired cases.
Besides the way a policy can be setup up and how it enforces breaches of the rules, a policy needs the information about what is expected or desired behavior and what is unexpected/undesired behavior. Depending on the type of meta data, the way this data is represented differs, from IP addresses to protocol names. This means that the system needs to be able to interpret the different data present in a policy, in example a policy could allow communication with only one specific IP address, in that case the gateway needs to be able to compare if the policy IP corresponds to the transmitted destination IP address. On the other hand, a policy could also define a maximum amount of data packets transferred within a given time period, in this case the gateway need not only to compare values, but count the number of packets transmitted within in a certain timespan and hold the value up to definition in the policy.
Therefore in general, the gateway should be able to enforce a policy depending on the type of metadata and the definition of the policy. How this enforcement should work will be described in the System Design.

The way to design a policy could be done in several ways, but in this case it makes sense to work with two different methods, namely black- and whitelists, which means that for a given IoT device and corresponding policy there should be either a black or whitelist for each set of metadata.
A blacklist, as the name suggests would be a set of actions/communication that is not allowed, anything else will be accepted. Whereas a whitelist would be a set of transmissions/actions allowed, everything else is disregarded.

Whether to work with a black- or whitelist is highly depended on the IoT device, and the knowledge of it. An example could be a device which is known always to communicate with the same server on the internet and nobody else. In this case it does not make sense to create a list of a huge amounts of IP not allowed, that has to be gone through for each transmission. Instead it makes sense to create a small list (in this example only one), which is checked, and if it matches it is allowed. On the other hand if the device is communicating with various different servers and services considered legit, but it has been discovered that communication with a certain server might be insecure, it makes sense to create a blacklist where the address of the insecure server(s) is stored, again so the gateway only has to check a small list for each transmission. In this way it makes sense to let the policies implement black- and whitelists as a mean of enforcing a given policy.

4.5.4 Training

In order for the IoT security gateway to work, naturally it needs to have policies to enforce, and the policies needs to be meaningful in order to be able to protect and detect security incidents. As the idea of the gateway is that it should be possible to raise the level of security no matter the IoT device connected to it, at list of preset policies will not be an efficient way to solve the problem. Of course the gateway could have some presets with blacklists of known malicious IP addresses and locations or similar, however this is not enough. As the devices can offer very different services and have different ways of communicating with the outside world, the gateway needs to be taught what is accepted and what is not. Therefore the gateway should employ a way of learning what is acceptable behavior for a given device.
This means that each time a new device, not known to the gateway is connected, a training period should be setup, in where the gateway will prompt the user when the device engage in activities. Naturally such prompts should be understandable by the user, meaning the gateway should not ask: “Your device is communicating with IP address: xyz.zyx.y.xz”, but rather: “Your device is communicating with a server in location Y, is this plausible?”. In the same way other questions could be asked, such as: “Your device transmits a lot of data, is it expected to do so?” And “Somebody is interacting with your device, was it you”? By the user’s respond it will be possible to let the gateway learn the acceptable rules for the policies, and thus be able to secure the IoT devices.

Even though a training period is needed in order for the IoT gateway to learn how policies should be setup, this can be a heavy task, therefore it makes sense to look into ways to ease the setup.
A way to do this is to introduce a policy repository as described in the related research. The policy repository should contain all policies for all known IoT devices connected to an IoT gateway, and it should be possible for the gateway to report incidents to the policy repository, so policies can be updated over time, when new knowledge gained. In order to

make such a repository really useful, it should be a shared resource between all users of the IoT gateway, this have two explicit benefits.

- It is same to assume that some user's have the same IoT devices connected, with a policy repository users with new IoT devices can ease the training period if knowledge about policies for the given device already exists in a central location, and can be fetched by the IoT gateway.
- Security incidents discovered by a single gateway can be sent to the policy repository and thus shared between users of the same type of IoT device.

By this, it makes sense, not only to let a policy repository be a part of the system, but let it be an external resource that can be shared amongst all users of the IoT security gateway, as it will ease the setup of policies and in the same time make incident respond faster as similar devices can share newly gained knowledge.

4.5.5 User Interaction

The last thing to look into is the user interaction, as it has become clear, both for the setup of the device and training period, as well as for the need of notifying users about security incidents, the system should be able to interact with the user.
Such a setup can be designed in several ways, however in it should be accessible to the user no matter where he/she is. As many private internet connections does not have a public IP address, letting the user interface run on the gateway itself will be problematic, as it requires additional configuration to make it accessible outside the local area network. Furthermore, as it has been argued that an external policy repository needs to be setup, it makes sense to let the user interface run as an external web service (possibly in combination with the policy repository), and let this web application be the link between the IoT security gateway and the user.

By the above analysis from including everything from the relevant logging data, how to protect the IoT devices and from what, to how this is possible and how the user will interact with the IoT gateway, the analysis part of this thesis will be concluded with an overview of how the setup will look from the knowledge gained (figure 17).

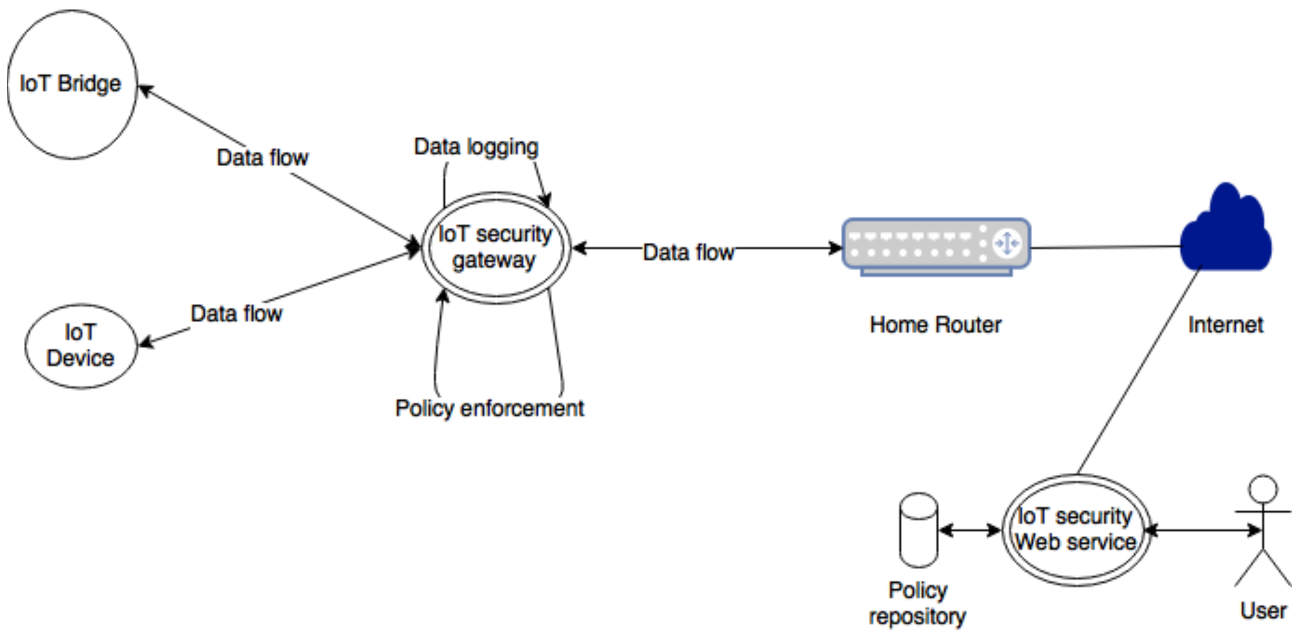


Figure 17: Overview of entities in the IoT security gateway system

All of the revelations in this chapter makes it possible to start defining the specific requirements for the systems, which will be the basis of how to design the system. The coming chapter will focus on defining all relevant requirements for the system.

4.6 Requirements

The coming chapter will primarily focus off the requirements for IoT security gateway system. The requirements elicited will be for all parts of the system as depicted in figure 17 in the preceding chapter. This means that it will not only focus on the gateway itself bu also on the other relevant parts of the system.
The requirement specification will be presented int the following way, each part of the system will have its own section where each requirement will have an ID and a specification. The requirements will be divided into Functional and Non functional where the functional requirements will be presented below, and the non functional requirements can be found in appendix B.
Often a requirement specification will present a prioritization of the all the requirements, however in this case all requirements are considered essential for the system to work as desired, therefore all of them can be considered a "must have".

Gateway	
Requirement ID	Specification
GWFR1	The gateway shall be able to log defined metadata (MAC address, IP: Total length, IP protocol, IP source address, IP destination address, Timestamp, Amount of data, location)
GWFR2	The gateway shall identify all connected IoT Devices
GWFR3	The gateway shall connect to a external service to be able to: Communicate with users and to exchange information
GWFR4	The gateway shall be able to fetch policies from the policy repository
GWFR5	The gateway shall be able to enforce policies
GWFR6	The gateway shall employ a training period for new devices
GWFR7	The gateway shall enforce policy breaches in correspondence with the most recent policy from the policy repository
GWFR8	The gateway shall report security incidents (policy breaches) to a web app
GWFR9	The gateway shall be able to block transmissions if the policy requires it
GWFR10	The gateway shall be able to associate an IP address with a location

Table 6: Functional requirements for the IoT security gateway

Policy	
Requirement ID	Specificiation
PFR1	A policy shall contain rules about all metadata specified in GWFR1
PFR2	A policy shall be identifiable to a specific device
PFR3	A policy shall be able to contain a black/whitelist to relevant metadata
PFR4	A policy whitelist must be enforced before a policy blacklist
PFR5	A policy shall be updatable
PFR6	A policy shall contain information about how to enforce it (block transmission or notify user)
PFR7	A policy shall be readable to both the gateway and the policy repository
PFR8	A policy can belong to a group

Table 7: Functional requirements for the policies

Policy Repository	
Requirement ID	Specification
PRFR1	The policy Repository (PR) shall contain all policies
PRFR2	The PR shall expose policies upon request
PRFR3	The PR shall be able to identify a policy upon request
PRFR4	The PR shall be open to policy updates
PRFR5	The PR shall update policies only if requested by the user
PRFR6	The PR shall be able to suggest updates to policies for devices of the same type
PRFR7	

Table 8: Functional requirements for the Policy Repository

Web application	
Requirement ID	Specification
WAFR1	The Web Application (WA) shall connect be able to connect to the gateway
WAFR2	The WA shall be able to connect to the PR
WAFR3	The WA shall be able to communicate with the user through a user interface
WAFR4	The WA shall be open to requests from the gateway
WAFR5	The WA shall be able to fetch policies from the PR
WAFR6	The WA shall be able to transmit the policies to the gateway
WAFR7	The WA shall be able to receive information about policy breaches from the gateway
WAFR8	The WA shall be able to communicate security breaches to the user
WAFR9	The WA shall be able to let the user interact on such security breaches
WAFR10	The WA shall be able to update a policy in the PR corresponding to the user's decision
WAFR11	The WA shall let the user choose a device category to which the device belongs

Table 9: Functional requirements for the Web application

In the tables 6 to 9 above all the functional requirements for the system has been define, these will be the basis for the design of the system presented in the coming chapter. The requirements will also conclude the analysis of this thesis.

5. System Design

The coming chapter aims at explaining the design of the system. This design will be based on the requirements elicited at the end of the analysis in the preceding chapter. The system design will carefully described from each requirement by the help of Unified modeling language (UML). In this way it will possible to deduct how every part of the system should be developed, in order to fulfill the system requirements.

5.1 System Overview

In order to start design the system in a correct way, an overview of the system will be created, in that way it will be possible to see the relevant entities of the system and how they are expected to interact.
Even though the IoT security gateway is the heart of the system, it does not communicate directly with all entities of the system. Therefore it makes sense to start with an overview of the web application, which can be seen as the link between the gateway, user and policy repository. An overview of this is presented in the Context diagram below.

