NTNU
Norwegian University of
Science and Technology

Cybersecurity in the Internet of Things

Kristian Weium Lange

Master of Science in Communication Technology
Submission date: June 2016
Supervisor: Jan Arild Audestad, ITEM
Co-supervisor: Anders Lund, EY
Aleksander Furnes Mallasvik, EY

Norwegian University of Science and Technology
Department of Telematics

Title: Cybersecurity in the Internet of Things

Student: Kristian Weium Lange

Problem description:

The Internet of Things (IoT) offers new and exciting possibilities in many parts of the society, and in the area of business, many companies and industries could benefit from adopting the IoT and integrate it into everyday operations or products and services. Potentially, companies could improve in areas such as quality, efficiency, and cost as IoT systems can be utilized or integrated into many types of operations or products. While some companies have the necessary competence and resources to develop and maintain such systems, many must engage businesses that specialize in the area of IoT to utilize its potential. Therefore, providers of solutions for IoT can allow their customers to focus on their core activities while they ensure that IoT systems, services or products are functioning according to customer expectations.

The concept of utilizing the IoT in operational technology is relatively new and unexplored territory, and there is a growing concern in the community about the security of connecting ”everything“ to the Internet. Consequently, there could be potential dangers related to directly connecting the operations of a company to one or more IoT systems. Should a company have its IT-systems compromised or disabled, and this system is essential for maintaining correct services, the consequences could in many cases be fatal.

This thesis should evaluate the possibility for IoT systems to operate in a solution that could be offered by a service provider, and further attempt to assess the security of both IoT systems in such a solution and generic IoT systems. By evaluating end-nodes, infrastructure, and data processing of IoT systems, and by applying information security theory, analyses of the attack surface of IoT systems should be presented. Furthermore, as availability is a primary concern of IoT systems used in critical infrastructure (for example water, electricity, and transport), this thesis should also employ graph theory in an attempt to identify how the structure of IoT systems affects their availability.

Responsible professor: Jan Arild Audestad, ITEM

Supervisor: Jan Arild Audestad, ITEM

Anders Lund, EY AS

Aleksander Furnes Mallasvik, EY AS

Abstract

Increasingly, the Internet of Things (IoT) is adapted by businesses to improve operations, processes, and products. This thesis presents a possible structure where IoT systems may utilize a common platform for connectivity, processing, and user interaction. By the use of graph theory, this structure is analyzed to identify the robustness and vulnerability of the IoT systems, as their availability could be essential to preserve. Furthermore, the thesis assesses and analyses the attack surface of generic IoT systems by studying the overall exposure the components in a system have to the surroundings. It also evaluates the various technologies and services that may be used by the components from a security perspective. The thesis introduces possible security mechanisms to give an understanding of how an IoT system can react to the identified attack surface. In the study, it is found that structural dependencies between IoT systems could pose a significant threat as a single point of failure is introduced. Multiple IoT systems can be attacked simultaneously when they share such a point. Additionally, the elements of the threat landscape that IoT systems face today have been identified. Also, significant threats have been described generally and through the use of possible attack vectors.

Sammendrag

I økende grad adapterer selskaper Tingenes Internett (IoT) til å forbedre drift, prossesser og produkter. Denne oppgaven presenterer en mulig struktur som tillater IoT-systemer å benytte en felles platform for tilkobling, prossessering og brukerinteraksjon. Ved bruk av grafteori analyseres denne strukturen nærmere, slik at robustheten og sårbarheten til IoT-systemene kan identifiseres, da det kan være kritisk å bevare tilgjengligheten til disse. Videre vurderer og analyserer oppgaven angrepsoverflaten til generiske IoT-systemer ved å studere den samlede eksponeringen komponentene i et slikt system har mot omgivelsene. Den evaluerer også de ulike teknologiene og tjenestene som kan brukes av komponentene, fra et sikkerhetsperspektiv. Oppgaven presenterer mulige sikkerhetsmekanismer for å gi en forståelse av hvordan et IoT-system kan agere på den identifiserte angrepsflaten. I studiet er det funnet at strukturelle avhengigheter mellom IoT-systemer kan utgjøre en betydelig trussel, ettersom et enkeltpunkt for feiling kan bli innført. Flere IoT-systemer kan bli angrepet samtidig når de deler et slikt punkt. I tillegg har det blitt identifisert elementer ved trusselbildet som IoT-systemer står overfor i dag, og betydelige trusler blitt beskrevet både generelt og ved hjelp av mulige angrepsvektorer.

Preface

This master's thesis finalizes five years of studying at the Norwegian University of Science and Technology (NTNU) and is submitted to the Department of Telematics (ITEM) as a requirement for fulfilling a Master of Science in Communications Technology.

I would like to thank my supervisor and responsible professor Jan Audestad for giving valuable guidance and feedback, almost continuously, during this project. From EY, I would like to thank Anders Lund for all the advice, answers, and suggestive questions I have received during our meetings, and Aleksander Furnes Mallasvik for the important input during the development of the problem description and for realizing our collaboration.

Finally, thanks to Lars Nedberg and Eirik Fosser for proof-reading this thesis.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background and Motivation

To gain a competitive advantage, businesses in multiple industries and markets strive to adjust their operations to improve quality, reduce costs, and increase efficiency, for example. By doing so their products or services might become more attractive to potential customers. Due to the rapidly developing digital world, many businesses look towards Information and Communications Technology (ICT) for solutions that might help to gain this competitive advantage.

Over the last few years, the Internet of Things (IoT) has gradually been adapted into products and services such as vehicles, TVs, and traffic management. This has opened a new world of functionalities to the consumer as ”everything“ is connected to the Internet and can be monitored or controlled through, for example, a smartphone application or a web page. Furthermore, also production and control systems are evolving through the usage of the IoT. Electricity grids, beer breweries, and manufacturing lines are all examples of areas where the IoT can be used to improve processes and procedures, and make day-to-day operations more effective and precise.

Although there are obvious advantages with embracing the IoT, some precautions should also be done before connecting any physical object that surrounds us to the Internet. It is known that cyber attacks can be performed by everything from curious kids with a computer in their rooms, to an entire nation’s military cyber force. Therefore, any company that considers utilizing the IoT in a product, service, production or control system should make considerable contemplation and planning before developing and releasing anything that is Internet-connected.

Clearly, there is a great potential in connecting health equipment to the Internet such that doctors and automated processes can monitor a patient’s health contentiously, but who would like to have a pacemaker that can be controlled by anyone of your neighbors? The usage of smart electricity grids are clearly a step in

the ”right direction“ as they make it easier to utilize more renewable energy sources, but imagine that an entire city's access to electricity can be shut off by an unknown source from across the globe. And who would like their railway control system to be controlled by terrorists? Or loose control of a nuclear power plant to a foreign government? What happens when the weapon industry embraces the IoT?

While the paragraph above might seem to exaggerate or push things to extremes, it does, in fact, mention multiple events that have occurred. The electricity grid in an entire region of Ukraine was successfully cyber-attacked in 2015 [Con16o], pacemakers have been shown to be attackable [HHR+08, Com15], and in 2015, a ”Smart Sniper Rifle“ was hacked such that it was possible to manually change its target to any given coordinates [Wir15].

Regardless of the exact reason for why these ”incidents“ took place, they illustrate that computer systems can be attacked in a way that affects human lives and potentially can have fatal outcomes. Furthermore, if connecting computer systems to the Internet, approximately 40% of the world's population are theoretically given the possibility of attempting to perform cyber attacks to these systems. The 40%, those with an internet connection, can do so from wherever they are located in the world. Consequently, this raises the need for measures to prevent cyber attacks from succeeding; cybersecurity.

Although it seems unrealistic why someone would risk connecting insecure computer systems to the Internet, there are some factors that need to be considered:

- Cybersecurity is difficult – As the forthcoming chapters illustrate, the IoT is highly complex and to be able to produce or utilize systems within it securely, access to sufficient resources is required.
- Cybersecurity is a cost – ”Sufficient resources“ means that leaders within companies need to prioritize security. In short terms, this might imply additional costs to the budget. However, in the long run, it could be significantly more expensive to not prioritize security.
- Cybersecurity is obscure – Manufacturers, leaders, consumers, and other actors does not necessarily understand the concept of cybersecurity well enough to realize the value of prioritizing it. It is first when things go wrong people tend to start asking questions.

It does, of course, exist products or services in the IoT where security does not constitute an important issue and where it seemingly might be natural or legitimate to overlook cybersecurity. Then it might be necessary to start thinking about what we accept to be right or wrong on a fundamental level. Should I be able to switch on

and off your Internet-connected lights bulbs? And should you be able to change the sound volume of the stereo in my car?

1.2 Problem description

As the IoT can be utilized by various types of industries, businesses, and users, this thesis should evaluate the possibility for IoT systems to operate in a solution that combines them on a structural level. In such a solution, the practical aspects of establishing and maintaining an IoT system is offered by a service provider that specialize in the area. Figure 1.1 presents an illustration of the potential situation. The figure contains three potential businesses, each utilizing the IoT, and an IoT provider that enables usage of IoT systems. As the figure is highly abstract, the components have no particular meaning or function.

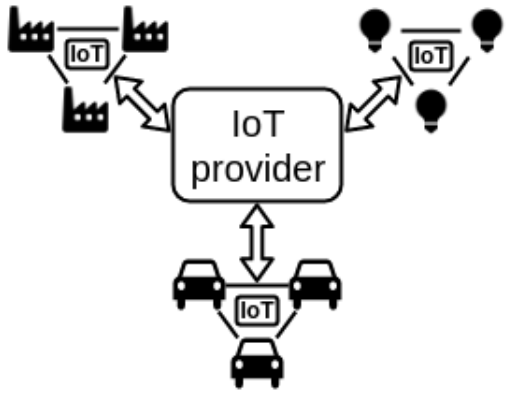

Figure 1.1: Three different businesses and one IoT provider are illustrated. The IoT provider helps the businesses to realize usage of the IoT.

To ensure that current and future computer systems in the IoT are secure and resistant to cyber attacks, it is necessary to identify the attack surface area of these types of systems such that adequate security mechanisms can be employed by those who develop or maintain them. This thesis should attempt to assess and analyze the attack surface of both generic IoT systems and IoT systems that operate combined on a structural level. It should also present state-of-the-art security mechanisms that could be used to prevent security breaches from occurring.

Furthermore, the *availability* of computer systems that operate with critical infrastructure is utterly important and may affect human lives. This thesis should evaluate how creating a structural dependency between IoT systems could affect their ability to remain accessible. The evaluation should be done by making an abstraction of the IoT systems and examine the structures they form.

1.3 Contributions

In this thesis, I present a proposal to a possible structural dependency between IoT systems that could emerge based on recent evolvement in the telecommunications market. Further, I employ graph theory to perform an analysis on how this structural dependency could affect the availability of the IoT systems as they experience random failures or targeted attacks. By looking at IoT systems from this point of view, a new and untraditional way of evaluating IoT systems is introduced.

Furthermore, I assess and analyze the attack surface of generic IoT systems and evaluate possible attack vectors to the different surface areas of such systems. By looking at an IoT system as a whole, instead of focusing on particularities of, for example, *one* possible attack vector, I give an orderly and comprehensive illustration of the threat landscape IoT systems face today. In each presented attack surface area, relevant technologies have been evaluated with respect to how they affect IoT systems. Necessary details on the technologies are included to understand why the surface areas are of interest.

Eventually, I propose relevant security mechanism to the identified attack surface and discuss advantages and challenges regarding their suitability in an IoT system. Basic concepts of the security mechanisms are included such that the general functions of the mechanisms also are described.

1.4 Limitations

Although this thesis contains multiple examples and discussions about IoT systems that exist in the real-world, no implementations have been studied in significant depth throughout this work. Also, the findings presented in the analyses have not been tested or simulated in any particular IoT system that potentially could verify or reject their validity. As any additional study of an actual IoT system would increase the workload of this thesis substantially, this was intentionally left out from the beginning. Instead, abstract IoT systems are described and used in various examples and scenarios.

Furthermore, the IoT systems that are considered in this thesis are all based on a model where the IoT devices communicate through a centralized entity. Thus, communication in mesh networks and direct communication between the IoT devices have not been specifically included, although it is covered indirectly in some parts of the analyses.

The subject of *privacy* has also been left out intentionally throughout this thesis. As the thesis is concerned with the area of attack surfaces and attack vectors, it seems appropriate to not address the general issue of privacy in IoT systems. Given that

all the discussed IoT systems in this thesis are assumed to comply with the general needs for privacy, the only way of compromising it would then be to compromise the confidentiality of the systems. Thus, only if an attack can cause a breach of confidentiality, which is discussed, then a breach of privacy would occur as a consequence.

As with almost any computer system, also IoT systems can be attacked by the use of *social engineering*. This is an attack vector that solely exploits the lack of security awareness amongst users of a system. An attacker would simply attempt to fool or lure users into revealing sensitive information about a system, and it does not directly involve any technical cyber attacks. Social engineering constitutes what is classified as an individual branch of cybersecurity and although it might be used frequently, the subject is excluded from this thesis.

1.5 Outline

In Chapter 2, basic terms regarding the IoT, a brief introduction to graph theory, and some general concepts of information security are provided as background theory. This could help the reader to better understand the analyses contained in Chapters 4, 5, and 6.

Chapter 3 introduces the concept of an IoT platform and describes both its structure and possible examples of usage. Chapter 4 continues with an analysis of the availability of IoT systems that operate on such a platform.

Then, Chapter 5 presents an analysis of the attack surface that is specific to the end-nodes of IoT systems. In Chapter 6, a similar analysis is performed, but here the security of the central component in IoT systems is discussed.

Finally, Chapter 7 gives a conclusion of the thesis and presents possible future work.

Chapter 2
Background

2.1 Definitions regarding the Internet of Things

In the following paragraphs, some standard terms and concepts are introduced with the aim of creating a foundation for better understanding the content presented in Chapter 3.