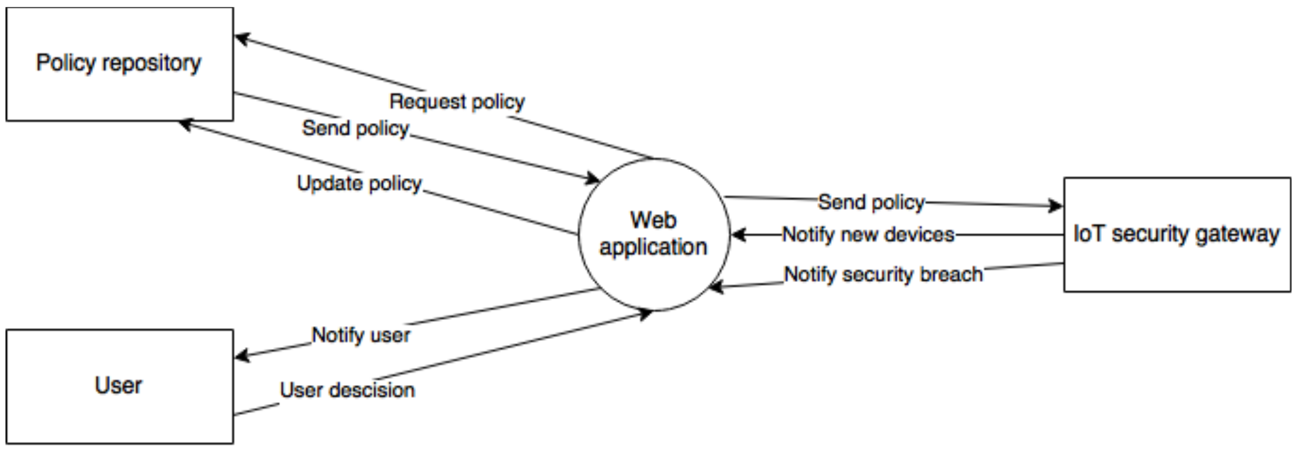


Figure 18: Context diagram of the web application

As shown in the diagram (figure 18), the web application interacts with both the user, the IoT security gateway and the policy repository, binding all of the communication together. The reason for letting the web application be the binding entity, is that the all updates to policies has to be confirmed by the user, to whom the IoT gateway does not have a direct connection, besides that the user has to be notified in the case of a security breach, which also has to go through the web application.
Regardless the IoT gateway is the heart of ensuring the security, therefore it also makes sense to create an overview of the entities communicating with the gateway.

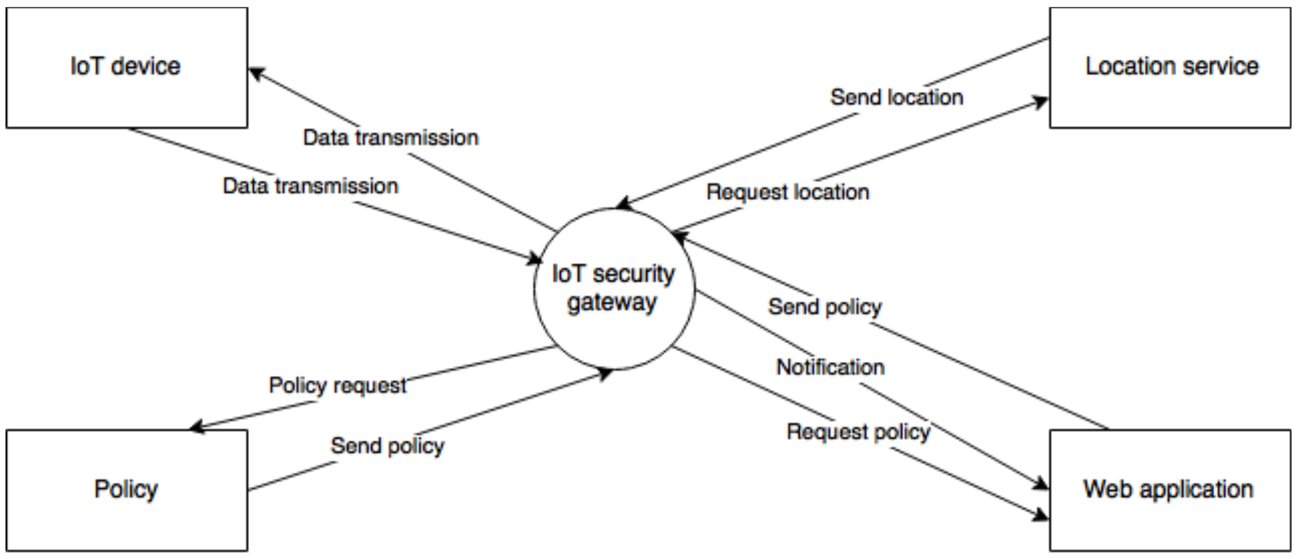


Figure 19: Context diagram of the gateway

From Figure 19 it can be seen that the IoT gateway communicates with not only the IoT devices and the web application, but also a location service and naturally the policies stored. The two entities “policy” and “location service” could in principle be fetched through the web application but as it is assumed that one gateway has a limited number of IoT devices, it makes sense to store the policies locally and update them when needed. In the case of the location service, which is binding IP addresses to a location, this service is a third party, thus it makes sense to request it directly instead of adding the web application as another link in the process.

5.2 Gateway

From the overview, an overall structure of the system has been presented, with information of all the entities and how they connect. The next step will be to design the gateway itself, which is the purpose of this sub chapter.

The first thing as corresponding to GWFR1 is to let the gateway log the metadata from the packets both transmitted and received at the IoT device, as depicted in figure 20.

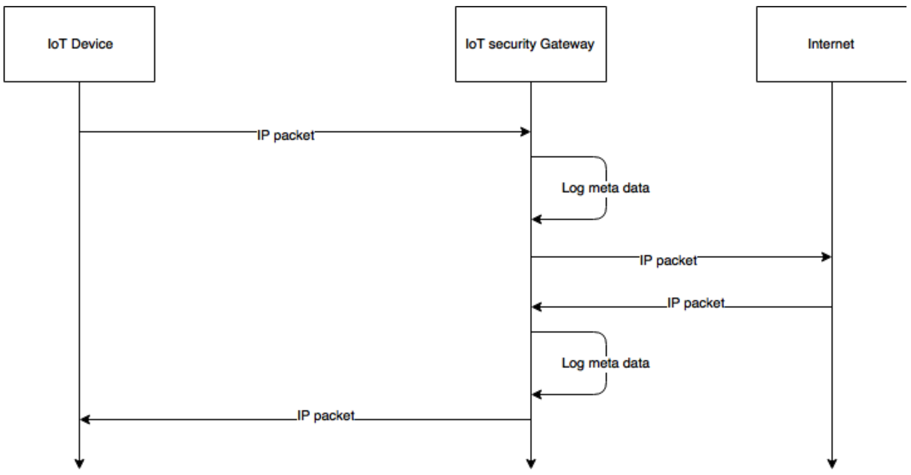


Figure 20: Sequence diagram of the data packet transmission and logging

From this the gateway is not only logging the metadata of the transferred packets, but is also able to identify the device transmitting data by the unique MAC address, thus the gateway is able to identify the device of interest corresponding to GWFR2.

From the requirements it is also specified that the gateway should be able to communicate with external entities (GWFR3) which is a natural consequence of the findings in the analysis, however it seems rather excess to show it here, therefore a generic overview of communication between different entities in the system can be found in appendix C. However it makes sense to show how the gateway will fetch a policy (GWFR4), which is shown in figure 20 below.

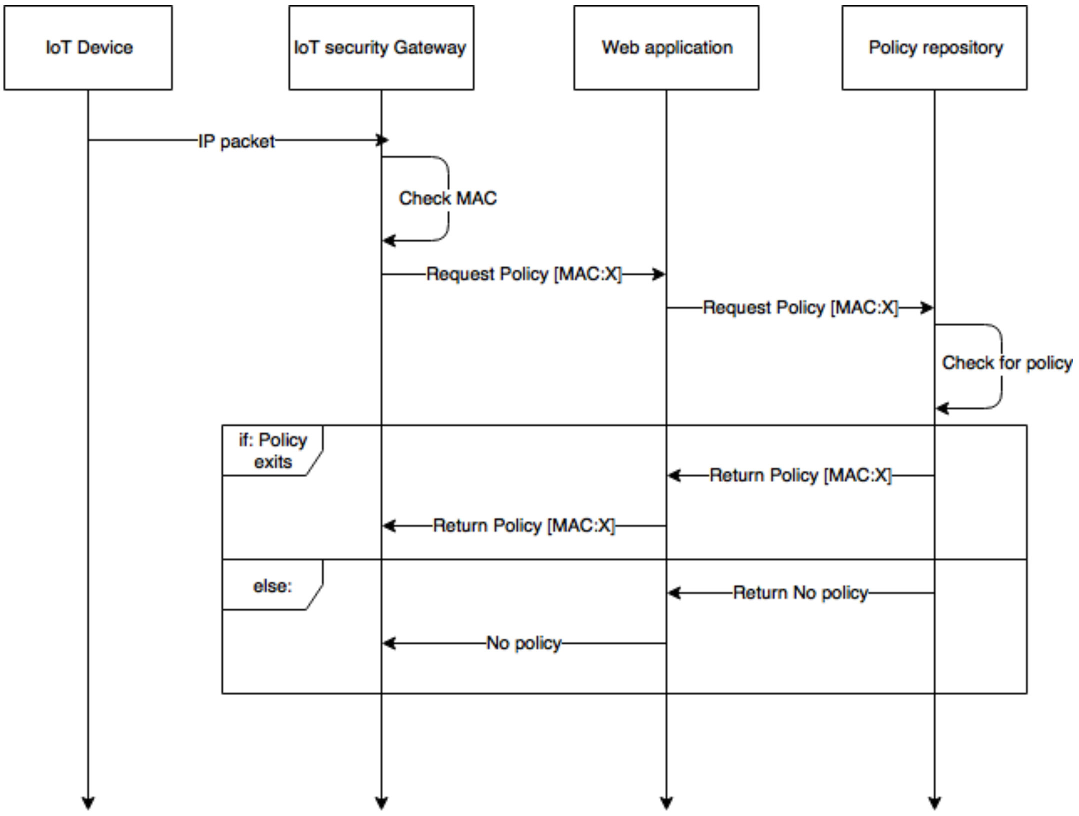


Figure 20: Fetching of policies

As it can be seen the gateway will request the web application for a policy with given MAC address, a request the web app will forward to the policy repository. If the policy exits it will be returned to the gateway, however if it does not exist, a "no policy" will be sent. This will tell the gateway that the IoT device transmitting is a new device and accordingly invoke a training period for the device(GWFR6) which can be seen in appendix D.

With the design of how the gateway logs and fetches policies, it makes sense to show how a policy is enforced. Naturally a prerequisite for this is that the gateway has at least one policy that can be enforced. The following sequence diagram (figure 20) will show how enforcement of the policies work which also covers the remaining requirements for the gateway (GWFR5, GWFR7, GWFR8, GWFR9, GWFR10).

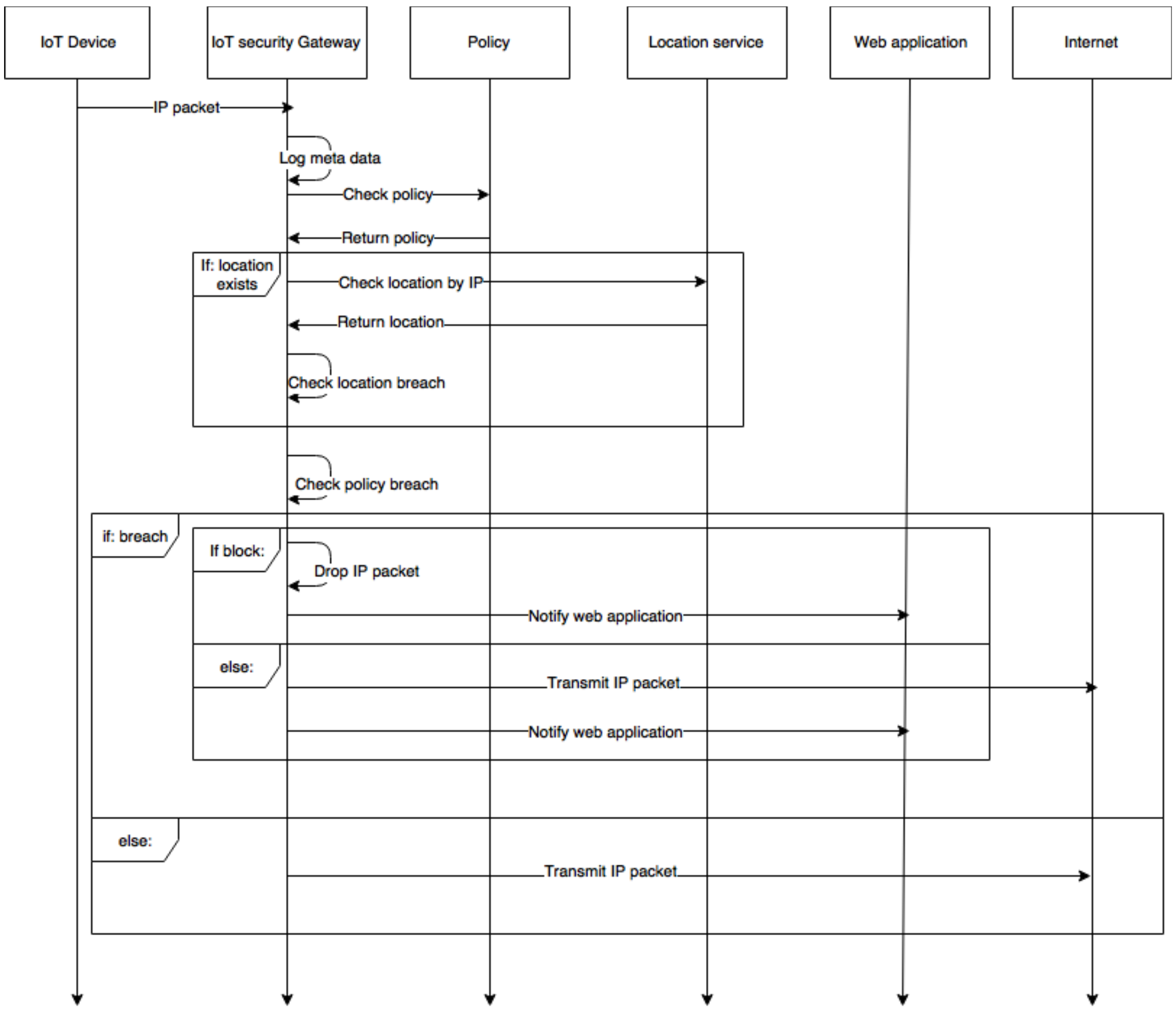


*Figure 20: Sequence diagram of how to handle policy enforcement and possi*ble breaches

As it is shown the gateway will check and IP packet from an IoT device against the corresponding policy. If the policy contains information about location restrictions, such as packets to China are not allowed, the gateway requests a location based on the destination IP, this is handled by a third party service. Depending on the policy, if a breach occur, the gateway will either drop the packet and notify the web app, or transmit the packet and notify the web application. If the policy is obeyed, the packet is just transmitted without further notice. The depicted diagram shows an IP packet transmitted from the IoT device, naturally the policy check works in the same way if the packet is transmitted from somewhere on the internet towards the IoT device. The functionality and possible impact would be the same, the packet is just going in the opposite direction.

By this, the system design of the IoT security gateway will be concluded, a hands on approach of how this design could work will be suggested in the chapter Implementation further down in this thesis.

5.3 Policy

The design of the policy is as important as other parts of the system, even though it can seem like a simple task. The design and contents of a policy has to be considered carefully in order to make sure the gateway acts as intended, and thus is able to raise the level of security. In this section the design of a policy will be defined, so it complies with the requirements and works as intended.
The first step is to look into, what a policy should contain and how it should be build. From chapter 4.3 it is known what kind of metadata is available, which would be the basis of the policy:

- Hardware: MAC address
- IP: Total length
- IP: Protocol
- IP: Source address
- IP: Destination address
- Various: Timestamp
- Various: Amount of data
- Various: Location

Besides this, the policy needs to know, whether the rule is for outgoing or incoming traffic, e.g., a destination IP for incoming traffic will always be known as it belongs in the local network, and thus will not be helpful to prevent attacks, whereas a destination IP for outgoing traffic may vary, and thus is relevant to decide whether traffic is legit or not.

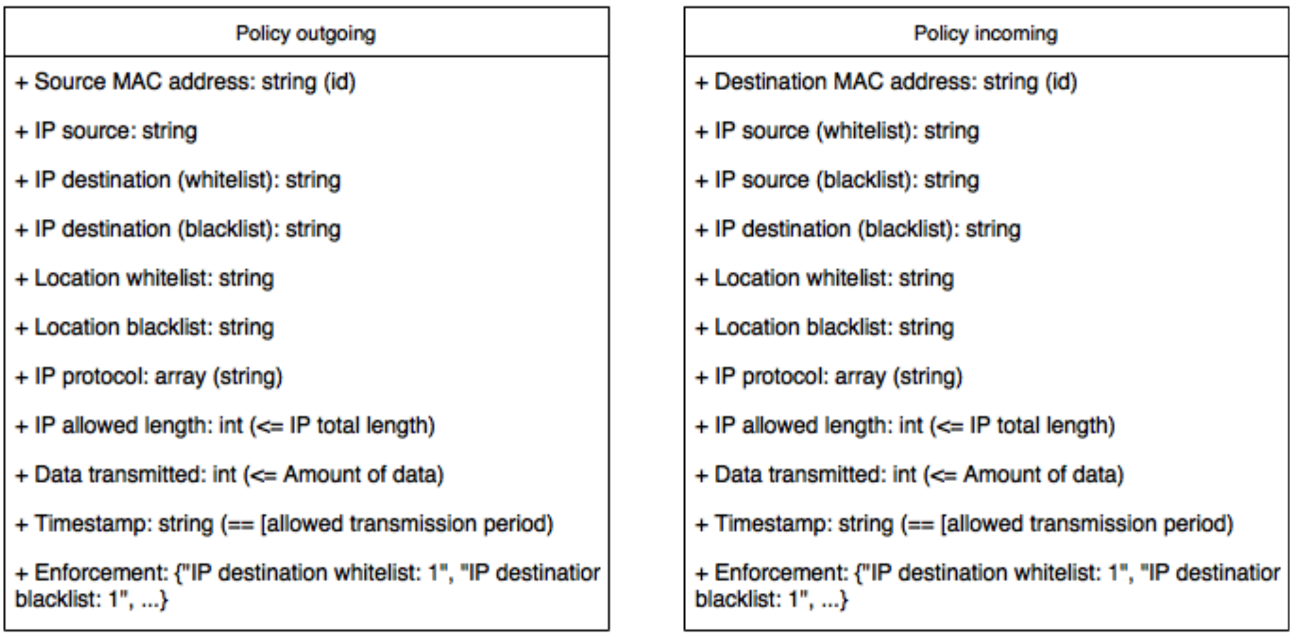

Policy outgoing
+ Source MAC address: string (id)
+ IP source: string
+ IP destination (whitelist): string
+ IP destination (blacklist): string
+ Location whitelist: string
+ Location blacklist: string
+ IP protocol: array (string)
+ IP allowed length: int (<= IP total length)
+ Data transmitted: int (<= Amount of data)
+ Timestamp: string (== [allowed transmission period)
+ Enforcement: {"IP destination whitelist: 1", "IP destinatior blacklist: 1", ...}
Policy incoming
+ Destination MAC address: string (id)
+ IP source (whitelist): string
+ IP source (blacklist): string
+ IP destination (blacklist): string
+ Location whitelist: string
+ Location blacklist: string
+ IP protocol: array (string)
+ IP allowed length: int (<= IP total length)
+ Data transmitted: int (<= Amount of data)
+ Timestamp: string (== [allowed transmission period)
+ Enforcement: {"IP destination whitelist: 1", "IP destinatior blacklist: 1", ...}

Figure 21: The information a policy must contain

In figure 20, two policies classes is depicted containing a policy for outgoing transmissions and for incoming transmissions.
The gateway must identify the type of transmission by the MAC address; if the source MAC address is identified as a device connected to the security gateway the outgoing policy must be used, and vice versa if the destination MAC address is identified as a device connected to the gateway the incoming policy must be used.
The policy is defined in the way it should be enforced, this is relevant especially to the white- and blacklists. If there is a whitelist (meaning the policy entity whitelist is not empty), this must be checked first, if the actual destination or source IP (depending on the policy in use) does not comply with the policy, the gateway must enforce the police. Next step is to check the blacklist and again enforce the policy and so on.
The last section of the policy is enforcement, which tells the gateway what to do, if a breach in one or several of the rules is detected, the enforcement will contain information about all the rules, and how to enforce them where the value “0” is very liberal and means no enforcement, the value “1” means notify the user and the value “2” means notify user and block transmission.
Above a single policy is described, however it is also important to know the relations a policy can have, which is specified PFR8, this is both to know to whom the policy belongs, and also to be able to interchange policies between users, by the help of the policy repository.