2.1.1 Machine-to-machine communication

Machine-to-machine communication (M2M) is a term used for describing communication between devices, regardless of communication channels, and can be both wired or wireless [Con16h]. M2M may well be initiated, transmitted, and received without human interaction, and is often processed directly, for example, in application software. Although specialized technologies and protocols for M2M are emerging [HTM+14], ”traditional“ communications protocols are also included in the concept of M2M.

2.1.2 The Internet of Things

The Internet of Things (IoT) can be considered as the interface between the physical and digital world that allows one to gather information from – and control – everyday objects [Mie15]. These ”things“ or objects can range from thermostats and heart rate monitors in ordinary households to sophisticated air pollution detectors in manufacturing plants. A connected device is with this referred to as an *end-node*.

Commonly, the main activities when applying the IoT are to gather, process, and present data in an autonomous manner [Ins15b]. By having end-nodes equipped with sensors and modules for communication, they can identify or measure location, temperature, and other properties, and transmit data to a particular device or server. Analysis of the data can then be performed, and valuable information can be presented to the user. In addition to sensory capabilities, the IoT also facilitates actuation. With actuation, users can operate the end-nodes remotely through giving

them instructions carried over the Internet. Examples of IoT applications where actuation is used are door locks and light bulbs.

Although related, the terms M2M and IoT should not be considered interconnected. While M2M regards links between devices on a lower layer, the IoT is more concerned about the utilization of data on the application layer. A possible relation between the two could be defined as the M2M being a component of IoT. For example, the IoT application could assume that transmission of data is handled properly by M2M.

2.1.3 Telecommunications Operators and Internet Service Providers

Although the Telecommunications Operators (TELCO) and Internet Service Providers (ISP) traditionally have been operating in two separate areas of communications technology; TELCOs in the field of mobile networks and ISPs in the field of accessing and using the Internet, these areas have increasingly become connected to each other over the past years. Today, one often finds that TELCOs also are ISPs [Con16n] as devices in mobile networks tends to be used for both mobile- and internet-related services. This may not necessarily apply the other way around.

Hence, in this thesis, any reference to a TELCO indicate a company that offers connectivity and services in the mobile domain, but also acts as an ISP. Mobile traffic generated by the TELCO's customers may then enter the Internet without immediately having to leave the network controlled by the TELCO.

2.2 Introduction to graph theory

To allow the reader a better understanding of the analysis in Chapter 4, some definitions and explanations regarding graph theory are provided. All of the following definitions are based on [Bol98] and [Aud11].

2.2.1 Network vs graph

Although the terms *network* and *graph* often are considered to be synonyms, a more precise definition would be to say that a *graph* is a strict mathematical representation of a *network*. While *graph* is more used in mathematics, the usage of *network* is more frequent in logistics, engineering, and other sciences where graph theory can be applied [Aud11].

2.2.2 The basics of a graph

A *graph* consists of a set of points called *vertices*, and a set of lines called *edges*, where each edge interconnects two vertices of the graph. Figure 2.1 is an illustration of a simple graph with eight vertices. The number of vertices is called the order of the graph. Two vertices connected by an edge are said to be *adjacent* to each other or *neighbors*. An edge can be undirected interconnecting vertices in both directions, such as in the Figure. Or the edges can be directed interconnecting one vertex to another, but not vice versa. A directed edge is also called an arc. Note that an undirected edge can be represented as two directed edges, one in each direction. A graph G_a is called a subgraph of G_b iff all the vertices and edges of G_a are contained in G_b

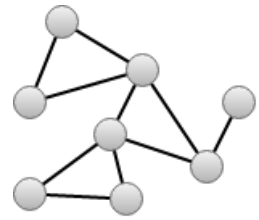

Figure 2.1: A simple graph.

In addition to having directed and undirected graphs, one also differentiate between *multigraphs* and *simple graphs*. While the vertices in a *simple graph* are connected by a single edge (or one *arc* in both directions), a multigraph allows for multiple connections between the vertices, see Figure 2.3. Two vertices may be connected by an unlimited amount of edges, which could appear redundant in plain sight, but is important when the graph is representing, for example, a telecommunications network. In such networks, redundancy is a key mechanism for ensuring that two nodes (e.g. switches) remain connected even if a link (e.g. network cable) should fail. Therefore, depending on the context, one could say that a multigraph is a more correct and accurate representation of a telecommunications network, than a simple graph.

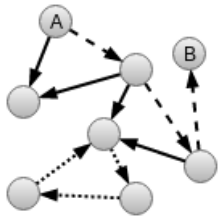

Figure 2.2: A directed graph.

A *path* is an alternating sequence of vertices and edges from one vertex to another vertex. A *cycle* is a path originating and terminating at the same vertex. It is commonly required that a cycle consists of at least three vertices. The condition for having a *connected* graph, is if a path exists between any two vertices of the graph. If the graph is directed, it is *strongly connected* if the same condition holds, or *weakly connected* if the condition holds when replacing the directed edges with undirected edges. In Figure 2.2, a *directed* graph is illustrated together with an example path from vertex A to vertex B market by the dashed edges, and a cycle marked by the dotted edges. The graph is weakly connected.

While having multiple redundant edges between pairs of vertices is one mechanism to improve the connectivity of a graph, one can also enhance the connectivity by introducing multiple paths between pairs of vertices. If adding edges that make new pairs of vertices adjacent, new paths from one vertex to another can arise in the graph. In networks where it is desirable that every node can reach all the others, such as the Internet, it is useful to have high connectivity and multiple paths between vertices. Given that a vertex or an edge is removed, the graph should preferably not split into multiple unconnected subgraphs.

The *degree* of a vertex of an undirected graph is the number of edges at that vertex. For directed graphs, we define similarly out-degree and in-degree as the number of directed edges originating or terminating at that vertex. Figure 2.3 illustrates a undirected multigraph, where the degrees of the vertices are marked on each vertex. In the case of a telecommunications network, a node with many links may be represented as a vertex with a high degree, in the corresponding graph. If the nodes of the network may fail, the equivalent action would be to remove the vertex and all the connected edges from the future. Intuitively, removing vertices with a high degree has a greater impact on the connectedness of the graph.

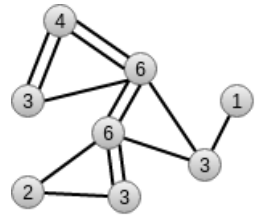

Figure 2.3: A multigraph.

2.2.3 Graph properties

To take advantage of graph theory in an analysis, there is a need for defining some measures of a graph. The measures state general properties of graphs, and can be used to evaluate whether or not a network answers to a prerequisite or demand. Note that as there is no relationship between isolated components of a graph, only fully connected graphs are considered.

Distance and diameter The minimum distance between two vertices is the number of edges in the shortest path between them. The diameter of a given graph is found by identifying the longest minimum distance between any two vertices in the graph. If looking at the entire world's population as a network of acquaintances, the distance between two vertices would be the number of acquaintances separating them. The diameter would be the highest distance between any two people on the planet.

K-core The k-core of a graph is the subgraph where all vertices have degree k or higher. When computing the core, only edges from other vertices in the core are accounted, and a k-core can be used to identify tightly connected portions of a graph. In Figure 2.4, the 3-core of a graph is identified, and the remaining vertices and edges are blurred out. Additionally, note that the 3-core in this particular example also is a *clique*. A clique is defined as a graph or subgraph where all vertices are pairwise adjacent.

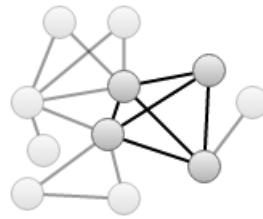

Figure 2.4: The 3-core of a graph.

Betweenness Betweenness for a vertex is defined as the total number of shortest paths that pass through that particular vertex, where the set of shortest paths considered are those between all other pairs of vertices in the graph. In the case of having multiple shortest paths between a pair of vertices, only the portion of the paths that run through the given vertex is accounted for. The betweenness could say something about a node's expected throughput in a network. Given that the flows in a network preferably follows the shortest paths and that the flows occur in an approximately random pattern, the nodes whose vertices have a high degree of betweenness could expect more throughput.

Clustering The clustering coefficient, C, of a graph is the average probability that two vertices that are adjacent to a common vertex also are adjacent to each other. From this, it follows that graphs with a high clustering coefficient often form groups or clusters of vertices at arbitrary locations. In Figure 2.5, a graph with a high clustering coefficient is illustrated. The dashed lines indicate examples of clusters in the graph.

Small world property A small world graph is often considered as a graph with a small average shortest distance between vertices and a large clustering coefficient. Although this is not a very strict definition, it provides an outline for how a small world graph looks like. In Figure 2.5, the graph has multiple clusters but a high average shortest distance, and without increasing the number of connections across the clusters, the graph is unable to satisfy the small world property. The notion *small world* comes from observing a network of acquaintances, which was done by Stanley Milgram in 1967. Milgram found that the median distance between any two living people on the planet, a graph consisting of six billion vertices, is 6 [Mil67]. The term "six degrees of separation" was introduced to describe this impressive result.

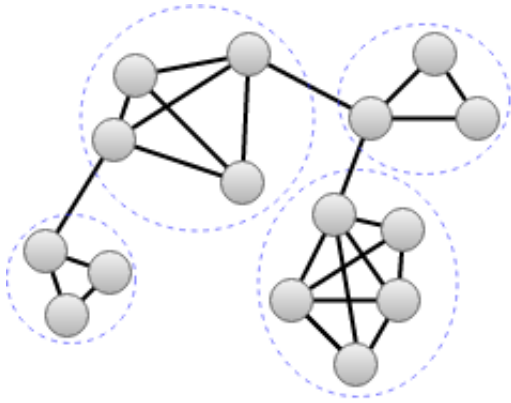

Figure 2.5: A graph with a high clustering coefficient

Degree	2	3	4	5	6	7	8	9
Number of vertices	10	0	0	2	0	1	1	1

Table 2.1: An extracted Table from [Aud11] with the degrees in scale-free graph.

Scale-freeness A characteristic of a scale-free graph or network is that the edge degree distribution of the vertices corresponds to a power law distribution [Con16l]. Consequently, it follows that the edge degree of the vertices is not concentrated around the average degree, but instead distributed over a much wider range of values. The *Matthew effect*, a phenomenon where the rich get richer [Con16i], is a good illustration of how some few nodes in a scale-free network attract many connections while the majority have a significantly lower edge degree. This is exemplified in Table 2.1, where the edge degrees of the vertices in a scale-free graph are contained. In the table, 10 out of 15 vertices have a degree of two, while 3 out of 15 have a degree of seven or higher. The term ”scale-free“ was introduced to the describe the large difference in the number of connections among nodes in scale-free networks, making the networks appear to have no scale [BB03].

Albert Barabási played a significant role in the discovery of the scale-free network concept [Con16b] and in 2003, Barabási and Eric Bonabeau explained the scale-free properties of the Internet [BB03]. Although this has been challenged by Willinger et al. in [WAD09], and other reported scale-free networks have been discussed by statistical analyses [CSN09], this thesis considers the concept of scale-free networks and graphs to be genuine.

2.2.4 Robustness and vulnerability of a graph

Failure A failure of a vertex in a graph is defined as the action of removing the vertex and all the edges connected to the vertex from the graph. When dealing with a failure of an edge, this simply involves removing that particular edge.

Resistance to failures Resistance to failures is *not* defined as the capability to avoid failures, but rather the consequence caused to the graph's connectedness by one or more failures. In this thesis, the outcome of failures can roughly be divided into two scenarios. First, is that a failure affects a relatively small amount of vertices and that the vast majority of the remaining vertices still are connected and have paths to the same vertices as before the failure. Second, is the scenario where the graph dissolves fully or partially, for example by isolating vertices into sub-graphs or disconnecting vertices that were connected before the failure(s).

Robustness of a graph When a graph is subjected to *random failures*, the graph's probability of being resistant to the failures is considered as the robustness of the graph. In a network perspective, a random failure could be a collapsing bridge in a road system or a malfunctioning transformer in an electricity grid. Furthermore, the robustness of the network is determined by the probability that the network will resist the random failure.

Vulnerability of a graph The vulnerability of a graph is similar to the robustness, but instead of considering *random* failures, the vulnerability measures how resistant a graph is to *targeted* failures, or *attacks*. As an attack is always assumed to target the vertex or vertices which cause the most impact to the graph if being removed, a graph with a high vulnerability is easy for an attacker to dissolve, given that it is possible to inflict targeted failures.

2.3 Definitions regarding information security

In Chapters 5 and 6, multiple terms regarding security of computer systems are used in the analyses of IoT applications. To give a fundamental understanding of the topic and to clarify the particular meaning of some of the used terms, this section describes some elements of information security to the reader.

2.3.1 Security objectives

Three main objectives define the heart of computer security, and these can often be applied to any computer system [Sta11]:

Confidentiality This term covers two related concepts:

- *Data confidentiality* assures that private or confidential information is not made available or disclosed to unauthorized individuals.
- *Privacy* assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Integrity This term covers two related concepts:

- *Data integrity* assures that information and programs are changed only in a specified and authorized manner.
- *System integrity* assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Availability

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

As mentioned in Section 1.4, the objective of privacy is not included in this study and is not found discussed in the further chapters. Nevertheless, the remaining topics are all evaluated in the forthcoming security analyses.

2.3.2 Concepts of cyber attacks

The below paragraphs defines terms that are useful to understand the course of how an attacker can be able to compromise computer security. The definitions are extracted from [Shi].

Vulnerability A vulnerability is a flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy.

Intelligent threat (or just threat) An intelligent threat is a circumstance in which an adversary has the technical and operational capability to detect and exploit a vulnerability and also has the demonstrated, presumed, or inferred intent to do so.

Attack An assault on system security that derives from an intelligent threat, i.e., an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Today, most computer systems experience being targets to some kind of cyber attack. While many attacks are avoided or prevented, some attacks are successful and allow an attacker to compromise the confidentiality, integrity or availability of the system or data that are being attacked.