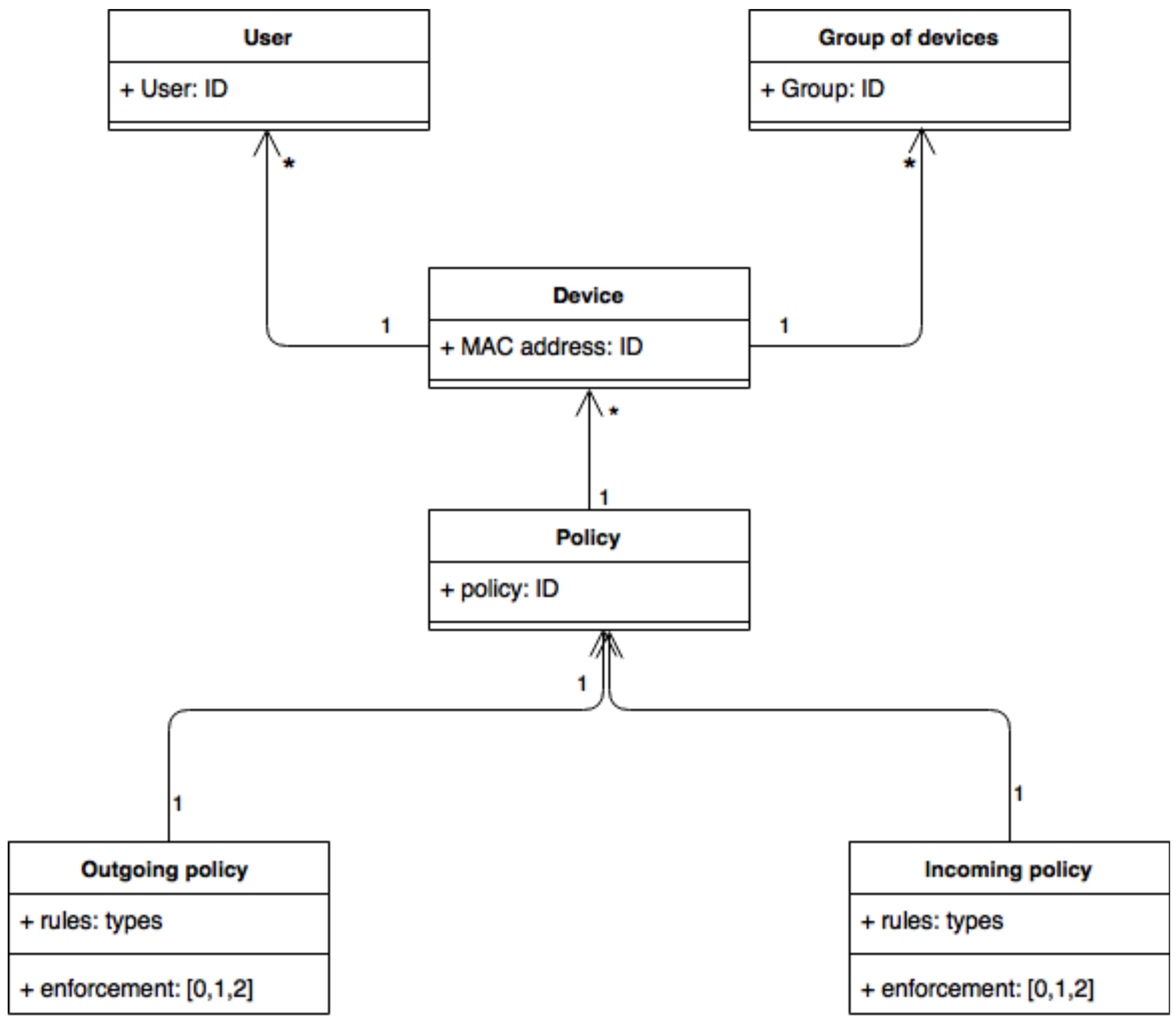


Figure 22: Policy associations

As it is shown in figure 22, a policy a policy consists of an incoming an outgoing policy (depicted in figure 21). A policy can belong to many devices, but a device only have one policy, which makes sense if a setup contains several identical devices. Furthermore a device belongs to only on user, and naturally a user can have more devices in his or hers setup. Lastly a device can belong to a group of devices. This makes good sense in terms of the policy repository, as it makes it able to group with similar functionalities and thus similar policies. Doing this, will in time make it possible to suggest suiting policies to new devices, so the training period will be minimized.
Thus all of the requirements for a policy has been covered and this concludes the policy subsection.

5.4 Policy Repository

The policy repository is an entity meant to contain all policies existing, not only for a single gateway but for all gateways used. Therefore it needs to run outside the local area

network, and in the same time expose itself to all connected gateways. Therefore it makes sense to let the communication with the repository go through the web application, as this is the entity the gateway and the user connects with.
The repository can basically be seen as a big collection of all policies with functionality to handle policies as defined in the PRFR specification. The fetching of policies (PRFR2) has already been described by figure 2o. Besides this the policy repository must also be able to act upon user decisions provided through the web application as shown in figure 23 below.

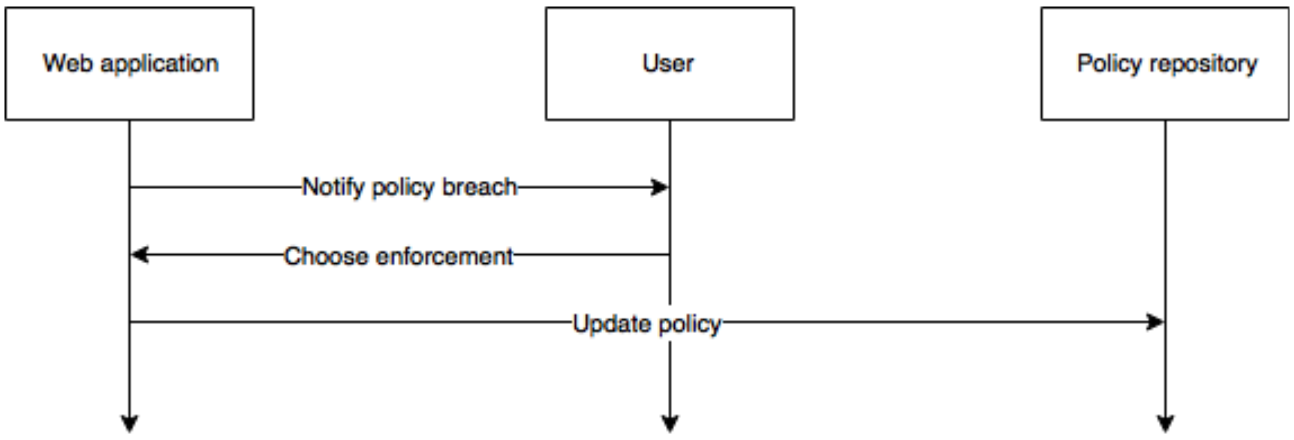


Figure 23: Policy update flow

By letting the policy updates go through the web application and let the user decide what should happen in case of a breach, it is ensured that policies is always compliant with the user's decision, and the policies wouldn't start to update itself autonomously.
The last requirement for the policy repository (PRFR6) will not be explained in this section as it is tightly coupled to the web application and user information, therefore it makes more sense to show the design in the coming chapter dealing with the web application.

5.5 Web application

The last part of the system design aims at showing functionality of the web application. The web application is the part of the system that binds the communication between all the devices together as well as letting the user interact with the system.
However, as this system should be seen as a whole, where every entity is needed in order for the system to work, some of the requirements has already implicitly been designed throughout the preceding chapters. Therefore this section will focus on how the interaction on the users should be.

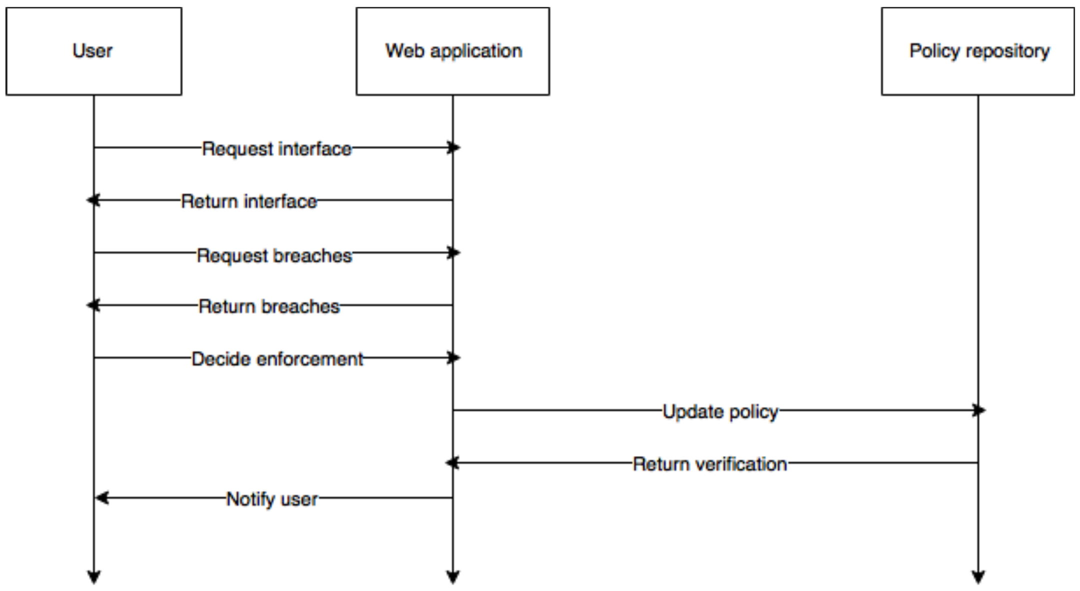


Figure 24: web application and user interaction

In figure 24 above a flow with user interaction of policy breaches can be seen, here it is shown that the user will be presented with policy breaches, when he/she requests it from the web application. Depending on the enforcement choice of the user, the web application will update the policy repository. Even though the above figure shows interaction in terms of policy breaches, the flow for a device in training period will be the same, as the training in principle can be considered as “constant” breaches of the policy of the device undergoing “training”.
It becomes clear that this does not state how exactly the user interacts with the web application, this is merely because the user interface could be reached by different means in example through a regular web interface or a smartphone application. The way of interacting with the interface does not change the flow and activities of the web application, for which reason the specific interface has not been designed, however this will to some extend be covered in the coming chapter about the implementation of the system.

This chapter has shown how the theoretical design of the system looks like, by the help of different diagrams in combination with description of the activities. It is believed that the system design in combination with the requirements will provide enough information of how the system containing an IoT security gateway, a web application and a policy repository should be designed, without demanding any specific technologies or needs to implement the system.
In the coming chapter suggestions to how to implement the system in practice will be discussed containing relevant technologies and means of implementation, in order to make the system a reality.

6 Implementation

The implementation chapter is going to focus on how the proposed system could be implemented in reality. This means that the section will focus on how the system design could be realized by the means of real methods and technologies. Some of the technologies in use have been described in chapter 3.3.4, and will be referred to from there, whereas other and more specific methods might be described on during the chapter. The chapter will end with a proof of concept, showing how system could work, it will not be a fully implemented system, but it will show the general idea of how the security gateway functions.

6.1 Deployment

To gather a more practical overview of how the system interconnects and communicates, a deployment diagram will be presented. This diagram serves the purpose of showing different entities in a more detailed and practical view than the context diagram presented in the early phase of the system design.

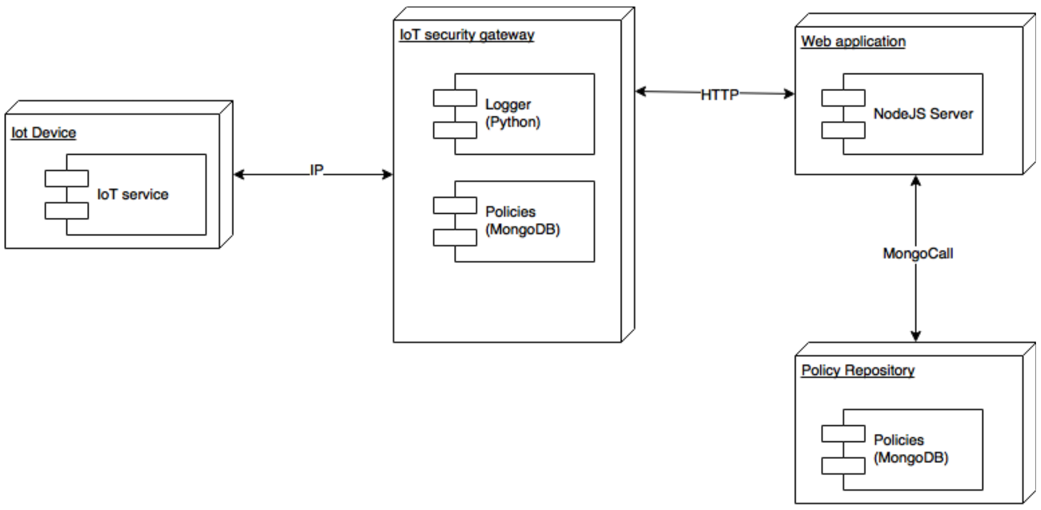


Figure 25: Overview of the entities in the system with specified technologies

As it can be seen in figure 25 above, the system works with four different entities with different objects. The communication between the IoT gateway and the IoT device has been defined as IP, which naturally will be through either Ethernet or WiFi protocol, as defined earlier. In the gateway a logger, handling the actual data logging is present along with a set of policies for the connected IoT devices stored in a MongoDB.
The communication between the IoT security gateway and the web application is through HTTP towards a NodeJS server, exposing the relevant API's to the IoT gateway. Lastly the NodeJS server is connected to the policy repository (PR) which is a MongoDB containing all policies. In this setup the policy repository is located at the same server as the web application, why only a Mongo call from node is needed to fetch or update a policy. The PR

could also be located at an external server, in which case the connection between the web application and the PR would be HTTP.

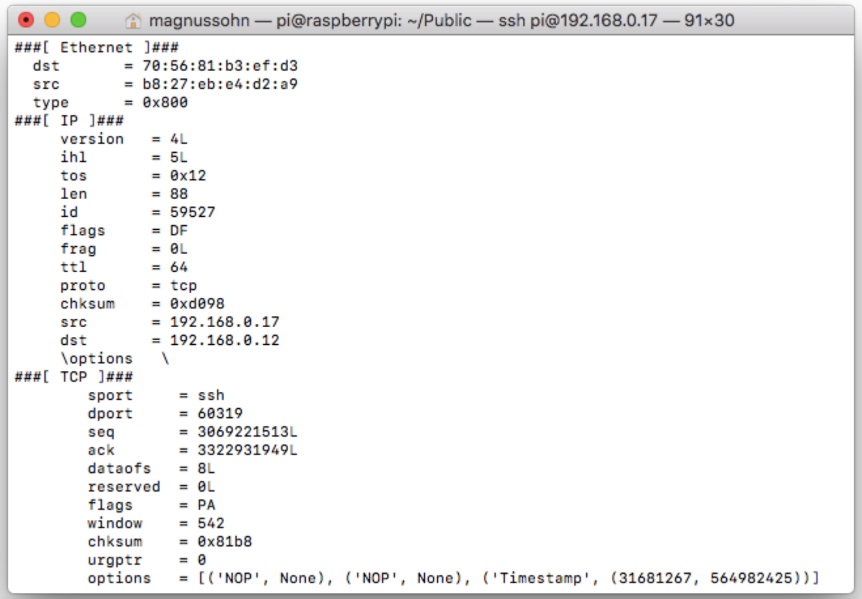
6.2 IoT Security Gateway

The first thing to take a look into is the IoT security gateway. It is known that this device is an intermediary between the IoT device and the home router connecting to the internet, therefore this gateway must be able to handle the logging of the data passing through it as well as enforcing the policies for a given device.

It has been decided to use a Raspberry Pi as the hardware resource for this project. The Raspberry can run a variety of different Linux distributions, but the most widely used is the distribution aiming directly at the Raspberry, namely Raspbian[55]. The operating system runs smoothly on the Raspberry Pi and works with Python out of the box.

6.2.1 Logging

The gateway needs to be able to log information about the data packets transmitted, there are several ways to do this, in this project it has been chosen to use the Python library Scapy[56]
Scapy provides a quite useful feature, called sniffing[57]. The sniffing method in Scapy simply provides a possibility to sniff data packets on a given network interface in example WiFi.


magnussohn — pi@raspberrypi: ~/Public — ssh pi@192.168.0.17 — 91×30

```
###[ Ethernet ]###
  dst       = 70:56:81:b3:ef:d3
  src       = b8:27:eb:e4:d2:a9
  type      = 0x800
###[ IP ]###
     version   = 4L
     ihl       = 5L
     tos       = 0x12
     len       = 88
     id        = 59527
     flags     = DF
     frag      = 0L
     ttl       = 64
     proto     = tcp
     chksum    = 0xd098
     src       = 192.168.0.17
     dst       = 192.168.0.12
     \options   \
###[ TCP ]###
        sport     = ssh
        dport     = 60319
        seq       = 3069221513L
        ack       = 3322931949L
        dataofs   = 8L
        reserved  = 0L
        flags     = PA
        window    = 542
        chksum    = 0x81b8
        urgptr    = 0
        options   = [('NOP', None), ('NOP', None), ('Timestamp', (31681267, 564982425))]
```

Figure 26: A dump of the output using Scapy's sniffing method

From figure 26 above it is shown that Scapy can provide most of the relevant metadata for every IP packet transmitted through a given interface, just by calling the sniff() method in Scapy.
The only data missing is the amount of data transmitted or received by a single device, the time and the location,

Time stamp:
The timestamp can be retrieved by using Python's time package[58], the current time can simply be retrieved by using the time() method, and add this to the package information. In example: currentTime = time.ctime() will add the current time in the following format: *Sun Oct 29 12:08:29 2017*

Amount of data:
The amount of data transmitted is a bit more arbitrary, as this have to be calculated within a given time period. It does not make sense to just count every time a packet is transmitted from a device, if the boundaries are not defined. Here it makes sense to create a counter that is limited to a certain time period, in example 60 seconds, again the time library from Python can be used:

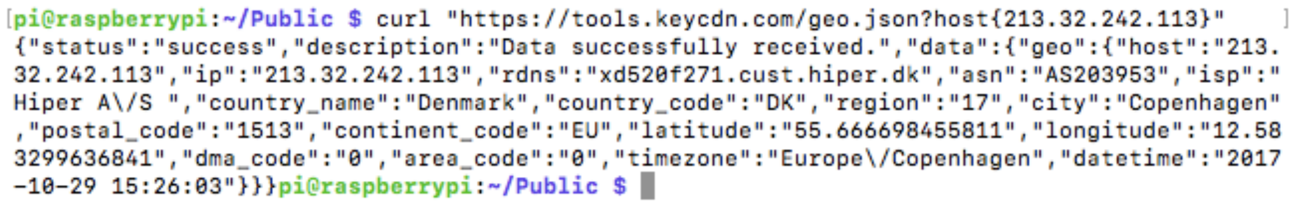
```
amountOfData = 0
currentTime = time.time()
countingTime = currentTime + 60

while time.time() < countingTime:
   if macSrc == [mac address]:
      amountOfData + 1
```

This above code shows an example of a counter that will count the number of packets transmitted by a given MAC address within 60 seconds. This will provide the desired information about how much data transmitted by a single device. Naturally counting time can be adjusted to what makes sense, however it will rely on actual testing on devices to know.

Location:
The last meta data needed is the location, which will be fetched from an external service as described earlier. The fetching of the location relies on the either the destination IP or source IP depending on the data flow. The service, keycdn[59] going to be uses, has an API where a simple get request including the IP address of interest will return the location information of the given address:
By calling the address:
"curl "https://tools.keycdn.com/geo.json?host={IP or hostname}"”

With the relevant IP, a JSON response would returned with all the information, see figure 27 below:



```
[pi@raspberrypi:~/Public $ curl "https://tools.keycdn.com/geo.json?host{213.32.242.113}"     ]
{"status":"success","description":"Data successfully received.","data":{"geo":{"host":"213.
32.242.113","ip":"213.32.242.113","rdns":"xd520f271.cust.hiper.dk","asn":"AS203953","isp":"
Hiper A\/S ","country_name":"Denmark","country_code":"DK","region":"17","city":"Copenhagen"
,"postal_code":"1513","continent_code":"EU","latitude":"55.666698455811","longitude":"12.58
3299636841","dma_code":"0","area_code":"0","timezone":"Europe\/Copenhagen","datetime":"2017
-10-29 15:26:03"}}}pi@raspberrypi:~/Public $
```

Figure 27: Request and response from keycdn geolocation tool

This respond can be made from python with a simple get request. The response will then be loaded as json, thus it is searchable for the actual location, the source code for the location search can be seen in the attached file “pythonGet.py” attached to this thesis.

By this it is possible to gather all meta data relevant to the logging device, in order to be able to check if the transmissions to and from IoT devices obeys the rules in the policy.

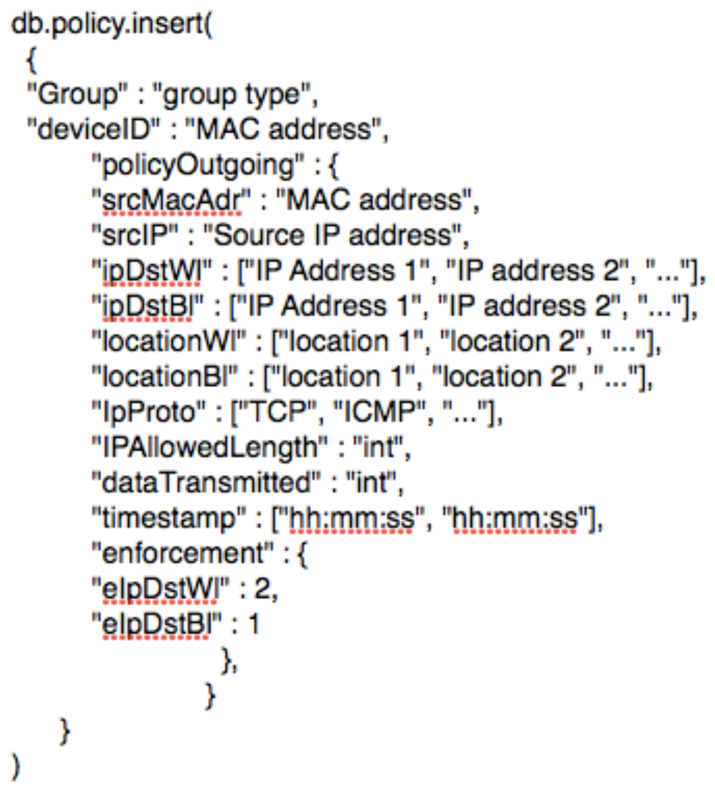
6.2.2 IP Tables:

The last thing the gateway needs to implement is IP tables[60]. IP tables is a program included in all newer Linux distributions. The program opens for the possibility to filter IP addresses within the system, and as such it can be used to implement the blocking functionality of the gateway.
The function “drop” in IP Tables, simply drops an incoming packet if it the IP is listed to be dropped [61].
By using IP Tables in the IoT gateway, it will be possible to drop packets, violating a policy and has the enforcement rule to block a package.

6.3 Policy Setup

As we know from the System design a policy contains many different types of information. All of the information that constitute a policy will be stored in a MongoDB described section 3.3.4. The MongoDB seems like a good choice as one does not need to define strict relations between the data contained in a policy. One document in the database will simply make up a policy.

The MongoDB naturally has to be setup, which is quite an easy task. For the case of this project we will create a database called policies, with a collection called policy to which the actual policy can be inserted as a document. As the MongoDB uses a JSON like format, it is quite easy to insert and structure the data that constitutes a policy. From the system design it has been shown how a policy should be designed, so the task is simply to use this is simply to insert this data into a document in the database.



```
db.policy.insert(
 {
 "Group" : "group type",
 "deviceID" : "MAC address",
        "policyOutgoing" : {
        "srcMacAdr" : "MAC address",
        "srcIP" : "Source IP address",
        "ipDstWl" : ["IP Address 1", "IP address 2", "..."],
        "ipDstBl" : ["IP Address 1", "IP address 2", "..."],
        "locationWl" : ["location 1", "location 2", "..."],
        "locationBl" : ["location 1", "location 2", "..."],
        "IpProto" : ["TCP", "ICMP", "..."],
        "IPAllowedLength" : "int",
        "dataTransmitted" : "int",
        "timestamp" : ["hh:mm:ss", "hh:mm:ss"],
        "enforcement" : {
        "eIpDstWl" : 2,
        "eIpDstBl" : 1
                        },
                      }
     }
)
```

Figure 28: Example of creating a policy in the mongoDB

In figure 28, an example policy inserted into a MongoDB is shown. The policy contains a group, in that way it is possible to let it belong to a certain group of devices, furthermore it contains a deviceID which is the MAC address belonging to the IoT device of interest. The policy itself, in this case a policy for outgoing transmission contains all the rules as described earlier, some rules can just be a single value, e.g. “IPAllowedLength”, telling the system what the maximum allowed length of a datagram is, whereas others can contain several values such “ipDstWl” which is the destination IP whitelist, in here all IP addresses that are whitelisted will be stored. Lastly the policy has enforcement part, where the rules of enforcement is specified. “eIPDstWl” which is the enforcement rule of the IP whitelist is set to the value 2, meaning that if destination IP is not on the whitelist it should be enforced by the rule 2, which is notify the user and block the IP address.
In the same way it is possible to insert data to a MongoDB, it offers quite simply query and update functions.