Briefly summarized; to be able to perform a successful attack, an attacker could start by attempting to identify *vulnerabilities* of the target. Furthermore, a *threat* arises if the attacker gains knowledge of a vulnerability, understands how to exploit it, and also has the capability and intent of doing so. Eventually, the target is *attacked* when a deliberate assault is performed as the attacker attempts to exploit the vulnerability of the system

2.3.3 Computer networking

Network service A network service is defined as an application running at the application layer, that provides data storage, manipulation, presentation, communication, or other capabilities [Con16k]. Typically, a network service is provided by a *server component* running on one or more computers, and accessed by a *client component* running on other devices. Although various architectures can be used for a network service, the most common are the client-server and peer-to-peer architectures. Examples of network services are Domain Name System (DNS), World Wide Web (WWW), File Transfer Protocol (FTP), and Secure Shell (SSH).

Web interface In this thesis, a web interface is regarded as a graphical user interface (GUI) which can present data and functionality to humans through a typical web-browser. The web interface could utilize multiple network services that run in the background. These services might be triggered by incoming events or trigger events due to interactions in the GUI.

Chapter 3
An IoT platform for digitizing operations and procedures

To illustrate how various companies could employ the IoT in their day-to-day operations, this chapter describes an IoT platform and exemplifies how it could be utilized. First, it is important to understand that the IoT platform is managed and operated independently from the companies who utilize it and that it belongs to a separate business area. In fact, the companies are customers of the IoT platform, and they buy a service that helps to digitize their operational technology. Second, the platform can handle multiple customers from all types of business areas, despite that they have a wide range of core activities. This is achieved by providing necessary processing capabilities and IT-infrastructure to be used for communication with the devices the customers employ in their own systems. An *application* can be considered as a particular service being performed by the platform, on demand from a customer. Examples of such applications are performing smart meter readings in electricity grids, managing water supply systems, and communication between a car manufacturer and its vehicles. Some example applications are described in more detail in Section 3.3.

3.1 The IoT from a telecommunication operator's view

With the emerge of the IoT, the need for connecting end-nodes of applications to networks such as the Internet also increases. While many IoT applications have end-nodes that are stationary and placed in locations where steady connectivity is available, other applications depend on access technology with a wider range. Applications could require that the end-nodes can remain connected while they move or are moved. It is at this point the telecommunication operators (TELCO) can play a significant role.

TELCOs are in a unique position where they are able to offer M2M over both great distances and to moving end-nodes, as they already own extremely costly infrastructure. This type of M2M can be achieved by using mobile network technology,

such as 3G and 4G. As M2M is no different from traffic generated by a smartphone or a computer, for example, the TELCOs can serve a huge variety of IoT applications as long as their end-nodes can transmit data over common network interfaces.

In addition to having mobile networks, TELCOs often own fiber optic cables that, either fully or partially, are a part of a backbone network. Through these cables, TELCOs are able to offer connectivity with a certain quality of service, and to some IoT applications, this could be imperative. Ultimately, this short section tries point out why TELCOs are in a very special position regarding M2M, and this is used as an inspiration in the following sections on the IoT platform.

3.2 An IoT platform

Today there are multiple providers of different types of IoT platforms. Take for example Amazon and Microsoft; these companies offer their customers interfaces for storing, reading or modifying data in the ”cloud“, together with the possibility of performing processing services or computations on the data [Inc16, Cor16b]. Many of the IoT platform providers focus solely on the same services as Amazon and Microsoft, and offer little attention to how the IoT end-nodes connect with their servers. However some actors, often TELCOs, own or manage infrastructure that enables them to offer both connectivity and processing. In the following sections, an IoT platform consisting of both processing capabilities and M2M infrastructure is presented. The following definition uses Telenor's Connected Objects report [AGS09] as a starting point, but is adapted accordingly to the development and demand of the market, as well as taking technological innovations into account. A brief look at Figure 3.1 is recommended to gain an impression of the structure of the IoT platform.

3.2.1 Processing center

The processing center is the brain of the applications. It is here data is transformed into information, users read their application's status, and instructions to the end-nodes are generated, and while some of the processes are fully autonomous, others might require user interaction. The center is made up of multiple physical servers that allocate virtual servers to the applications running on the IoT platform. Also, the processing center could be operating in multiple physical locations and function in a distributed manner if certain applications are highly sensitive to response time. If there is a high demand for operational stability and redundancy is needed, a split solution could be necessary for safety reasons. However, for now, the processing center is considered to be placed at one single location.

Although the actual functionalities of the processing center rely on the types of applications running on it, there are some fundamental capabilities which need to

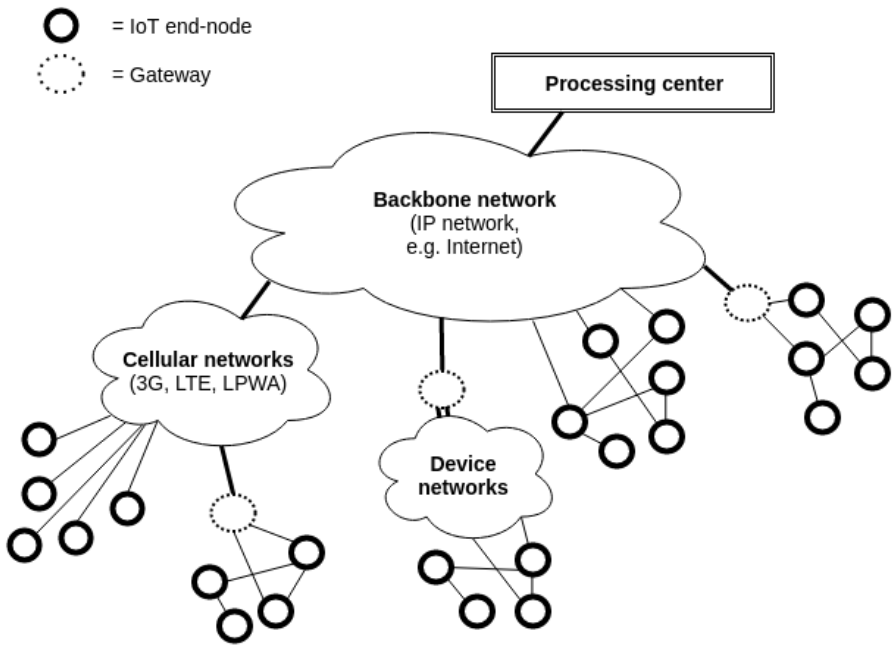

Figure 3.1: An IoT platform with infrastructure, a processing center, and a visualisation of the ways end-nodes can achieve connectivity. The figure is inspired by [AGS09].

present in order to execute logic for typical IoT application [AGS09].
First, the center must have the basic set of communication capabilities with the end-nodes of an application. This includes receiving data at both expected and unexpected moments of time, performing remote invocations at the end-nodes, scheduling of remote invocations to avoid large traffic peaks, and managing the operational status of the end-nodes.
Second, after receiving raw data, it is important that the center is able to process, store, or accumulate the data according to the application logic. This can include execution of algorithms, database integration, or taking use of plug-ins. Other factors such as time management and data backup management could be of importance for some applications, but might not be critical in general.
Third, the status of the application should be accessible to its owners or other involved actors. Therefore, the processing center must support interfaces for displaying the data gathered from the end-nodes, preferably as information which is formatted, filtered, or interpreted. These interfaces should not only support reading information but also allow for user control, for example if actuation is a part of the application.

While the capabilities mentioned above should all be present in the processing center, they do not serve as an exact specification for how the center must be built up. The specifications are included with the purpose of giving a small introduction on how the processing center operates. The application software running in the processing center can belong to either the IoT platform provider or the customer. This depends on the particularities of the services offered by the provider.

3.2.2 Infrastructure

As a part of defining an IoT platform, this section covers how the infrastructure of the platform is built up. The infrastructure is considered to be the networks and connections that allow the IoT end-nodes to communicate with each other and the processing center. This is illustrated in Figure 3.1, where all elements between the ”Processing center“ and the ”IoT end-nodes“ can be considered as infrastructure.

The main component of the IoT platform's infrastructure is the backbone network. All traffic to and from the processing center have to, at some point, travel within this network. Although IoT end-nodes can be directly connected to the backbone network, many applications will have the end-nodes placed behind some gateway, for example, a router using WiFi technology. Also, dedicated device networks or personal area networks can be used to connect the end-nodes to the backbone network. Bluetooth is a common technology for this purpose, and networks using Bluetooth or similar technologies enable end-nodes to share resources, such as connectivity, amongst each other. An end-node can also connect with the infrastructure through mobile technology. 3G, LTE, and LPWA networks are the access technologies expected to grow the most by 2020 [CS16] and are included in Figure 3.1. LPWA (Low-Power Wide-Area) is particularly interesting in this context, as these types of networks are and will be designed especially for M2M.

While this section contains few details on how the infrastructure is used in practice, Section 3.3 presents multiple IoT applications that potentially could run on the platform and exemplifies how the infrastructure could be utilized.

3.2.3 End-nodes

The end-nodes are devices that serve a specialized purpose in the IoT platform. They can be simple and perform trivial tasks such as measuring temperature, or they can be advanced and integrated into complex machinery, for example. Typically, the end-nodes are acquired and owned by the companies and are connected to the IoT platform's infrastructure when a customer relationship is established. While their capabilities could range from simple sensing to having complex control functions, for them to work properly within the IoT platform they must have a network interface and be configurable. Possible types of configurations could be support

for inserting a SIM-card, entry of destination address for data traffic, or specifying security parameters. As Section 3.2.2 describes, connectivity could be provided by IP networks, mobile networks or proprietary device networks connected to a gateway. As long as the end-nodes have an interface for communicating within any of these, the nodes can take part in the application running on the platform.

Besides accomplishing connectivity to the processing center, the end-nodes can have all sorts of functions, but are often constrained by certain factors. If not connected to the electricity grid, the end-nodes have to run on battery and should ration this to avoid costly and time-consuming battery changes. The devices might also be constrained in terms of available bandwidth and must adapt accordingly for the application to work without blocking the network. In addition, if the end-nodes' application is designed for handling real-time data, limited bandwidth could affect the performance of the application itself. Other factors that could affect the end-nodes are having limited computational power or the effects of operating in a harsh physical environment.

Summarized, there are many challenges regarding making the end-nodes function in the IoT. However, many companies are working intensively on making the end-nodes better and better and frequently push improved solutions into the market. An interesting question, that probably too few think about, is whether or not these companies also consider information security.

3.3 Example applications

While the sections above try to describe the IoT platform through the concepts of having a processing center, an infrastructure, and end-nodes tied together, this section instantiates two applications which potentially could run on the IoT platform. As previously mentioned, the IoT platform is designed in a way that allows multiple applications to run simultaneously and to share the different parts of the infrastructure and processing center.

3.3.1 Smart meter in electricity grid

By introducing smart meters in electricity grids, a vast amount of new possibilities arises which could benefit both the electricity supplier, the customer and also the environment. The potential of installing smart meters in homes is to such an extent that authorities in Norway have decided that all homes and metering locations should have one installed by the beginning of 2019 [Lov11]. However, as utilizing this technology to the fullest requires competence which usually lies beyond the core activities of an electricity supplier, employing a provider of an IoT platform might be a reasonable alternative. The following paragraphs describe how this example

application is carried out, where an electricity supplier would be the typical customer and application owner.

End-nodes The end-node, or the smart meter in this application, is a device attached to the electricity entry point of a home or metering location. Multiple types of smart meters already exist, some more advanced than others, but adapted from [Lov11] the smart meter in this example application has the capability of performing measurements of electricity consumption and transmitting the metering values over a mobile technology communication interface. The devices have a Universal Integrated Circuit Card (UICC) installed for this purpose. From being placed on the electricity entry point of a metering location, the smart meters also have the capability of breaking or reducing the power outlet to the local electricity system, if such an instruction is given.

Infrastructure As the specification of the end-nodes implies, the application utilizes mobile networks to connect the end-nodes to the remaining infrastructure. Furthermore, a gateway between the mobile network and the backbone network, which in this application is the Internet, ensures correct data transmission to and from the processing center. This solution ensures that it is the IoT platform that is responsible for managing the connectivity of the end-nodes and eliminates the need for a private connection to the Internet on the metering location. Figure 3.2 illustrates some details on the peripheral parts of the infrastructure in the application, where the houses and buildings represent the metering locations, each having an individual connected smart meter. The two large antennas are the base stations in the mobile network.

Consequently, the infrastructure is made up of wireless links from the smart meters to the data carrying, mobile network. From the mobile network, the traffic is routed to the Internet and forwarded to the processing center. At this point, the application logic handles the packets, and they ”leave“ the infrastructure.

Processing center The processing center has multiple functionalities and can be further developed depending on the application owners requests. The most fundamental function of the center is to receive smart meter readings from the end-nodes and store the readings so that they can be used in scheduled calculations. At given intervals, or upon a request, the processing center performs these calculations and provides near real-time overviews of the consumption in the electricity network. This information can be very useful to electricity suppliers, and could, for example, be used to more easily include power plants that utilize renewable energy sources in the power grid. Control of the network consumption is key to taking advantage of renewable energy sources, as these types of plants often have an unstable delivery of power. Another fundamental function of the processing center is to keep track

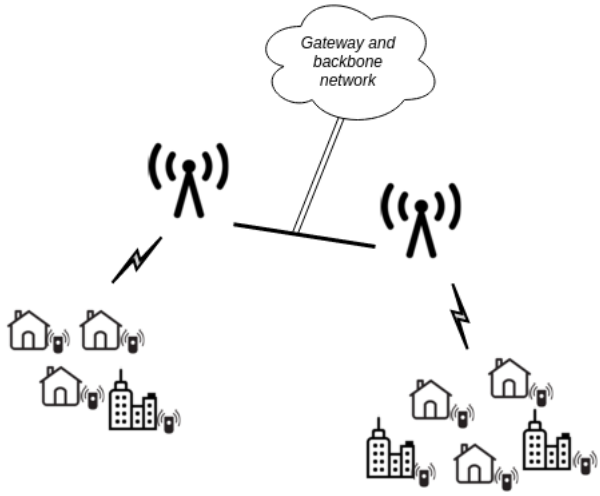

Figure 3.2: Mobile network technology is used for connecting the smart meters to the remaining infrastructure.

of the end-nodes of the application. If a node is not providing data, the processing center should become aware of this and initiate a procedure for managing faults in the peripheral parts of the application.

By authenticating, the customers can also access the processing center to see their own consumption and browse their meter readings. As the data is already there, the application logic needs only to query it to extract the information and present it to the users. Additionally, the electricity suppliers can introduce ”smart pricing“ which involves adjusting the price of the electricity correspondingly to the accumulated usage in the network. Through this practice a more stable demand for electricity can be created, as people will pay less for the electricity they consume it in the ”low demand“-hours. Regardless of pricing strategy, billing is far easier for the electricity supplier as data is collected automatically and at correct times.

While the mentioned capabilities of the smart meter application are the main focus for further discussion, the smart meter could also perform tasks like detecting errors in the local electricity system which it monitors or alert individual consumers in the case of a power outage that affects them. Ultimately, the smart meter can be built as advanced as technology allows, and other resources, such as gas and water, could also be metered by the same device. And what prevents connecting burglar/fire alarms to the same smart meter? It is necessary to acknowledge the fact that smart meters potentially could control multiple important segments of a house or site, and bring this into account when thinking about the security of the application.

3.3.2 Connected healthcare equipment

Hospitals and hospital equipment are predicted to be areas where the IoT will have a vast growth the forthcoming years. Multiple actors have already entered the field [AB16, plc16] and are attempting to create solutions for increasing quality and efficiency in the treatment of patients. By connecting equipment which monitors and assists patients, both time and money could be saved, as well as making the diagnosing of the patients better and more accurate.

End-nodes For this application, the end-nodes are assumed to be of two types. First, a monitor device that measures a patient's heart rate and body temperature. This device performs simple sensing and displays the data, as well as transmitting the same data through a network interface. Second, is an infusion pump that has settings for the rate and amount of fluid, for example medicine, that is to be infused into a patient's circulatory system. While this device can perform sensing of the remaining level of medicine, it can also adjust its settings for medicine infusion according to instructions which are given by manually pressing a panel. Also, the rate and amount can be configured through a network service that receives requests with particular parameters. The network interface of these devices uses WiFi technology and is thus able to communicate over common wireless links.

Infrastructure For the end-nodes to work properly, this application relies on having a WiFi connection on the location where the devices are to be used. When connected, the equipment transmits packets to the default gateway of the WiFi connection, and here the packets are forwarded into the backbone network. As with the smart meter application, the backbone network used in the health equipment application is the Internet. However, in this scenario, connectivity to the Internet is not fully managed by the IoT platform, and a private Internet connection has to be present at the WiFi connection's default gateway. Further, this implies that the end-nodes not only can be used in hospitals or other health institutions but also in patients' homes, as long as the prerequisites are fulfilled.

Service platform When the devices transmit data to the processing center, which is done at given intervals, the center stores this and is able to track changes in heart rate, temperature, and medicine consumption over time. By linking the equipment to a patient register, health personnel can have a near real-time record of patients' status, and can access the information from any location. To protect the privacy of the patients, the processing center implements access policies which ensure that only those who need information about certain patients are able to see it. Upon accessing this information, the user can see charts and diagrams to gain an overview of the patient's development for a given time period, for example through the night. Should a doctor access the processing center and, based on the "online and live health

record“, see that a patient has an obvious need for a changed amount of medicine, this can be achieved by adjusting the parameter of the connected infusion pump through the processing center's interface. The application logic and the infrastructure will then generate and transmit proper instructions to the end-node at the patient's location, and store all the performed actions in logs.

By using an application for connected health equipment on an IoT platform, the need for manually logging and storing records is reduced and potentially removed. It could open for more efficient and correct treatment of patients, and also to some extent allow patients to receive treatment where they desire. While only two types of devices are introduced here, the principle would be the same for any other equipment, and when diagnosing patients in the future, this could possibly be done based on a much higher amount of factors and perhaps even automatically?

3.3.3 Need for security

Although adapting the IoT can have multiple positive effects on the society in general, there is a need for some degree of moderation. In many industries and businesses, the consequences of failures can be vital, and if these actors embrace the IoT without being cautious, there are multiple fatal outcomes. As briefly mentioned in the introduction, there was conducted a cyber attack on a Ukrainian power supplier's IT-systems in 2015, causing about 700'000 Ukrainians to be left without electricity for six hours, in the end of December [Con16o]. This event tells two things; first, attacking IT-systems used in critical infrastructure is possible, and second, there are people or groups with the capacity and will to perform such attacks. When applying this to the IoT platform and the provided examples, the motivation for considering security should be quite clear. If those with wrong intentions obtain the ability to control other people's infusion pump, shutting down electricity and water supply, or control signaling systems in road and railway traffic, lives are in danger.

From the previous sections of this chapter, one get an insight to the complexity involved with IoT application and an IoT platform. As multiple technologies, networks, interfaces, hardware components, and other elements are supposed to work together seamlessly, securing such computer systems is a tough task. Potentially, an attacker could exploit the tiniest vulnerability to compromise the entire application or platform. Therefore, it is important that all ”gates“ are closed to prevent breaches of security from occurring. It does not help to seal the ”main entrance“ if the ”back door“ is wide open. The challenge with IoT applications is that there are doors everywhere that need to be shut, and as an attacker's mindset often can be described through the expression: *to pick the low-hanging fruits*. An attacker would regard any open door as good enough, as long as it leads inside.

An interesting feature of the IoT platform is that multiple customers utilize, and

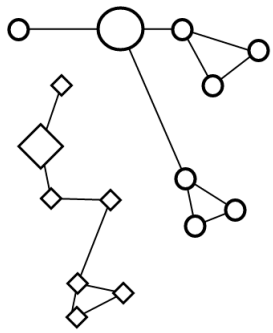

Figure 3.3: Two abstract graphs of seemingly separated businesses.

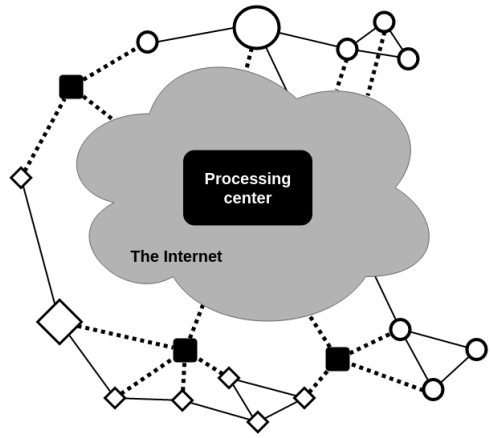

Figure 3.4: A simplified illustration of the same IoT platform serving two separated businesses.

also *rely*, on the same IT-infrastructure and the same network nodes in order to have their own operations function properly. Consequently, businesses with completely unrelated service areas and non-overlapping supply chains, suddenly have a common relation in the IoT platform they utilize. Figure 3.3 and 3.4 illustrate this by using two businesses, circle and diamond shaped, as examples. In Figure 3.3, the network of the core activity for each business is drawn highly simplified. While the lines could be structures such as roads or water pipelines, the shapes could be road intersections or pumping facilities. In the first figure, the businesses are seemingly unrelated regarding operations and dependencies.

However, after digitizing their operations through engaging the same IoT platform provider, for example by adding connected traffic lights or water pumps, they find themselves relying on many common components in the network. In Figure 3.4, the dotted lines now represent connectivity in the IoT platform's infrastructure. The solid shapes represent components that allow the end-nodes to access to the platform. What the figure means to express is that the two previously unconnected businesses suddenly have become parts of a network that appear to connect them rather explicitly. Assuming that the IoT platform is essential for maintaining their operations, what happens if an attacker starts attacking the solid shapes in the figure? And how severe could the consequences become if the attack affects water, electricity, hospitals, and railway all at once? In Chapter 4, this mutual dependency is analyzed using graph theory.

Chapter 4
A structural analysis of an IoT platform, based on graph theory

In this chapter, the IoT platform is studied and discussed with the help of graph theory. This is done in an attempt to clarify how its structure affects the robustness and vulnerability, which again can affect the overall availability of the platform. Because the IoT platform can serve multiple applications simultaneously, and because some applications might be used to supervise and control critical infrastructure, the availability of the IoT platform could be excessively important. This analysis is quite brief but aims at pointing out a new direction for how IoT systems can be analyzed. No actual IoT applications have been regarded in the analysis, but instead, simplifications have been made when analyzing the structure of the IoT platform. Three particular areas are regarded – the applications' dependence on the processing center, the behavior of the Internet, and possible enhancements of the processing center's structure.

4.1 A star topology

From simple reasoning, one finds that the Internet consists of sub-networks formed as stars practically everywhere. The simple network topology, see Figure 4.1, consists of a central node with multiple connected outer nodes. The topology occurs in homes, offices, mobile networks, for example, and are instantiated by routers, switches, base stations, and other devices that merge and split networks. An immediate observation of the star topology is that if the central node fails or stops working, the remaining nodes become isolated and unable to communicate. For example, in a mobile network, a failure of a base station could leave a physical area without service. Or a neighborhood could be left without a connection to the Internet if a large switch malfunctions in an ISP's network. By assuming that it does not exist redundant components in these examples, the behavior of a star shape is emphasized; if the central node is removed, the remaining nodes are left isolated.

Stars can also be formed by looking at networks from a broader perspective.

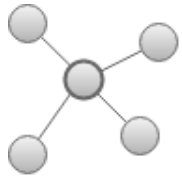

Figure 4.1: A simple star topology.

Take for example a power plant that delivers electricity to every house in an entire city. Although there probably would be an advanced power grid connecting the houses to the plant, the houses' dependency on the power plant can be simplified and illustrated by a star topology. Every house needs the power plant to operate if they are to have electricity[1]. Hence, a failure of the central node would leave the outer nodes isolated, or without electricity in this case. Arguably this broad perspective approach could also be applied to, for example, computer networking, where the *client-server model* [Con16d] is a natural example.

4.2 The IoT platform as a star

As Chapter 3 describes, the IoT platform can serve multiple applications and provide connectivity for the end-nodes in multiple ways. However, if the details of connectivity and infrastructure are removed, the remaining parts of the platform form a star. The star, illustrated in Figure 4.2, consists of the end-nodes of multiple applications, which all communicate with the processing center. Although the figure only contains two applications, when in reality there could have been many more, the message still applies; every application's graph form a star. When joining the graphs, all the stars have the same central vertex. While there exist applications where end-nodes can communicate without going through the processing center, either directly or through "normal" internet routing, these are not included in the provided simplification. The general idea of using an IoT platform would be to utilize a processing center for gathering data and controlling the application.

4.2.1 General observations

In a star, the only path from one outer vertex to another is through the central vertex. Therefore, it is the point where all paths *must* cross, and constitutes an obvious single point of failure. If all vertices are operational but not the central, they are all isolated. Interestingly, the central vertex is also the point where all paths *can*

[1] Again, in reality, there probably exist redundant solutions, such as transmitting electricity from a neighboring power plant into the electricity grid, but here this is neglected to illustrate how a star topology can affect a network.

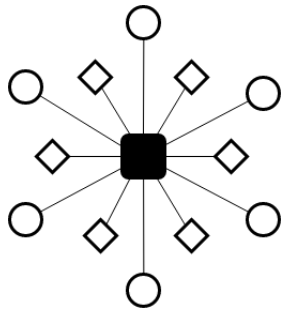

Figure 4.2: The architecture of the simplified IoT platform with two applications. A star shape with a central node is formed as the same IoT platform provider is used by the applications.

cross. From being a graph, the central vertex does not distinguish between circle or diamond shaped vertices and it can thus act as a point of *spreading* as well as a point of failure. This calls for attention when trying to avoid attackers from accessing arbitrary devices within one or multiple applications. Should an attacker be able to control or access the central vertex in a graph, attacks against all outer vertices could be launched from the central.

4.2.2 Attack resistance

Although the graph has a single point of failure, it can be considered highly resistant to random attacks or failures. If attacking a random vertex of the graph, the probability of striking an outer vertex is significantly higher than striking the central. Given that the graph contains n vertices, the probability that more than the attacked vertex will be affected by the random attack is $p = 1/n$, and nearly neglectable as n grows very large. The existence of dependencies between the outer vertices of the graph has not been considered, as these would not constitute a significant amount.

On the contrary, as the graph has a single point of failure, it is very susceptible to a targeted attack. A successful attack on the central vertex would tear the graph apart, and as this graph, in fact, is a product of multiple graphs, this implies huge damage to numerous applications. As the direct edges between the outer vertices and the central vertex would not exist in the IoT platform, attacks on these have been left out intentionally.

4.2.3 Additional appearances of stars

As already mentioned briefly, the star topology is formed by routers, switches, and base stations, for example. Because these are common components of the Internet,

stars occur in multiple locations throughout the IoT platform. This leads to a graph where there are a significant amount of vertices that, if failing, could isolate other vertices. While the central vertices of these sub-stars should be considered as critical elements of the graph, they do not come close to the importance of the processing center of the IoT platform. Attacking a central vertex of a sub-star could limit operations in a particular area, building, or home, but would leave the other parts of the graph unharmed. In Figure 4.3, a highly simplified illustration of occurrences of sub-stars is given. The dark vertex indicates a failure that affects the blurred vertices.

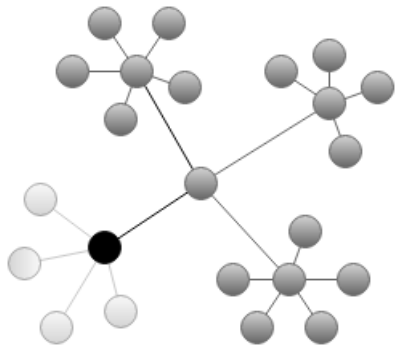

Figure 4.3: A star graph consisting of multiple sub-stars.