Querying a policy for a given device, is simply done by a find method:
db.policy.find({key : value})

By this a policy for a given device can be fetched by providing the key “deviceID” and the MAC address of the device as value.
In the same way the database can be updated by the method db.policy.update() providing a key value for the collection to be updated, and what to be updated.
In example the value of data transmitted could be set by the following method:

db.policy.update({“deviceID” : “MAC address”}, {$set: {“dataTransmitted” : “newValue”}})

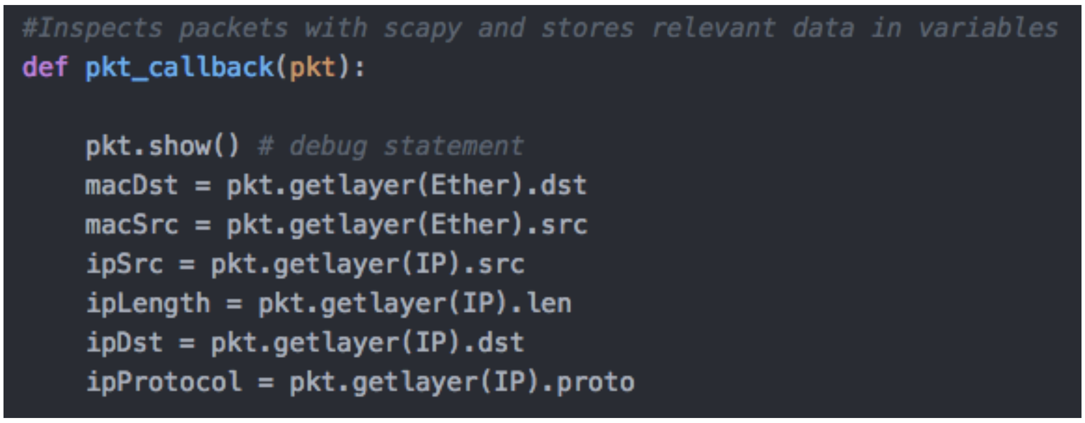
This way of implementing policies to different devices is quite simple and easy to setup. Furthermore the setup can be used both by the IoT security gateway and the policy repository, as both supports the MongoDB. The results are presented in JSON which makes it easy interpretable by Python, as shown with the location service, thus it is possible to compare policies with logged data from the gateway and by that enforce the policies.

6.4 Proof of Concept

Due to the time limitations of this project and the fact that this is a single person project, it has been decided not to implement a full version of the system, but instead a proof of concept showing that it is possible to log data flowing through an IoT gateway and act upon it.
This also means that the web application discussed throughout the thesis will not be implemented at the current time, however it is a hope to be able to create an implementation later on, so the system will function to its fullest. Even though the web application is not going to be a part of the implementation, it is believed that considerations from the analysis and system design are usable and can be used to implement the web application at some point.

The proof of concept focuses on the gathering of metadata and the possibility to act upon it by the means of a policy. This all done by a Python program running on the Raspberry Pi, in combination with a MongoDB which acts as the policy repository.

Sniffing packets:
The Scapy sniffing function is the first that will be used to capture the meta data in the packets transmitted. Furthermore all relevant meta data will be stored in order to compare with the policy as shown in figure 29:

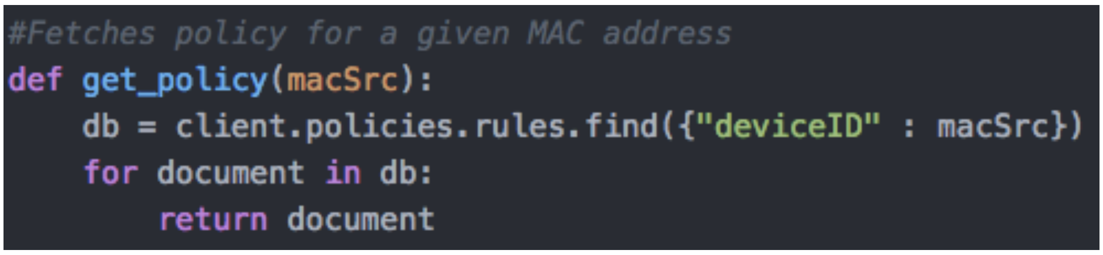


```
#Inspects packets with scapy and stores relevant data in variables
def pkt_callback(pkt):

    pkt.show() # debug statement
    macDst = pkt.getlayer(Ether).dst
    macSrc = pkt.getlayer(Ether).src
    ipSrc = pkt.getlayer(IP).src
    ipLength = pkt.getlayer(IP).len
    ipDst = pkt.getlayer(IP).dst
    ipProtocol = pkt.getlayer(IP).proto
```

Figure 29: Sniffing and saving metadata

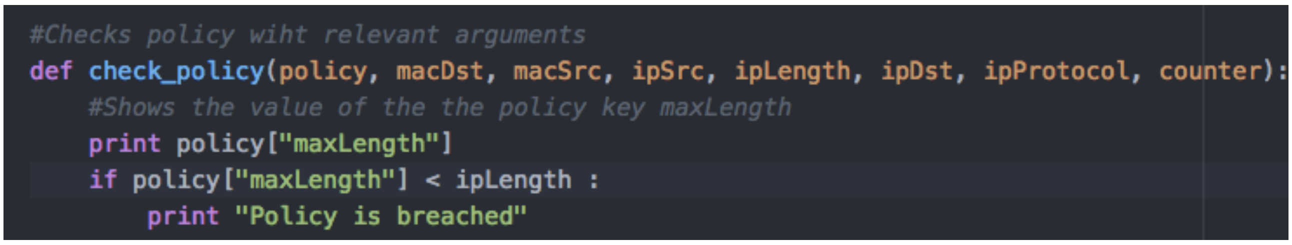
The next thing is to fetch the policy relevant to package, this is done by connecting to the MongoDB and thus using the find() method to fetch the relevant policy as shown in figure 30 below:



```
#Fetches policy for a given MAC address
def get_policy(macSrc):
    db = client.policies.rules.find({"deviceID" : macSrc})
    for document in db:
        return document
```

Figure 30: Connection to MongoDB and requesting policy.

The policy from the Mongo DB is then saved to a variable named “policy” to use when checking for breaches, which is done by a method called “check_policy”. The method simply checks if there is there is a difference in what the policy allows and what the packet contains, and example of this is shown in figure 31 below.



```
#Checks policy wiht relevant arguments
def check_policy(policy, macDst, macSrc, ipSrc, ipLength, ipDst, ipProtocol, counter):
    #Shows the value of the the policy key maxLength
    print policy["maxLength"]
    if policy["maxLength"] < ipLength :
        print "Policy is breached"
```

Figure 31: Policy check

As the web application has not been implemented, the policy breach in this scenario is only to print a statement, that the policy has been breached, which is also shown in the bottom of the picture. Naturally this is not a very intuitive, much less useful way to notify about a policy breach, but it fits the purpose of testing if the gateway actually reacts to policy breaches, which it does.

Even though the implementation is far from being fully done, it is believed that this small part has shown that it is possible to create a system that is able to check whether the systems is exposed to security risks by the help of logging and interpreting data transmitted to and from IoT devices connected to a gateway in the home. Furthermore this can be done without interfering with the way the IoT devices work from the manufacturer. This means that it is actually possible to create a system that raises the level of security in any for any IoT device connected through the gateway.

7. Future Work

The aim of this chapter is to discuss some of the future work and improvements to the system. The discussion will be based on suggestions and known solutions from other services implementing similar solutions.

7.1 Web application

As it has not been possible to implement to web application in the current state of this project it is natural to do this as the next step in the system.
The web application can be implemented in various ways, but it is believed that using a technology that supports RESTful communication will be a good basis, such as NodeJS described in chapter 3.3.4.
The web application should bind the user, IoT gateway and policy repository together, and is depended on a user interface in order to make decisions as most actions in this system is depended on a given user's choice.
The reason to build the web application using RESTful principles, is that it is stateless. In a distributed system like the one proposed, it makes sense as the application does not need to store previous states of the system, everything can be done by requesting and updating information in the moment of the transaction and then forget about it, no matter whether the application needs to talk to the IoT gateway or the policy repository and vice versa.
Furthermore the user interface needs to be discussed, as no decision has been made towards how to interact with the system. However it is believed that creating it as a responsive web site, would cover the needs of the interface, as it will run on any modern device containing a web browser and access to the internet. A further development could be to develop a smartphone application taking advantage of the possibilities provided by a such, in example it would be easier to send push notifications directly to the user when an incident occurs and thus let the user react immediately instead of having to login to a web site, to check for breached of a policy.
However this will require a lot more work, and therefore would not be the first choice of designing the user interface, but it could be an add-on later in the implementation of the system.

7.2 Registration

The coming discussion emerges around some of the subjects that is not a part of this thesis, but deserves to be discussed, namely the registration of devices in the system.
It has been described that the IoT security gateway should work as an intermediate between the actual IoT devices, and the home router and thus the internet. This means that every device needs to connect to the IoT gateway. This connection is imagined to

work in a similar way to when one connects a device directly to a router, you either plugin a ethernet cable or connects through the WiFi by choosing the WiFi name to connect to and the correct passphrase to the WiFi.
This could be done by creating an interface directly at the gateway, where it is possible to login and set WiFi parameters, similar to how it is done with a home routers first setup.
Besides this, it should also be possible to register the IoT gateway towards the web application so the gateway can be bound to a user.
This could be done by letting the gateway generate a pseudorandom key upon the first login, combined with a serial number of the gateway, this information should be used by the user to register a gateway when logging in to the user interface of the web application.
In this way a single gateway can be tied to a user account within the web application.
Furthermore the user should be able to create an account at the web application, whether it should be a regular registration process with email and password or it should use a third party identity provider, such as Google or Facebook, to validate a user would be up to debate when designing the web application. However it could be argued to use a third party to avoid storing sensitive information about a user and password in the system itself, which will require additional focus on the security in the web application. By letting a third party handle the login process, the storing of such information is not to be considered, making the web application more simple to implement.

7.3 Automation

The last thing to look into would be far in the future, but nevertheless a significant improvement of the system, namely to implement a logic, that will be able to automate the policy rules.
Machine learning and automation in general, would be a huge possibility for this system, if the amount of users and data in the system would reach a critical point it would be possible to combine and analyze data from all devices, and let the policies be updated automatically based on a such analysis. This could help to raise the level of security even further. A benefit of this could be better protection of IoT devices before an attack has even been detected.
An example of this could be that the web application discovers that a lot of IoT devices is under attack in a specific region of the world, if the attack is expanding, the web application could automatically push policy updates to all types of IoT devices under attack, and thus provide improved protection before the damage is done.
Naturally such a function requires a lot of data and a logic that is able to analyze the data flow, however it is believed that it could raise the level of protection significantly.

By this the small chapter on future work will be concluded. The next part of this thesis is going to discuss the work done in terms the problems stated in the introduction of the report.

8 Discussion

The coming chapter is going to discuss the findings in this thesis in relation to the problem stated in the beginning. It is going to touch some security problems related to IoT and what has been done to mitigate does, and try to draw a bigger perspective of the work done.

There is no doubt the term IoT has been a buzzword for many years, predicted to improve anything from home automation over traffic control to the production industry. Also there is no doubt that the IoT market in many ways have been a wild west with different manufacturer launching all kind of products of varying quality.
One of the biggest problems in terms of IoT is probably the one sided focus on smart functionality without keeping an eye on the security and possible misuse of these kinds products.
The "regular" computer industry has had many years to improve life cycles of products and ensure right implementation of the software in the products. This is not the case in the IoT industry. Ironically, the simplicity of many of the products also is the biggest drawback in terms of implementing well known features such as continuous updates and encryption. Until the industry is able to implement standardized frameworks and lifecycles into their product, this problem will most likely exist.