4.2.4 Key idea

Ultimately, the key idea of this section is to realize the importance of the central vertex of a star. From a graph theory perspective, the central vertex would be the natural mark of a targeted attack due to its role of connecting the graph. In the IoT platform's context, the central vertex would be representing the processing center, and the need for protecting this point coheres with its role in the platform. By taking out the processing center, the entire portfolio of applications would be unavailable, all at once. Additionally, the processing center is the node which creates a physical link between every end-node, across every application running on the IoT platform. This also makes the processing center a desirable node to control or manipulate for an attacker. Conclusively, all of these observations should be included when considering security measures for the IoT platform.

4.3 The Internet as a graph

Primarily, the processing center is accessed over the Internet, and the Internet is the main network for transmitting data to and from the end-nodes in the applications. As this part was highly simplified in the previous section, this section tries to identify,

using graph theory, whether or not there are properties of the Internet that could affect the overall robustness and vulnerability of the IoT platform.

Traditionally, the architecture of complex networks, such as the Internet, has been considered completely random. Because of the simplicity and elegance of Erdös-Rényi graphs [BB03], they have been used to describe these networks for decades. However, after the discovery of *scale-free* networks, these became the model for describing the properties of complex networks, such as a brain's neural network, a social network, and transport logistics [Aud11, BB03, Con16l]. Therefore, when making a graph of the Internet, it is natural to give it scale-free properties. An important observation to notice regarding the Internet is that it is a constantly evolving network, where nodes and links are added and removed gradually. This allows for the *Matthew effect* to take place, which means that the popular nodes of the network become even more popular as the evolvement takes place.

4.3.1 The importance of hubs

Because the Matthew effect applies to scale-free graphs, hubs tend to occur. The hubs are vertices with a relatively high number of connections or links, and their roles are to interconnect vertices across distant or diversified parts of the graph. Ultimately, they are the reason why the small world property can be found in scale-free graphs. The small world property allows for the existence of paths that cross large graphs with a relatively low number of jumps, and ensures that the peripheral vertices remain within a limited[2] distance to the other vertices, even as the graph grows large. Hubs are also important in the way that they allow for multiple paths from one vertex to another, introducing redundancy in the graph. A scale-free graph is illustrated in Figure 4.4[3]. The hubs colored in light gray.

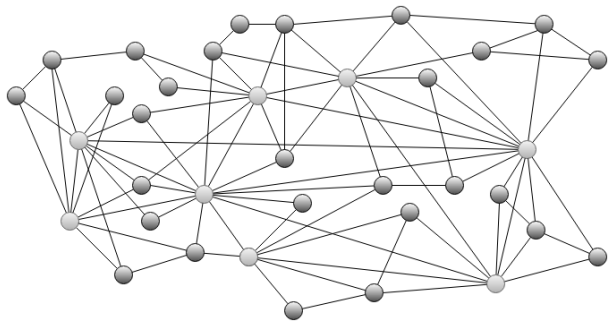

Figure 4.4: A scale-free graph with 36 vertices, where eight of them are hubs.

[2] According to Cohen and Halvin [CH03], the diameter of scale-free graphs are $\sim \ln \ln n$.
[3] The Figures 4.4, 4.5, and 4.6 are all based on, and similar to, the figures in Chapter 8 of [Aud11].

4.3.2 Attack resistance

As there are only a few hubs relative to the total number of vertices in a scale-free graph, statistically there is a lower chance of having random attacks or failures affecting these. Even if a random attack is able to strike a hub, there is a good chance that the remaining hubs introduce enough redundancy to keep the graph connected, or at least without isolating too many vertices. Recall that the main concern of an attack is to see whether or not it can dissolve the graph partially or completely, as mentioned in Chapter 2. Figure 4.5 illustrates a random attack to a scale-free graph. Although two hubs were hit, the graph is still connected.

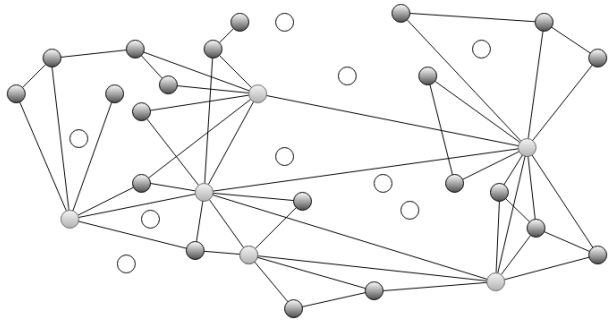

Figure 4.5: A random attack on the graph, where nine vertices have been removed.

While scale-free graphs are resistant to random attacks, they are very vulnerable to targeted attacks. If an attack can take out the hubs of the graph, the entire structure collapses, and multiple vertices break into isolated components. By comparing the figures 4.5 and 4.6, one can easily see the difference between the connectedness of the graphs although the same amount of vertices have been removed.

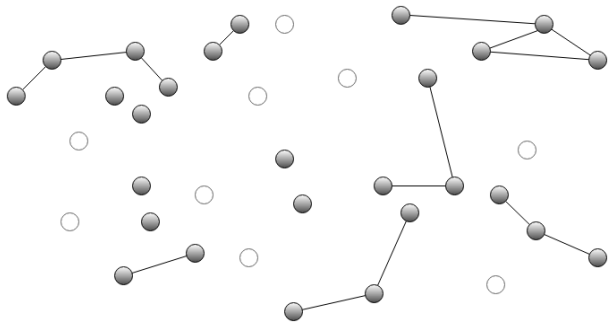

Figure 4.6: A targeted attack on the graph, where nine vertices have been removed. Eight of the nine removed vertices were hubs.

4.3.3 Key idea

When the data is transmitted to and from the processing center, the Internet must be relied on to function properly. In this section, the robustness and vulnerability of the Internet have been described briefly and without any special regards to the IoT platform's physical location or architecture. What the analysis shows, is that the graph of the Internet is vulnerable to targeted attacks, just like the stars. However, because the Internet is a huge network, scaled for an enormous amount of traffic, an attempt to affect the IoT platform's performance by performing attacks to arbitrary nodes on the Internet would be extraordinarily difficult. From an attacker's point of view, spending resources on weakening the structure of the Internet seems ineffective and inexpedient, especially as star structures exist in the IoT platform and the centers of these have been identified as critical nodes. Furthermore, the processing center and its nearby network elements would be much more logical points to attack if having an objective of causing damage, unavailability, or disturbance to the IoT platform. Consequently, the structure of the Internet is not further discussed as other structures have been identified as more critical with respect to the availability of an IoT platform.

4.4 Redundant processing centers

Until now, the processing center of the IoT platform has been assumed to have one physical location and consist of servers that operate coupled. However, a natural approach to strengthening the platform's robustness and reduce its vulnerability would be to divide the processing center and all its functions into multiple centers that can operate independently while at the same time cooperate. While introducing redundancy could strengthen the availability of the processing center, it could also introduce new possible attack vectors. An attacker could, for example, attempt to interfere with the mechanisms used for cooperating properly. In Chapter 6, the attack surface of the processing center is analyzed, but because the concept of redundant processing centers is briefly presented in this chapter, it does not contain any further elaboration on the alternative attacks the concept could allow.

4.4.1 A distributed solution

One way of introducing redundancy is to have multiple separate centers that handle application traffic independently but maintain a universal state of the applications. The centers would be scattered over various strategic physical locations and interconnected by private high-speed networking that ensures constant connectivity between the centers. Through specialized algorithms for distributed systems, the centers keep each other informed about events that affect the state of the applications. Optimally, this happens at a pace that leaves the scattering transparent to the end-nodes and

users of the IoT platform. In the occurrence of one center failing, the others would be able to function normally and without extensive delays or loss of data due to the distribution of information. If an end node's primary center is unavailable, it would simply access another one without any need for manual reconfiguration. It is notable that implementing complex distributed systems is known to be very challenging [CDKB11], but is assumed to be conceivable without further specification in this context. "Normal" traffic, such as application data from the end-nodes, would be carried over the Internet, just as with the single processing center.

A possible graph of the distributed processing center can be seen in Figure 4.7. In the graph, the solid vertices represent processing centers, and the hollow vertices represent public internet elements. The latter is included to show that the interconnected processing centers, hereby referred to as the *core*, are all connected to the Internet as well as having private network connections, represented by thick lines in the graph. As the Internet has already been established as an unlikely target for attacks, see Section 4.3, the core of the graph is the part that is considered here. An underlying condition for the graph is therefore defined; in the event of all solid vertices failing, the remainder of the graph becomes superfluous and the entire graph can be regarded as dissolved. Thus, a failure of the core is considered equivalent to a failure of the central vertex in a star.

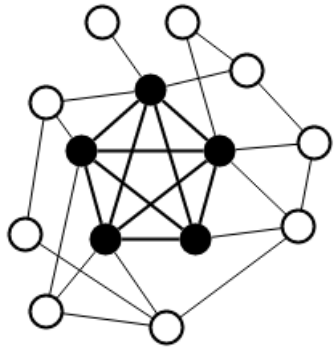

Figure 4.7: A distributed processing center, interconnected by private networking

As all pairs of vertices in the core are adjacent, the core is, in fact, a clique. The clique has a diameter of one, $d = 1$, and any additional edge would only cause a redundant direct path from one vertex to another. In the case of an attack or failure of a vertex, the clique is reduced by one member, but the failure does not affect the connectivity of the remaining vertices within the clique. If an attacker targets an edge of the clique, this would only impact the distance between the two vertices that were connected by the edge. The distance becomes two while the remaining pairs of vertices still have a distance of one, and the attack would be quite uncritical. As

a result, if attackers attempt to take down the entire processing center, they now have to successfully attack the entire clique. From this, it follows that the larger the clique, the more resistant it is to random or targeted attacks, as every new vertex would increase the number of successful attacks needed to dissolve the graph. Given that the vertices of the core have a connection to the Internet, both the processing center and the IoT platform are assumed to function properly even if the clique is reduced to a single vertex. Seemingly, a distributed processing center would be a considerable improvement in terms of robustness and vulnerability from the IoT platform's previously described single point of failure.

In addition to significantly increasing the robustness and reducing the vulnerability of the IoT platform, a clique also brings along a high *linking expense*. As all pairs of vertices in a clique are adjacent, the total number of edges needed to connect them, E_{tot}, follows a quadratic growth; $E_{tot} = n(n-1)/2$, where n is the number of vertices in the clique.
While it is not expected that the processing center will consist of a great number of nodes, the cost of acquiring and maintaining private network connections is still known to be substantial. Therefore, implementing a structure that is as well connected as a clique must be a matter of consideration, especially when also taking into account the complexity involved with creating a distributed solution, such as ensuring consistency and sufficient performance.

4.4.2 Backups of the processing center

An alternative approach to achieve redundancy of the processing center is to have backup nodes in the network. The backup nodes would continuously mirror the main processing center and be able to take over its role at any given time. Naturally, a failure in the main node would lead to some delay and potentially some loss of data, but the applications would be able to function properly after a short period of time. In the event of a failure, the backup node must announce to all end-nodes that it has become the new processing center and the applications must resynchronize with the end-nodes in the cases where it is necessary. Moreover, the takeover by a backup node is from here on assumed to successfully follow some well-defined procedure in every situation this is required.

Also in this solution, the main- and backup processing centers would be spread across various strategic physical locations, and communication between them would ensure backup of the data. As the principles of this solution are to anticipate a failure and thereafter recover, a resynchronization period already lies within the nature of the solution. This allows for some tolerance on the network performance, and thereby that the communication between the processing centers can be carried over the Internet. Figure 4.8 illustrates a graph where the blue vertices, the possible

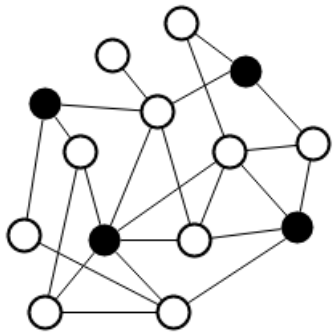

Figure 4.8: A graph where the processing center has multiple backups.

processing centers, are connected through edges and vertices that represent public elements of the Internet.

Because the vertices of the processing center ”blend“ into the Internet, it is natural that the graph follows the same properties as those uncovered in Section 4.3, where scale-free graphs were studied. From there, the importance of hubs and the vulnerability to targeted attacks were established, and these properties also apply to the graph in Figure 4.8. However, as with the distributed solution, the solid vertices of the graph serve a special purpose and the graph can be regarded as dissolved if failures or attacks leave the graph without any solid vertices.

The attack resistance of backup solution's graph becomes very much alike distributed solution's. As the solid vertices would be the natural targets, increasing the number of these would increase the robustness and decrease the vulnerability of the graph. By removing the element of private networking, the linking expense becomes lower, but this could affect the performance of the processing center. Otherwise, the two solutions are not that different when comparing properties of the graph.

4.4.3 Mixed configuration

Finally, it is realistic to assume that an IoT platform provider would utilize both of the two proposed mechanisms for having redundancy. As briefly indicated, the costs are a factor which needs to be considered by any company, and it is likely that IoT platform providers would put a lot of effort into balancing quality, performance, stability, etc., with cost. Consequently, a third solution which might be employed to have redundancy of the processing center is a mix of the solutions presented in the above sections. As the graph properties of a mixed configuration would variate from one solution to another, depending greatly on the implementation, these are not discussed any further.

4.5 Summary

In this chapter, three main elements of the IoT platform have been analyzed with the use of graph theory. While no actual graph of the entire IoT platform has been given, certain parts of its structure have been illustrated and discussed as graphs. By overlapping graphs of multiple IoT applications, the processing center has been identified to constitute a potential single point of failure for all the applications running on the IoT platform. While the Internet has been shown to be susceptible to targeted attacks, it still seems as if it would be more efficient to attack structures directly related to the IoT platform. Finally, some general solutions show how to improve the robustness and vulnerability of the processing center by adding redundant nodes in the network.

Chapter 5

Assessing the security of the IoT platform's end-nodes

The purpose of this chapter is to identify the attack surface of the end-nodes belonging to an IoT platform. Also, the chapter evaluates security mechanisms that could be used to prevent successful attacks from occurring. Possible attack scenarios are also presented to better understand how the attack surface relates to the security mechanisms. The assessment has a generic approach in the sense that no particular surface or mechanism is studied in depth or is heavily weighted in the analysis. The platform can handle multiple types of applications, which may have various requirements for security.

5.1 Introductory notes

Before the more technical material of this chapter is given, this section presents some brief considerations of the ”softer“ aspects regarding IoT end-nodes.

5.1.1 Manufacturers of end-nodes

For the IoT to grow, the manufacturers of IoT end-nodes also have to grow. An expected increase of billions of end-nodes within few years [Ins15b] require rapid development and manufacturing by companies that produce microprocessors, embedded systems, and other types of hardware components used in the end-nodes. For each new IoT solution or IoT service, there is a need for someone to produce devices with given functional and non-functional requirements such that the product can be realized.

Although the manufacturers are specialists in producing hardware, it does not automatically follow that they also are specialists in cybersecurity. In fact, as there is a constant ongoing race for creating end-nodes that are, for example, smaller, faster, and stronger than the competitor's end-nodes, it might be demanding to also achieve

an acceptable level of security for the devices. Particularly, if manufacturers have to prioritize how resources are spent, security could risk being neglected[1].

5.1.2 Motives for attacking end-nodes

An attacker can have any type of personal motives for attacking an IoT application, such as economic crime, sabotage, terror, and espionage. However, an attack to the end-nodes can be narrowed to some more specific motives, independent of the underlying agenda of the attack. Deliberately, these motives are described in short and on a non-technical level in the list below:

1. An attacker could wish to disable, disturb or invalidate an end-node in such an extent that it becomes useless to the application it serves.
2. By secretly listening on the traffic of the end-node, the attacker could learn sensitive information.
3. If gaining the ability to modify data traffic to and from the end-node, the attacker has control over the end-node and can use this to falsify data and actuate the actions of the end-node.
4. As it is already operating within a system, the end-node can be used as a gateway to launch attacks to other parts of the IoT application.

Because the end-nodes in many cases are spread over multiple physical locations and often communicate using wireless technology, they are very vulnerable to attacks. Attackers have the possibility of being in physical proximity of end-nodes, either by accessing it physically or accessing its wireless signals. Thereby, one could argue that attackers by default have a foothold within the IoT application. Attackers might also target end-nodes from across the Internet by trying to exploit vulnerabilities in various network services that are used. Additionally, the end-nodes can be attacked via the IoT platform to which it is connected. The attacks could then come from other end-nodes that are malicious or compromised, or from the processing center if it also has been compromised or contains vulnerabilities. The following section elaborates more regarding the attack surface.

5.2 The attack surface

To better understand what types of security mechanisms that are required or should be prioritized to secure the end-nodes of the IoT platform, an overview of the

[1]To avoid going into the area of business strategy this topic is not further discussed. However, the thoughts are based on the idea of balancing a triangle of security, convenience, and cost [Hen14].

attack surface is presented in this section. The mapping of the attack surface is partially adapted from the OWASP IoT project [Org16], Daniel Miessler's talk at DEFCON [Mie15], as well as Jan Audestad's compendium on network security [Aud12]. The focus of the mapping is towards the context of evaluating applications running on the IoT platform. Nevertheless, the identified attack surface should apply to a broad specter of IoT solutions.

5.2.1 Physical access to the end-nodes

If an end-node is located somewhere with weak or non-existing access control, an attacker is given multiple possibilities for compromising the end-node or the IoT application in general,

Physical user- or administrative interfaces The end-nodes could have various types of available ports, such as USB, Ethernet, and Thunderbolt, which could be configured to give the connected device defined privileges automatically. In practice, an attacker could have to run some software or service to communicate with the end-node.

Force insecure state Although an end-node might be configured securely, an attacker could be able to reset these configurations or downgrade the security level. This could be done, for example, by pressing a physical button on the end-node. Potentially, it would be left without any protection and can be accessed by anyone through its channel of communication.

Data and firmware extraction By having physical access to an end-node, an attacker could extract various types of data from it. First, the device memory could be dumped to find clear text passwords, encryption keys, and other sensitive information. Second, by extracting the firmware, data such as hardcoded passwords, sensitive URLs, and various keys might be located in the code. Finally, the local storage of the device could be investigated. Also here, confidential information could be found. If the data is encrypted, how well is it protected?

In Figure 5.1, a brief illustration of the described attack surface is given. An additional attack vector the IoT end-nodes could suffer from, given that the attacker has physical access to the nodes, is plain destruction or elimination of the IoT device's components. While this is an actual concern and should be considered by the application owners, it is beyond a certain distance of what one could call a cyber attack, and therefore not further discussed.

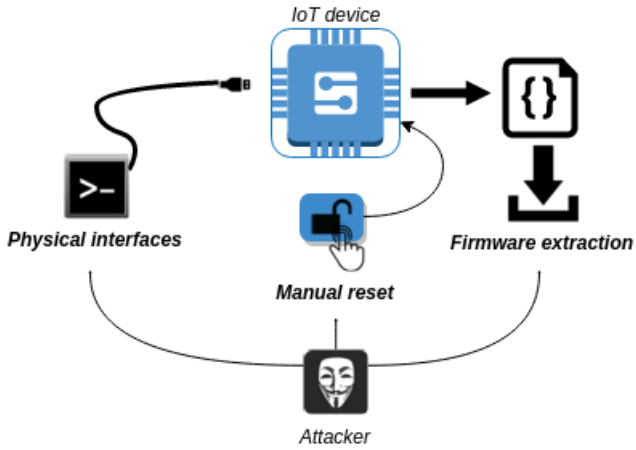

Figure 5.1: An overview of some possible attacks within the physical attack surface of an end-node.

5.2.2 Device web interface

To give the users of the end-nodes the possibility of adjusting parameters such as define the application context, and perform various configurations, many devices have a web interface. An example of a common device interface is the web interface of a router, where a user can change default settings and reconfigure the security parameters. While these web interfaces often are useful and necessary, they pose a security threat if they are not kept up-to-date. Because the device web interfaces often are embedded by the manufacturer, they tend to have outdated security features and could pose a serious threat to the application. Although there are multiple possible attack vectors to a web interface, only a few common vulnerabilities are described in the following paragraphs to avoid an extensive study of this particular topic. The examples give an idea of how an attacker could attempt to compromise the end-nodes using flaws in the web interface. Figure 5.2 contains a rough illustration of some possible attack vectors for an attacker.

SQL injection The possibility of performing SQL injections have been a frequent vulnerability in web interfaces where an SQL-database is utilized for storing usernames and passwords. Now these vulnerabilities are more common in old and unpatched software than newly developed login mechanisms. By filtering user input incorrectly, login data fields can be used by an attacker to inject code and create malicious SQL statements. When evaluated, the statements can grant an unauthorized user access because, in reality, the authentication is bypassed by the malicious statements.

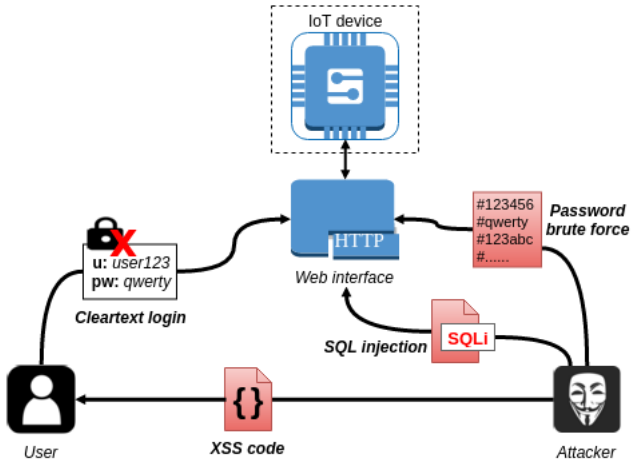

Figure 5.2: An overview of some possible attack vectors which can be used against an IoT end-node's web interface.

Cross-site scripting (XSS) A common vulnerability on many web interfaces is the possibility for XSS. The fundamental principle of XSS is that the attacker can make the victim's browser execute some malicious code with the current privileges of the victim. Consequently, this can be exploited such that the victim, or the victim's browser, performs actions defined in the code. The actions could be to, for example, reveal current session cookies to the attacker or to execute commands or requests that possibly compromise the victim or the system the victim is using.

While there are multiple types of XSS attacks, a *reflective* attack would be the most relevant for device web interfaces. When a reflected attack is performed, the attacker lures the victim into visiting the vulnerable web interface, where the victim unintendedly executes the malicious code. Often, the victim is persuaded into visiting the web interface after receiving a URL from the attacker. If not observant, the victim fails to discover that the malicious code is obfuscated within that very URL. The code can be used to reveal sensitive information to the attacker automatically, or force some action in the web interface without the control of the victim. For example, if an attacker knows that the victim is in control of a particular end-node and is familiar with the web interface of that end-node, an XSS-attack could be used by the adversary to make the victim shut down functions of the end-node without the victim realizing it.

Username enumeration As a password brute force attack is a commonly used vector by attackers, username enumeration is performed to reduce the size of the attack. If the web interface provides unequal responses to login attempts with correct

and incorrect usernames, an attacker can reduce the brute force attack to usernames that are known to exist. If the end node's web interface is accessible to a number of people, and the usernames are generated by some formula (e.g. U0001, U0002, ...), multiple usernames can easily be deduced if an attacker can find only one valid.

Weak credentials Many web interfaces support the use of weak credentials, such as short passwords with low entropy or default credentials that are publicly known. The former includes allowing passwords that do not contain a mix of upper and lower case letters, numbers, and special characters. The user would then be able to choose the weak passwords "123456" and "password", for example. The latter, default credentials, involves that the web interface uses credentials that are predefined by the manufacturer and common to all devices running the same interface. An attacker can do a simple search on the Internet to find if the device interface is delivered with default credentials, and what those credentials might be.

No use of HTTPS When communicating with a web interface, a user should be able to trust that the counterpart is a legit site and be able to share its username and password securely when authenticating. However, without the use of HTTPS an attacker can perform a *man-in-the-middle attack* to view sensitive information and credentials being transmitted in cleartext. HTTPS is, by far, the most used standard for end-to-end encryption in web interfaces.

When a man-in-the-middle attack is performed, an attacker intercepts traffic to and from one or more entities, and reads, modifies or deletes the exchanged data. For example, an attacker could create a false network access point and have the victim connect with it because of excellent signal strength. Further, the attacker redirects all traffic to an actual access point and acts as "a man in the middle", as illustrated in Figure 5.3. If the web interface does not employ HTTPS or other end-to-end encryption, the attacker is able to monitor all traffic passing through the false access point, and inspect the traffic for sensitive information. Additionally, the attacker could also modify particular messages, such that they contain malicious data.

5.2.3 Network services

In addition to having web interfaces, the end-nodes could also have other network services running, where some might be vulnerable. While network services could seem less accessible or interpretable than traditional web interfaces, these services are just as important as a web interface and constitute a significant part of the attack surface. As mentioned in Chapter 3, a skilled attacker would look for vulnerabilities in all areas of an application and pick those who are easiest to exploit.

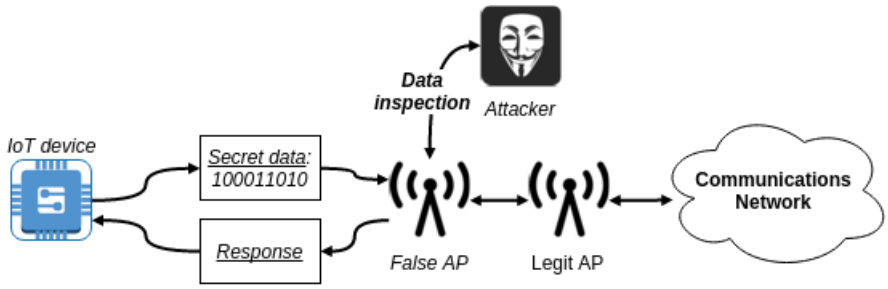

Figure 5.3: A false access point is used to intercept all legitimate traffic to and from the IoT end-node. The traffic may or may not be end-to-end encrypted.

Unintended open ports For the end-node to be able to receive instructions or commands from the processing center, there must exist some open port to allow data flowing to the device software. While this alone does not pose an immediate threat, open ports must be configured carefully to avoid giving attackers an increased operating surface. For example, a network service could be protected by a login mechanism, but wrongfully allow anonymous logins. Any unintended open port could potentially compromise the end-node as it is unlikely that it has received the same proper configurations as an intended open port.

Available command line interfaces (CLI) On the network layer, there could be left CLIs or shells that are open and listening for incoming connections. If finding one, an attacker could gain both user or administrative privileges depending on the type of shell, and further perform actions to compromise the application.

Lack of end-to-end encryption When end-nodes transmit or receive sensitive or confidential information, end-to-end encryption of the relevant network services should be present. An attacker could far easier gain unauthorized knowledge by listening to services where TELNET, SMTP, and HTTP are used than their encrypted equivalents SSH, SMTPS, and HTTPS. Here the same security principles apply as in the previous section about HTTPS; if an attacker in some way is able to intercept messages in transit, sensitive information could be compromised.

Information disclosure Although being properly protected and encrypted, it is still possible that network services leak sensitive information about the end-node or the application itself. When an attacker uses the banner grabbing technique, connections are made to the end node's network services by using plain networking tools. After connecting, the attacker looks for system information which is unintendedly disclosed about the end-node or other parts of the application. Also by passively listening on

network traffic, an attacker could be able to gain knowledge of, for example URLs and APIs, where some might be regarded as sensitive.

Denial of service A Denial of Service (DoS) attack involves overloading the target with dishonest traffic such that it is unable to handle legitimate traffic. There are multiple techniques for doing this, some more advanced than others, and if conducting a successful attack, the attacker causes unavailability of the end-node, such that it is unable to operate properly. As an end-node have limited resources, preventing sophisticated DoS attacks could be very challenging, or even close to impossible. Particular security mechanisms for preventing DoS attacks of end-nodes are therefore not discussed further.