Trying to improve the security of IoT devices is a huge task, and this thesis has only grasped a bit of what could be done. With the limitations of not being able to change the implementation of a given IoT device, well known security principles such as ensuring confidentiality and integrity are hard to accomplish. This is why the approach to create a gateway to help mitigating the problems has been taken. The gateway cannot directly ensure the general CIA principle stated in the beginning of this thesis, however it can notify the user of possible problems with the communication with the IoT devices, and by this indirectly help to accomplish the principles. It cannot know whether the data is encrypted, but it can tell if it is transmitted through an insecure channel, or transmitted to someone it is not designated to, and thus alert the user of a possible breach. In the same way it cannot ensure the availability of the service, but it can, with the correct setup protect against an attack aiming at making the device unavailable, and maybe even more relevant it can detect, whether a device is acting abnormally, in example being used to DDOS attack against others, and block the device from communicating, thus protect other entities on the internet about becoming unavailable.

Does this system help at all one could ask, the answer is yes and no. With the current state of implementation it would not be very useful, however a design has been provided, showing how it could work with a full implementation, in which it can raise the level of security for a user of the system. However a real benefit of the system will depend of the adoption of it. As stated IoT products is still very undeveloped in terms of security and is expected to be many years to come. Implementing such a system will help raise the

security both for specific devices who have adopted the system, but also entities on the internet, that might suffer attacks from IoT devices.
Imaging router manufacturers adopting the principles and design shown in this thesis, so every home router will be able detect and protect against perpetrators misusing the devices. This would be a huge advantage to the internet as a whole and to the single user of IoT devices.
That would probably not be the case in the near future, but depending on how much IoT devices, and their traffic will take part of the transmission on the internet, it could be a future scenario.
Nevertheless, until the IoT industry has figured out a way to secure their products and ensure they aren’t misused, the proposals in this thesis could be a way for the individual user to raise the level of security in the devices he or she employs.

9 Conclusion

The first question to ask in this concluding chapter of the thesis is:
Has it been shown how to raise the level of security in IoT devices by the means of implementing a gateway as an intermediary?
The short answer is that it has, regardless the system has not been fully implemented. However the research done for this project has shown that it is a feasible way to raise the level of security. The background chapter uncovers that IoT devices in general lack sufficient security in their design, and therefore there is a need to address the problem. Naturally the best solution would be to implement security by design in the devices, but as this cannot just be accomplished over night, it is believed that using a gateway as shown in this thesis can help mitigate some of the problems. The functionalities and design of the gateway has been based on generic systems derived from real world products, combined with known vulnerabilities. Security frameworks has been used to break down the different risks and threats IoT devices are exposed to in order to deduct the requirements for the system. This has lead to a system design showing how a security gateway by the help of a set of policies can raise the security in IoT devices, lastly a proof of concept has been shown of how data could be logged and interpreted in a gateway, to notify the user about possible threats towards a device.
It is believed that this system design, and the principles and thoughts behind, with further work can be much needed contribution to securing IoT devices, until the time has come where the single devices are mature enough to ensure the security by design.
Therefore the research question being the basis of this thesis is seen as answered, thus concluding the current work on this thesis

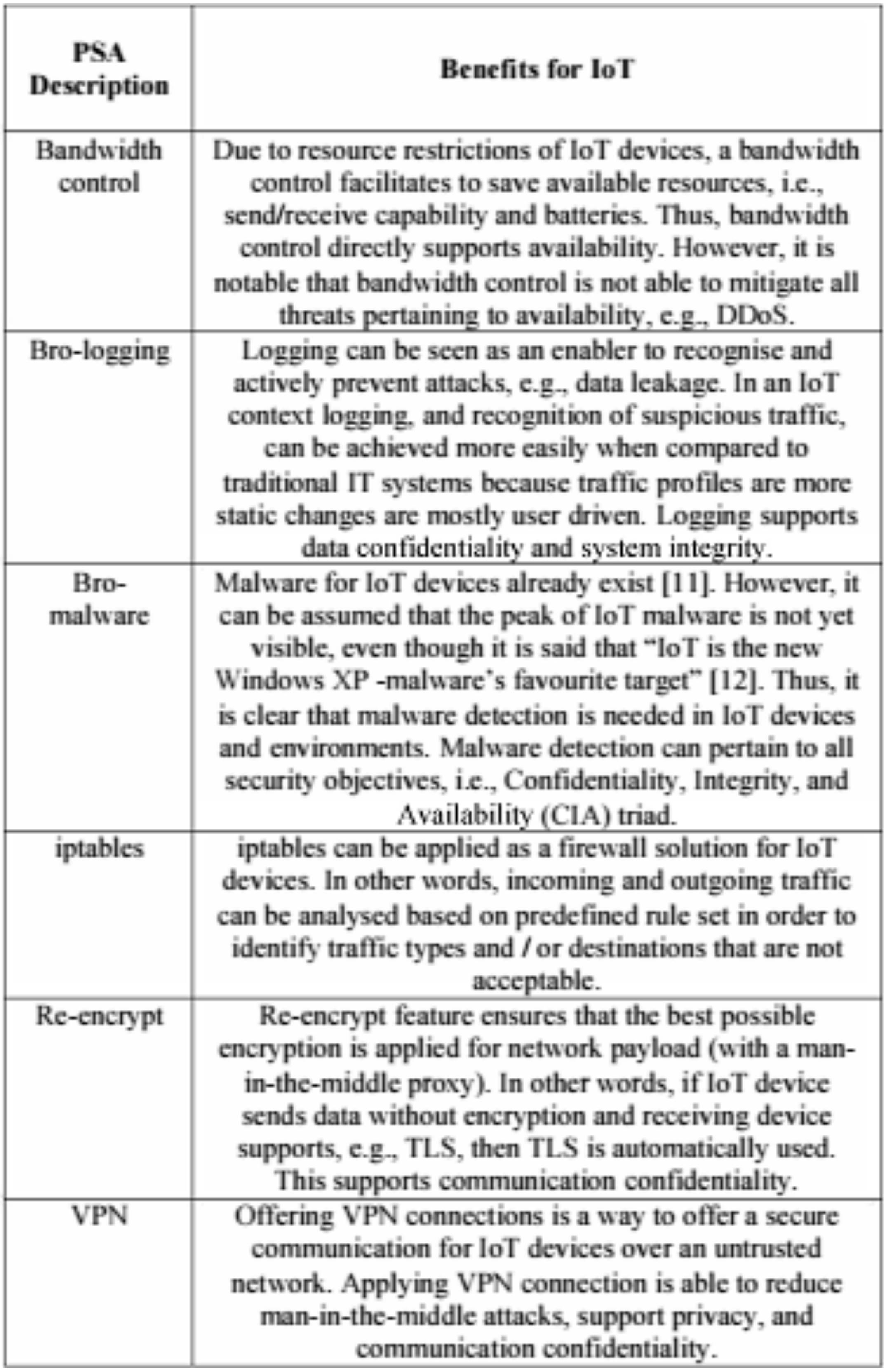
Magnus Nebel Sohn

References

[1] Stallings W. Network Security Essentials - Applications and Standards, Fourth Edition 2011, pp. 4
[2] Stallings W. Network Security Essentials - Applications and Standards, Fourth Edition 2011, pp. 5
[3] Kevin Ashton. 2009-06-22. That ‘Internet of Things’ Thing. Available: http://www.rfidjournal.com/articles/view?4986
[4] Sekhar Sarukkai. 2016-05-19. Ransomware and the Internet of Things: A Growing Threat. Available: http://www.esecurityplanet.com/network-security/ransomware-and-the-internet-of-things-a-growing-threat.html
[5] Canonical. 2017-01. Whitepaper. Taking charge of the IoT'ssecurity vulnerabilities.
[6] Gaona-garcía, Paulo, Montenegro-marin, Carlos Prieto, Juan David Nieto, Yuri Vanessa Distrital, Universidad José, Francisco Mateo, Fundación San. 2017. Analysis of Security Mechanisms Based on Clusters IoT Environments.. International Journal of Interactive Multimedia and Artificial Intelligence Special Issue on Advances and Applications in the Internet of Things and Cloud Computing Analysis vol 4 issue 3 pp. 55-60.
[7] Atamli A. W., Martin A. Threat-based Security Analysis for the Internet of Things. 2014. International Workshop on Secure Internet of Things. pp. 35-43
[8] Mohammad Irshad. December 2016. A Systematic Review of Information Security Frameworks in the Internet of Things. IEEE 18th International Conference on High Performance Computing and Communications pp. 1270 - 1275
[9] Atamli A. W., Martin A. Threat-based Security Analysis for the Internet of Things. 2014. International Workshop on Secure Internet of Things. pp. 35-43
[10] Abomhara M., Køien G. M. Security and privacy in the Internet of Things: Current status and open issues. 05-2014. Privacy and Security in Mobile Systems (PRISMS)
[11]Kozlov D., Veijalainen J., Ali Y. Security and privacy threats in IoT architectures. 2012. BodyNets '12 Proceedings of the 7th International Conference on Body Area Networks. pp. 256-262.
[12]Kozlov D., Veijalainen J., Ali Y. Security and privacy threats in IoT architectures. 2012. BodyNets '12 Proceedings of the 7th International Conference on Body Area Networks. pp. 260.
[13] Babar S., Stango A., Prasad N. Proposed embedded security framework for Internet of Things (IoT). Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on. 03-2011
[14] Pasha M., Shah S., Syed M. W., Pasha U. Security Framework for IoT Systems. International Journal of Computer Science and Information Security. 11-2016
[15] Oh S., Kim Y., Security Requirements Analysis for the IoT., 2017 International Conference on Platform Technology and Service (PlatCon), Feb 2017, accessed: 2017-05-25
[16] Kuusijärvi J., Savola R., Savolainen P., Evesti A. Mitigating IoT Security Threats with a Trusted Network Element. 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), december 2016, accessed: 2017-05-20. pp. 260
[17] Kuusijärvi J., Savola R., Savolainen P., Evesti A. Mitigating IoT Security Threats with a Trusted Network Element. 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), december 2016, accessed: 2017-05-20. pp. 262, figure 1.
[18] Kuusijärvi J., Savola R., Savolainen P., Evesti A. Mitigating IoT Security Threats with a Trusted Network Element. 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), december 2016, accessed: 2017-05-20.
[19]Kuusijärvi J., Savola R., Savolainen P., Evesti A. Mitigating IoT Security Threats with a Trusted Network Element. 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), december 2016, accessed: 2017-05-20. pp. 262
[20]Kuusijärvi J., Savola R., Savolainen P., Evesti A. Mitigating IoT Security Threats with a Trusted Network Element. 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST), december 2016, accessed: 2017-05-20. pp. 262
[21] Contu R., Kish D., Lawrence P., Deshpande S., Predicts 2017: Security Solutions, November 2016. Accessed: 06-2016