5.2.4 Miscellaneous

As not all attack surface areas can be included in a particular category or subgroup, this section lists some areas that are equally important to consider but are difficult to align with a group of others. Because an IoT application consists of multiple components and has a wider threat landscape than for example a standard web application, more ”untraditional“ attack vectors may also be applied to IoT applications and their end-nodes.

Use of insecure communication technologies End-nodes have no defined restriction on how they achieve connectivity or communicate. Multiple types of technologies can be utilized in mobile networks, IP-networks, local area networks, or personal area networks, for example. While the most common protocols and technologies utilize mechanisms for adequately securing communication, there are older protocols that should be avoided to prevent attackers from performing well-known attacks. For example, usage of WEP in WiFi networks can easily be broken by multiple attacks [CHWW03]. And usage of GSM in mobile networks fails to authenticate the serving network [KSBB10], making man-in-the-middle attacks highly achievable.

Even though common and up-to-date protocols are used, the presence of particular vulnerabilities and possible attacks is often nonetheless true. An issue related to mobile networks is the matter of downgrading security. UMTS is a technology that employs multiple security measures and preserves user confidentiality through these. However, the ”UMTS-GSM interworking“ is a backward compatibility feature which can be exploited by attackers [ASS09] and used to compromise UMTS mobile stations. To allow interworking between GSM mobile stations and UMTS networks, and vice versa, migration between the two technologies were introduced. In 2004, Meyer and Wetzel showed how a false GSM station can be used to compromise a UMTS subscriber by exploiting shortcomings in the GSM authentication and key agreement [MW04]. After the UMTS subscriber connects to the false base station,

the attacker decides to use ”no encryption“, or weak or broken encryption algorithms. This enables an attacker to intercept and read all traffic to and from the mobile station.

Also in other common technologies, vulnerabilities and possible attacks have been proved to exist. In personal area networks, the key exchange in the state-of-the-art technology Bluetooth Low Energy (or Bluetooth Smart) has been showed to be broken. Due to reuse of the long term key (LTK), all communication is effectively compromised [Rya13]. And in Transport Layer Security (TLS), multiple attacks on the most commonly used ciphers and modes of operation have been uncovered. The BEAST attack exploits issues with predictable initialization vectors in the implementation of Cipher Block Chaining (CBC) for TLS 1.0. An attack may efficiently decrypt and obtain authentication tokens embedded in HTTPS requests [DR11].

Summarized, the critical vulnerabilities are those who are related to usage of old and outdated technologies. These often have known practical attacks which easily can be performed by an attacker. Vulnerabilities in up-to-date protocols might be harder to exploit or occur less frequent, but should be treated as actual threats to the end-nodes.

Administrative web interface While Section 5.2.2 describes a device web interface which runs on the device itself, there could also exist administrative interfaces that run separated from the devices. These may contain the same possibilities for controlling the end-node. In an IoT-platform, the processing center's administrative interface could be such an interface, but there might also exist similar functions. For example, an interface created by the manufacturer. If such an interface is uncovered, it could have vulnerabilities similar to those described in the referenced section.

Update mechanism As it often is necessary to update the software or firmware of the end-nodes, an update mechanism is required. Performing these updates remotely should, or even must, be possible as it would be highly impractical to perform manual updates on hundreds, thousands, or millions of end-nodes. Consequently, the end-nodes receive their updates over their network interfaces and install the received package to gain new or improved functions and configurations. However, an attacker could attempt to tamper with this mechanism for multiple purposes.

Given that the update corrects an already existing vulnerability, the attacker could simply try to stop the update from taking place, such that the vulnerability remains. Thus, if the attacker is located somewhere in the middle of the data stream between the end-node and the issuer of updates, the attacker could simply prevent the update from reaching the end node. Moreover, an attacker could try to modify

the updates such that they introduce vulnerabilities. These vulnerabilities might then be exploited after they are installed. Also, the attacker could attempt to learn about existing vulnerabilities by inspecting the update and identify what it attempts to fix or correct. Either way, the update mechanism is an area an attacker definitively could try to assess and exploit.

Ecosystem implicit trust There are multiple degrees of how implicit trust can affect the security of an IoT end-node. The end-node might be configured to automatically trust devices that claim to belong to a particular ecosystem and to be of a particular type, and exchange ”trusted messages“ with these without verifying their authenticity or integrity. This can be exploited by an attacker that have learned how to mimic the ecosystem behavior, which then is able to create false messages that will be accepted.

In the IoT, many end-nodes transmit messages periodically and with identical, or close to identical content. An IoT connected thermometer is an example of such a device. Assuming that the thermometer ”reports“ its status to a processing center every 60 seconds, then there could be very few changes in the content of the roughly 500 messages that are sent during the eight middle hours of the day. If adding protection on all these messages, and considering that the same amount of messages also is generated by thousands of other end-nodes, the accumulated additional cost of having protection becomes substantial. Therefore, ecosystems might not employ protection mechanisms at all, or just implement it for particular services. Either way, unprotected messages can be exploited by an attacker.

Another potential vulnerability within the area of implicit trust is if an end-node is configured to trust that an authenticated ecosystem device only acts within its predefined boundaries. A valid device, which is under the control of an attacker, would still be able to authenticate to a targeted end-node within the same ecosystem. It is then important that the allowed actions of the malicious device are limited to those that are predefined by the authentication level. If the attacker attempts to trigger actions beyond its actual privileges, this should be denied. However, this might not be if the end-node lacks mechanisms for distinguishing privilege-levels.

Use of vendor and third-party APIs In the event of an end-node utilizing multiple APIs in addition to those of the actual application, it is important to be aware of the data going to and from these APIs. Depending on the API, there could be numerous reasons for why it poses a risk to the end-node and the user of the end node. As the security of the third-party API might not be up to date, an attacker could poison the third-party API and compromise the end-node by making it utilize malicious resources. Also, the APIs might receive sensitive information from the end-node, possibly compromising the user's privacy if they actually are

malicious. Additionally, even if the APIs are legit, they could be less strict in the use of encryption, and if sensitive information is shared with any of them an attacker could disclose this information by listening in on the exchanged data.

5.2.5 Summary of attack surface

While the previous sections present the main attack areas and include possible vulnerabilities for each of them, briefly summarized in Figure 5.4, there are of course multiple other specific attack types that an IoT end-node may be a target for. It is important for developers and manufacturers to keep in mind that attackers, in many IoT applications, are able to physically own an end-node themselves. This means that extensive testing of the end-node can be done in a controlled environment, increasing the likelihood that an attacker will find one or more vulnerabilities if they actually exist. Furthermore, as IoT end-nodes often are small and have limited access to resources, implemented security mechanisms might be inadequate, insufficient, or simply non-existent. As attackers know this, the IoT end-nodes could be attractive initial targets for attempting to compromise an entire IoT application, which otherwise is properly secured from the outside.

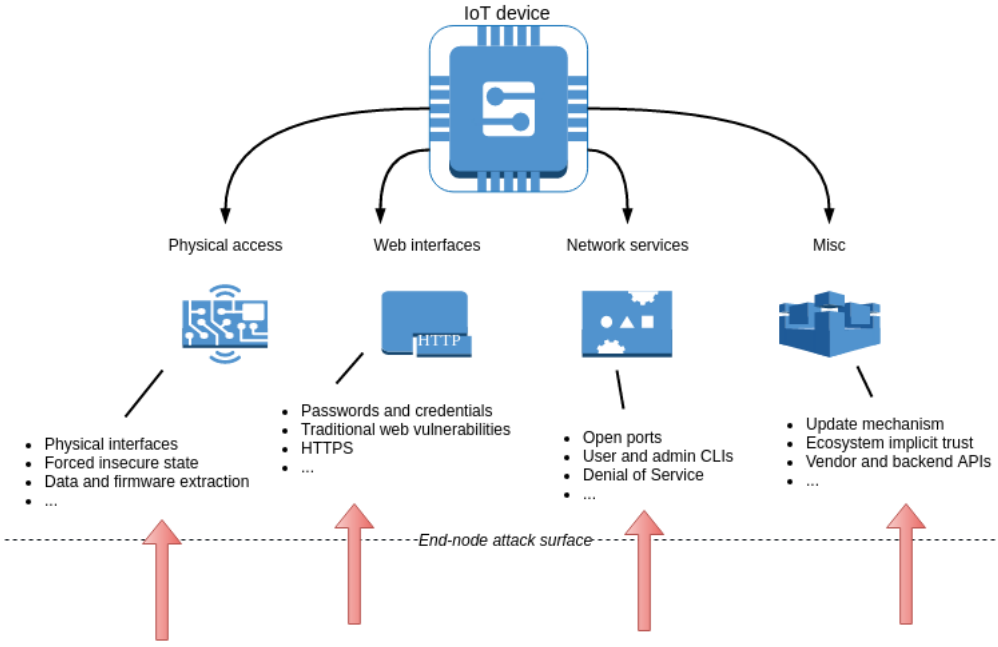

Figure 5.4: An overview of the proposed attack surface of IoT end-nodes.

5.3 Attack scenarios

In this sections, some plausible attack scenarios are provided based on the identified attack surface. While these are not walkthroughs, proof of concepts, nor contain actual details on how to attack the IoT platform and its application, they intend to illustrate how the end-nodes of an IoT application can be used to compromise the system or inflict damage to people or material.

5.3.1 Attacking a smart meter with physical access

Section 3.3.1 describes an IoT application where smart meters are used for metering the electricity consumption of households. In this scenario, such a smart meter is a target for an attack were some exemplified vulnerabilities are exploited.

Gaining physical access In the event of a failure or unexpected error within the smart meter, there might be a need for having a technician manually re-configuring it. The re-configuration could, for example, be done through a physical port that gives administrative access to the device. Therefore, if an attacker gains physical access to such a smart meter, attempts to compromise the IoT application can be made. Furthermore, as apartment buildings often have meters installed in bulks or in common rooms, the need for necessary access control could sometimes be difficult to enforce or simply overlooked. If placing smart meters with open administrative ports at such locations, attackers could access end-nodes quite easily and discretely, and potentially compromise the application.

Exploiting an unprotected physical port If an attacker connects to the smart meter through a USB port and obtains an interface with administrative rights, the attacker could be able to perform multiple types of attacks, such as cheat with the readings sent to the electricity provider, turn on and off the access to electricity, or monitor the communication to and from the end-node in detail. By doing the latter for some period of time, the attacker is able to learn when electricity readings are sent to the processing center and what they contain. With this information, falsified readings can be replaced with the legit readings and cause, for example, lower electricity bills to a resident.

5.3.2 Attacking an infusion pump over a web interface

Section 3.3.2 describes an IoT application where connected infusion pumps are used for delivering fluids, such as medicine, into a patient's body. In this scenario, such an infusion pump is a target for an attack were some exemplified vulnerabilities are exploited.

Attacking the weakest link While the infusion pump can be configured through a physical panel, it also allows adjustments of the rate and amount through a network service. This can be done, for example, by following a protocol on a predefined port. In this scenario, the network service is assumed to implement the necessary security mechanisms to ensure that it does not constitute a weakness in the system. However, as many other IoT end-nodes, in this scenario the infusion pump has an available web interface. Because the web interface offers the same functionalities as the network service, an attacker could gain administrative access through exploiting vulnerabilities of this interface instead of the secured network service.

Exploiting use of default credentials From Section 5.2.2, multiple common vulnerabilities of devices' web interfaces were identified. This scenario takes on a "forgotten" web interface, where a possible vulnerability that could be exploited is the use of default credentials. An attacker could be able to access the web-interface of the pump and learn the type of software running on it. The attacker can search the Internet, and other sources, for known default credentials that are used on the identified software. If finding so, the attacker can easily pass the login mechanism and gain unauthorized access to information and functions of the end-node.

Small misconfigurations – huge consequences As the fundamental flaw of the end-node is the outdated and unconfigured web interface, the attacker is given the possibility to stop and start the infusion pump, decrease or increase the rate of infusion, and cause a significantly lower or higher amount of medicine to be infused into a patient's body. Without the necessary security mechanisms, death could be directly inflicted by attacking the end-nodes in this IoT application, and this should be strongly considered by manufacturers of connected health equipment. Unused or unnecessary services must be disabled, and access interfaces must be properly secured due to the significant consequences that are related to health equipment.

5.4 Security mechanisms

As the attack surface and possible attack scenarios have been covered in the previous sections, this section describes possible security mechanisms that may be implemented to prevent successful attacks towards the end-nodes of an IoT application.

5.4.1 Physical access control

While access control includes both physical access control and computer access control, it generally means to enforce a selective restriction to a place or resource. Depending on the context, *access* includes the act of consuming, entering or using [Con16a].

In the applications where it is possible, physical access control should be enforced to avoid increasing the possible attack surface of the IoT end-nodes, as described in Section 5.2.1. By placing end-nodes within some perimeter that is locked from the public and only accessible to a restricted amount of users, the possibility of exploiting physical vulnerabilities or simply destroying the end-nodes are reduced significantly. Also, in the applications where it is unnatural to restrain access to the end-nodes, such as a smart door lock, the end-nodes should have limited physical ports and configuration components. This is to avoid increasing the possible attack vectors to an attacker.

5.4.2 Computer access control

Four types of services should be present to ensure that computer access control is handled properly, not only for an IoT end-node, but also in general. *Identification and Authentication*, *Authorization*, *Access approval* and *Accountability* is a possible categorization of the services that combined can be said to constitute an *access control system*. With the use of a proper access control system, many attack vectors can be stopped, as they often exploit non-existing or fragile access control.

Identification and Authentication As objects, for example files, computers, servers, and end-nodes, should be accessible only to legitimate subjects, there is a need for identifying the subjects and verifying their authenticity. Commonly, username and password are used to enforce identification and authorization, where the username is the identifier and the password is used as a token for authentication or proving legitimacy. However, there are multiple techniques for ensuring that subjects are legitimate. Various computer systems utilize sharing of secret cryptographic keys, being in possession of a smart card, or having certain biometrics properties to authenticate a user. In the case of the IoT, usage of passwords and cryptographic keys would in almost any case be the most appropriate solution for having authentication. It is cheap, can be done autonomous, and it scales well. Both passwords and cryptographic keys can be generated automatically and in large numbers, and be stored securely on tamper-proof devices [Cor16a].