[22] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity v. 1.0, February 2014, accessed: June 2017
[23] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity v. 1.0, February 2014, figure 1, pp. 7, accessed: June 2017
[24] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity v. 1.0, February 2014, pp. 7, accessed: June 2017
[25] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity v. 1.0, February 2014, pp. 8-9, accessed: June 2017
[26] Kim D. H., Cho J. Y., Lim J., Developing IoT Security Requirements for Service Providers, International Information Institute (Tokyo). Information; Koganei vol 19, Feb 2016, accessed: June 2017.
[27] Kim D. H., Cho J. Y., Lim J., Developing IoT Security Requirements for Service Providers, International Information Institute (Tokyo). Information; Koganei vol 19, Feb 2016, Figure 2, pp. 600 accessed: June 2017.
[28] Chirgwin R. Two million recordings of families imperiled by cloud-connected toys' crappy MongoDB. The Register. https://www.theregister.co.uk/2017/02/28/cloudpets_database_leak/ . published: Accesed April 2017
[29] Pauli D. IoT worm can hack Philips Hue lightbulbs, spread across cities, The Register. http://www.theregister.co.uk/2016/11/10/iot_worm_can_hack_philips_hue_lightbulbs_spread_across_cities/. Published: 2016-11-10. Accessed: April 2017
[30] Krebs B. Hacked Cameras, DVRs Powered Today's Massive Internet Outage, KrebsonSecurity. https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/. Published: 2016-10-21. Accessed: April 2017
[31] Kan M. Chinese firm admits its hacked products were behind Friday's DDOS attack. computerworld.com. http://www.computerworld.com/article/3134097/security/chinese-firm-admits-its-hacked-products-were-behind-fridays-ddos-attack.html. Published: 2016-10-23. Accessed: April 2017
[32] Philips. https://developers.meethue.com/documentation/how-hue-works. How Hue Works. Accessed 04-2017
[33] Philips. http://www2.meethue.com/da-dk/om-philips-hue/#aboutthesystem. Intelligente lysstyringssystemer. accessed: 04-2017
[34] Nest. https://nest.com/about/. Nest is home. accessed: 04-2017.
[35] Nest. https://nest.com. accessed: 04-2017
[36] Smarter. http://smarter.am/coffee/. Accessed 04-2017
[37] Smarter. http://smarter.am/support-coffee/. Smarter Coffee. Accessed: 04-2017
[38] Nespresso. https://www.nespresso.com/dk/da/kaffemaskine/prodigio. Nespresso Prodigio Kaffemaskine. accessed 04-2017
[39] Raspberry Pi FAQ, https://www.raspberrypi.org/help/faqs/#introWhatIs, accessed: October 2017
[40] General Python FAQ, https://docs.python.org/3/faq/general.html#what-is-python, accessed: October 2017
[41] Python Package index, https://pypi.python.org/pypi?:action=browse&show=all&c=460, accessed: October 2017
[42] Rodrigues A., RESTful Web services: The basics, https://www.ibm.com/developerworks/library/ws-restful/index.html, 11-06-2008, accessed: October 2017
[43] What is MongoDB, https://www.mongodb.com/what-is-mongodb, accessed: October 2017
[44] NodeJS, About NodeJS, https://nodejs.org/en/about/, accessed: October 2017
[45] NodeJS, Anatomy of an HTTP Transaction, https://nodejs.org/en/docs/guides/anatomy-of-an-http-transaction/, accessed: October 2017
[46] Digital Ocean, Droplets, https://www.digitalocean.com/products/compute/, accessed: July 2017
[47] English Oxford Living Dictionaries, metadata, https://en.oxforddictionaries.com/definition/metadata, accessed: July 2017
[48] DARPA Internet Program, RFC 791, September 1981, https://tools.ietf.org/html/rfc791, accessed: July 2017

Appendix A[62]



PSA Description	Benefits for IoT
Bandwidth control	Due to resource restrictions of IoT devices, a bandwidth control facilitates to save available resources, i.e., send/receive capability and batteries. Thus, bandwidth control directly supports availability. However, it is notable that bandwidth control is not able to mitigate all threats pertaining to availability, e.g., DDoS.
Bro-logging	Logging can be seen as an enabler to recognise and actively prevent attacks, e.g., data leakage. In an IoT context logging, and recognition of suspicious traffic, can be achieved more easily when compared to traditional IT systems because traffic profiles are more static changes are mostly user driven. Logging supports data confidentiality and system integrity.
Bro-malware	Malware for IoT devices already exist [11]. However, it can be assumed that the peak of IoT malware is not yet visible, even though it is said that "IoT is the new Windows XP -malware's favourite target" [12]. Thus, it is clear that malware detection is needed in IoT devices and environments. Malware detection can pertain to all security objectives, i.e., Confidentiality, Integrity, and Availability (CIA) triad.
iptables	iptables can be applied as a firewall solution for IoT devices. In other words, incoming and outgoing traffic can be analysed based on predefined rule set in order to identify traffic types and / or destinations that are not acceptable.
Re-encrypt	Re-encrypt feature ensures that the best possible encryption is applied for network payload (with a man-in-the-middle proxy). In other words, if IoT device sends data without encryption and receiving device supports, e.g., TLS, then TLS is automatically used. This supports communication confidentiality.
VPN	Offering VPN connections is a way to offer a secure communication for IoT devices over an untrusted network. Applying VPN connection is able to reduce man-in-the-middle attacks, support privacy, and communication confidentiality.

Appendix B

Non Functional Requirements

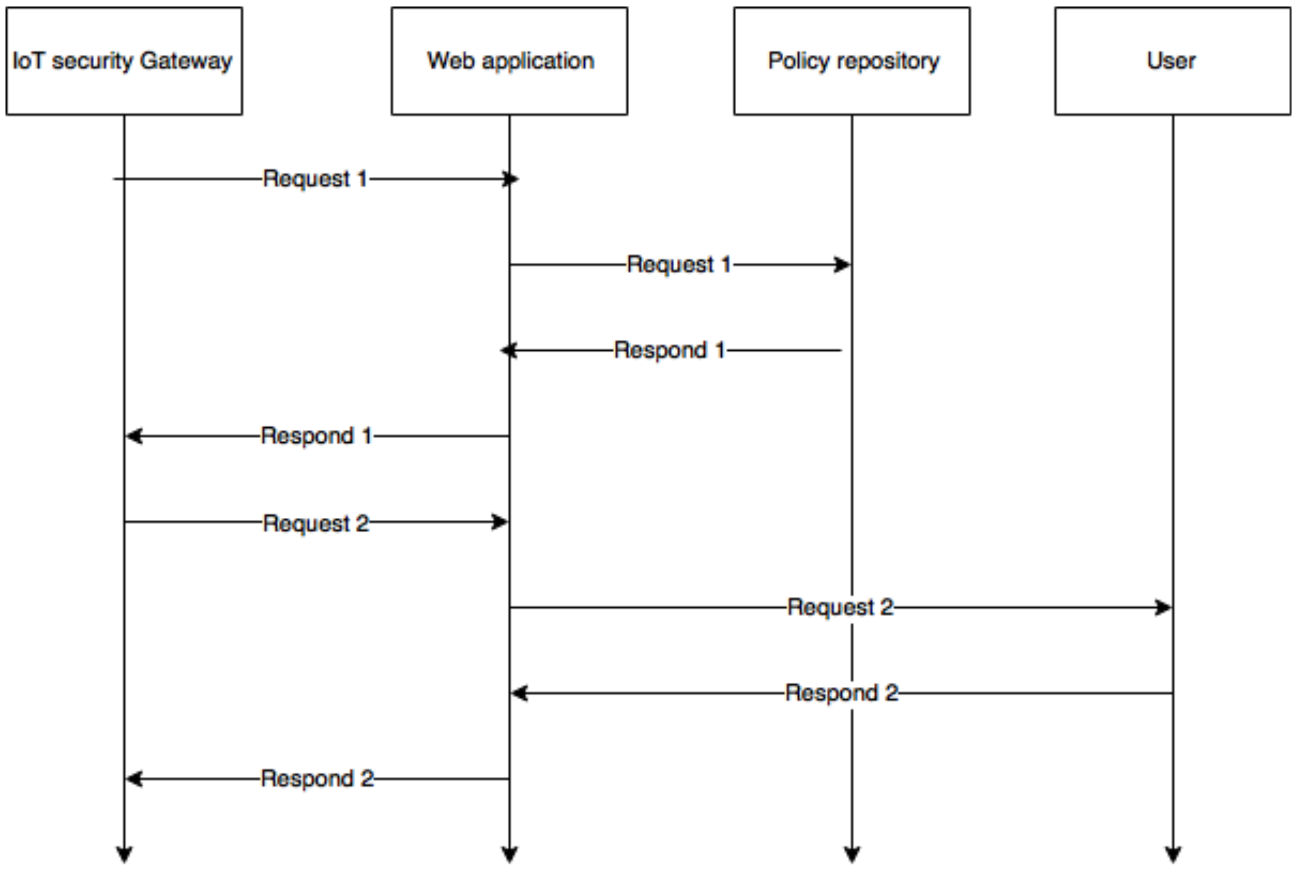
Gateway	
Requirement ID	Specification
NFGW1	The gateway must contain connection through WiFi and Ethernet
NFGW2	The Gateway must be connected to the home router
NFGW3	The gateway must run on a linux platform
NFGW4	The gateway must have internet connection

Policy	
Requirement ID	Specification
NFP1	The policy must be stored in easy updatable format

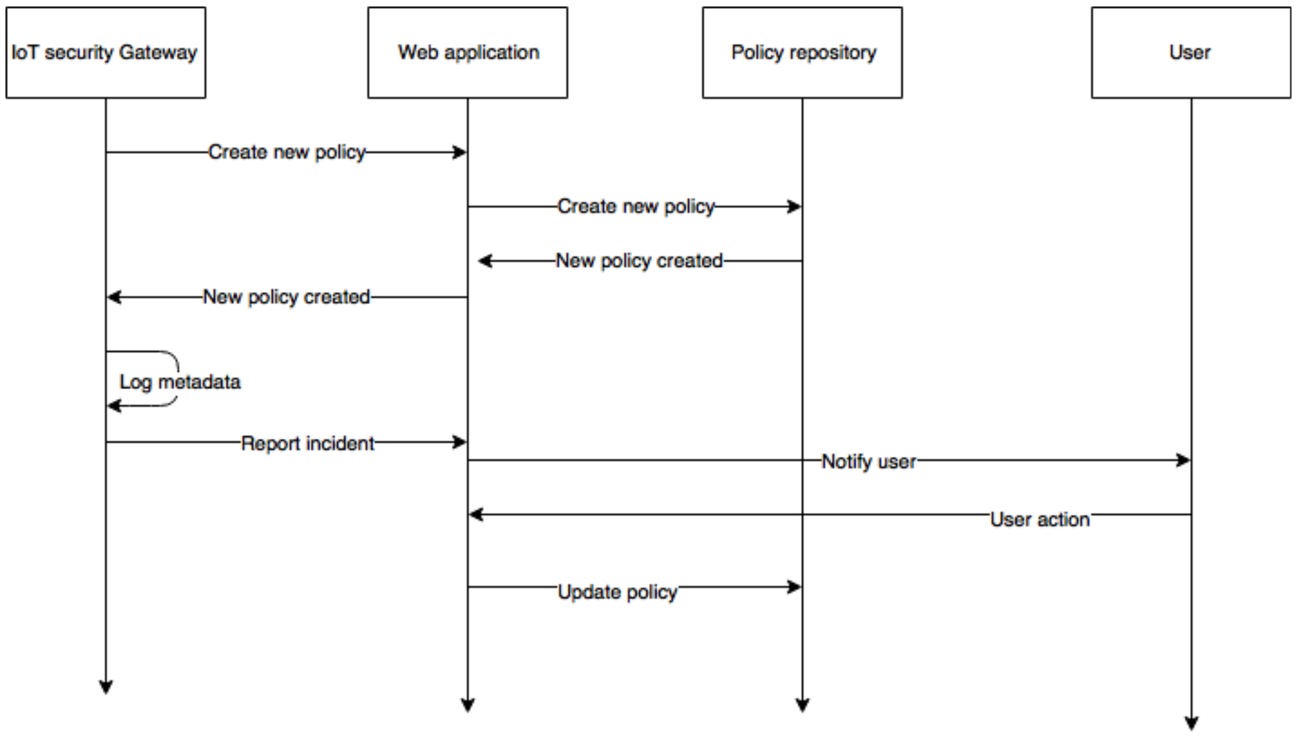
Policy Repository	
Requirement ID	Specification
NFPR1	The policy repository (PR) must be stored so its accessible to all gateways

Appendix C

Generic communication flow between entities



Appendix D
Sequence diagram of how to invoke training period
Prerequisite: No known policy exists


IoT security Gateway
Web application
Policy repository
User
Create new policy
Create new policy
New policy created
New policy created
Log metadata
Report incident
Notify user
User action
Update policy