Authorization Authorization is performed to define the access rights for each individual subject, for example a user, an end-node, or another device. Basically, authorization specifies what a subject is allowed to do. For example, upon logging in through a web interface, a person could be granted administrative or user rights.

Access approval Whenever a subject seeks to access an object during operations, there is a need for a service to grant, or refuse, permission to interact with that particular object. As authorization is already assigned to the subjects, the job of access approval is to comply with the authorization policy. For example, a user with

administrative rights should be allowed to change the configurations of a smart meter, while a ”regular“ user should not. If attempting to do so, it should be prevented by the access approval service.

Accountability Finally, to be able to associate actions in the computer system with subjects, logs and records should be kept. By continuously analyzing such logs, system managers could more quickly detect security violations or attempts of so. Also, in the event of an actual security breach, the logs can be used for re-creating the actions which caused the incident. For example, if some non-legitimate user has accessed the network service of a connected infusion pump and performed unauthorized modifications of the equipment, it is vital that the security breach is discovered and understood. The vulnerability such be corrected on that particular device and every other that have the same flaw.

End-node access control As Section 5.2 describes, there are multiple attack areas of an end-node, and many of these can be kept resistant to attacks by implementing proper access control, either physical or digital.

5.4.3 Host intrusion detection systems

Because the threat landscape for computer systems has become tough and unpleasant, for many devices it is no longer a matter of *if*, but *when* they get compromised. Cyber attacks evolve constantly and at a pace which is difficult to keep track of for security engineers. A possible mechanism that could be employed for preventing ”out-of-hand breaches“ is to anticipate that arbitrary parts of a computer system will get compromised. Then, by employing procedures for isolating the fault, one can avoid that the attacker is able to affect other areas of the system. Consequently, an essential prerequisite for making this approach useful is to discover the initial breach at the device that is attacked [Ins15a]. For this purpose, intrusion detection systems(IDS) could be employed.

A Host IDS (HIDS) runs on individual hosts or components in a network, such as an IoT end-node. It monitors inbound and outbound packages while looking for suspicious activity and known attack patterns. In the event of a suspected attack, the HIDS takes a snapshot of the system files on the device and compares it with the previous snapshot. If any critical system files were modified or deleted, the HIDS alerts the system administrator who can perform necessary measures and investigate the event more thoroughly [Con16g].

Although being an approach that looks ahead, there are some concerns regarding actually taking it into use. First of all, the security of the system becomes highly dependent on the HIDS being able to detect an intrusion. If this goes unnoticed, an

attacker can inflict severe damage on the system while acting as a ghost. Furthermore, for an HIDS to work properly, the device it runs on should have a sufficient amount of resources to work with. Unfortunately, this is not the case with the average IoT end-node, which preferably should operate with low battery consumption and limited computational power.

5.4.4 Network address translation

The Network Address Translation (NAT) technique is often found in routers, firewalls, or other entry points to a network. Hosts behind these entry points commonly have addresses in a ”private address range“ instead of individually accessible addresses. By utilizing NAT, the true address of a host is hidden from the public and prevents that devices, such as IoT end-nodes, are directly addressable by attackers using network reconnaissance tools. Initially, NAT was introduced to deal with the limited number of IPv4 routable addresses available but has later proved to be important in a network security perspective as well.

5.4.5 Encryption of data

In the context of the IoT, the main purpose of encrypting data is to allow messages and information to be transmitted over an unsecured channel, for example air or public internet, without being concerned about its content being revealed to unauthorized parties. From cryptography, the process of encrypting is regarded as encoding the data such that only authorized parties are able to read it [Con16e]. Consequently, the sender and recipient must share some secret, or parts of a secret, such that when the message is encrypted, only the intended recipient can decrypt the data into its original form.

By encrypting data properly, an attacker is unable to disclose sensitive information that is transferred to or from the end-node, even if the encrypted data is accessible to the attacker. Although confidentiality of information is the main purpose of employing encryption, it is also worth noticing that controlled modification or injection of packets to the end-nodes becomes significantly harder once a layer of encryption is applied.

In the technologies utilized by the IoT, there are various practices regarding use of encryption. For example, the communication systems GSM and UMTS both employ encryption only on the radio access link and transmit data in cleartext within the core network. However, if there is a need for protecting data more thoroughly, the protocols within TLS can be used for ensuring end-to-end encryption. Multiple layers of encryption are then used. The IoT end-nodes should be able to preserve confidentiality at an adequate level if the need for encryption is evaluated, and the necessary technologies are combined.

An issue regarding encryption of data generated by end-nodes is that the messages often are very short, very similar to each other, and are sent periodically. Thereby, the percentage of overhead that is inflicted, regarding computations and additional payload, can grow colossal compared with the importance of encrypting the data. Ultimately, encryption becomes a matter of privacy, confidentiality, and principles; is it acceptable to give anyone the possibility of reading data from the end-nodes of an IoT application, although employing encryption seems unnecessary and costly?

Symmetric encryption When symmetric encryption is applied, the same key is used for encrypting and decrypting the message. For this to be possible, the key must be pre-shared between the communicating parties and both must employ the same encryption/decryption function. To ensure secure encryption, usage of known cryptographic algorithms, such as Triple-DES and AES, is recommended with key lengths of 112 bits for Triple-DES and 128 or 256 bits for AES [oST15].

In the IoT, there are two main areas where symmetric encryption could come of use. First, when two end-nodes are communicating, and they exchange sensitive data, encryption may be employed to avoid disclosing the data to potential attackers listening on the communication. However, if one of the end-nodes is malicious or has been compromised, the data is not protected after the transmission is completed, as decryption is done by the receiving end-node. Second, when an end-node is communicating with the processing center symmetric encryption can be used to achieve end-to-end confidentiality. This means that even if the data is transmitted via other end-nodes or networks that are malicious, the data cannot be decrypted before it reaches the intended destination which holds the encryption/decryption key.

Asymmetric encryption Asymmetric encryption is based on public-key cryptography and allows for encryption and decryption using two different keys. With the help of a *trapdoor one-way function*, asymmetric encryption of a message can be done with a public key, a key known to anyone, but only be decrypted with a corresponding private key, which is known only to the indented recipient of the message [MvOV97]. While the recommended key lengths for the various asymmetric encryption schemes vary a lot depending on their type of trapdoor function, the length for the schemes using elliptic-curve cryptography is 224 bits [oST15].

Because asymmetric requires far more computational power than symmetric encryption, asymmetric encryption is mostly used for establishing a session key, which is to be used for symmetric encryption for a limited amount of time. As resources are limited in the IoT, some applications could find that asymmetric encryption requires too much additional computation and power consumption, such that the end-nodes cannot operate effectively.

On the contrary, with asymmetric encryption a boundary within the IoT can be removed. By using public and private keys, two end-nodes that have no pre-shared keys are able to communicate encrypted and securely. In fact, communication between any two types of IoT entities can be encrypted, regardless of their manufacturer, ecosystem, etc. Although securing the exchange of information from others, there is still an issue of establishing trust between the two parties. This is briefly discussed towards the end of the following section.

5.4.6 Data integrity and message authentication

When an end-node receives a message or instruction from either another end-node, the processing center or through some other network service, the end-node should be able to verify that the message has not been modified or altered in transit and to verify that the message actually came from its reputed source (message authentication or data origin authentication). Also messages *from* the end-nodes should comply to the same properties, such that modified or false messages can be detected by the receiving party. Continuing, the concepts of data integrity and data origin authentication cannot be separated as an effectively altered message have a new source, and if a source cannot be determined, then the question of alteration cannot be settled (without a reference to a source). Message authentication is therefore implicitly provided by integrity mechanisms and vice versa [MvOV97].

While data integrity and message authentication are desirable properties to have in an IoT application, also here overhead should be considered. The overhead could, for example, increased computations and additional payload. In Section 5.4.5, about encryption of data, the matter of similar and periodic messages is discussed. The same principles as those discussed there applies to data integrity and message authentication.

Message authentication code (MAC) A MAC is a short piece of information that is computed by the use of a hash function which takes a message and a secret key as inputs. The MAC can be added to the original message and used to confirm that the message came from the stated sender and confirm the that the message has not been altered in transit. To use MACs properly, the communicating parties should comply to three algorithms [Con16j]:

1. A key derivation algorithm – A common secret key is needed to authenticate the origin of the message.

2. A signing algorithm – The message and the key should be compressed into a short tag by a hash function.

3. A verification algorithm – The receiver should *accept* or *reject* the received message.

To verify the integrity of the data, the recipient of the message and the MAC has to use the same hash function and know the secret key, and use these to produce a new MAC. The new MAC is compared to the one received, and if they differ, the received message should not be trusted.

Although there are multiple possible choices for hash function when creating a MAC, not all of them are desirable in terms of strength of security. Because hash functions reduce the size of messages with arbitrary length into a fixed size tag, collisions must exist. Take for example the hash function *SHA-1*, which has known collisions [WYY05]. If using this function, an attacker has an increased probability of successfully breaking the implemented security mechanism, and it could potentially lead to a successful attack. Therefore, the end-nodes of an IoT application should be equipped with capabilities for signing and verifying messages using strong cryptographic hash functions.

Authenticated encryption (AE) As MACs provide data integrity and message authentication but no confidentiality, the use of AE is a possible approach to include this as well. There are three main techniques for AE [Con16c]:

1. Encrypt-then-MAC – The plaintext is first encrypted, then a MAC is produced from the ciphertext. The ciphertext and the MAC are concatenated and sent to the receiver.
2. Encrypt-and-MAC – The plaintext is encrypted, and the MAC is produced from the plaintext. The ciphertext and the MAC are concatenated and sent to the receiver.
3. MAC-then-Encrypt – The MAC is produced from the plaintext and concatenated with the plaintext. Then they are encrypted together and sent to the receiver.

The three techniques are used by various applications (IPsec, SSH, SSL/TLS), depending on the desired properties and performance. An important observation regarding AE is the usage of the same key for encryption and integrity. This should be avoided as it prevents the receiver from detecting if one of the two mechanisms has been compromised.

Digital signatures A digital signature is a mathematical scheme that can be used to sign digital messages and documents. When holding a valid digital signature on

some message, the receiver is able to determine three things; the origin of the message, verify that the message has not been altered in transit, and be ensured that the sender cannot repudiate having sent the message (non-repudiation). Digital signatures are based on public-key certificates. Basically, these are proofs of ownership of some public key, and give the receiver reason to trust both the integrity of the message and the identity of its origin. For this to be possible, the receiver uses some scheme for trusting that the certificate is valid. The receiver then proceeds by verifying that the message signed with a private key can be decrypted properly with the public key of the certificate.

In the IoT, digital signatures can be used to sign software updates sent to the end-nodes, to sign application messages, and to create trust between entities both in and across applications. However, there are challenges regarding the performance and the key distribution and management of public-key certificates for IoT end-nodes. As IoT end-nodes tend to have limited computational power, memory, and bandwidth, for them to employ advanced cryptography is challenging in practice [Pat15]. Therefore, use of MACs is generally a better option in IoT applications, provided that certificates are not needed.

Chapter 6
Assessing the security of the IoT platform's processing center

The purpose of this chapter is to identify the attack surface of a processing center belonging to an IoT platform, such as described in Chapter 3, and further evaluate possible security mechanisms to prevent successful attacks from taking place. Furthermore, this chapter has a similar structure as the previous chapter (security of end-nodes) and presents some possible attack scenarios to better understand how the attack surface relates to the security mechanisms. The overall assessment focus on both on features that are specific to an IoT platform , such as serving multiple applications simultaneously, and features could be valid for any IoT application.

6.1 Risks related to the processing center

As identified in Chapter 4, the processing center is the key component of the IoT platform for ensuring that all applications are able to operate. If the processing center is "removed", all IoT-functions of all the applications goes down and many end-nodes would not be able to operate at all. Consequently, a processing center poses a tremendous risk, as multiple applications within multiple areas of the society could be attacked simultaneously by someone controlling a computer half the world away. If these applications are used for controlling or operating critical infrastructure, the risk becomes even greater as lives are affected immediately. Also, there is a risk for cascading failures when dealing with critical infrastructure. This means that an attack on an electricity grid could affect other critical infrastructures, such as finance, logistics, and communications.

Another potential risk is if the applications of the processing center, instead of being shut down, are compromised and controlled by attackers. The attackers could then cause errors or inaccuracies that seem arbitrary, but eventually has a significant impact on the customers of the IoT platform, potential end users, or the platform providers.

To the IoT platform provider, the security of the processing center is of special

interest due to the factor of business risk. While the IoT end-nodes might be specified, acquired and managed by the customer, the availability of the processing center is the full responsibility of the platform providers. A processing center that does not correspond to the established service level agreement[1] might inflict loss of reputation or even lawsuits. Therefore, it is in all best interest that the security of the processing center is preserved and that the IoT applications are available to the customers.

6.2 The attack surface

Because an IoT platform's processing center has to be highly accessible to multiple actors within the IoT applications, an attacker has multiple surface areas to work with when trying to find a way to breach the security of the applications. There are multiple outcomes of a successful attack, and while some could endanger the entire IoT platform, other outcomes might only constitute isolated and minor security breaches. Nevertheless, in the following sections, the possibility for breaching security is the main focus regardless of the consequences the breach might cause[2]. In those paragraphs where exploits of vulnerabilities actually are elaborated, it is only to clarify and make the possible threat more understandable, and it does not mean that the remaining paragraphs are less significant in the overall assessment.

It is important to realize that there is a lot of actors in the IoT platform that are supposed to be able to interact with the processing center. Consequently, there is a need for a system that allows multiple types of entities to interact with multiple types of functions, without compromising the security by implementing loose and inaccurate policies. In Figure 6.1, the main actors for communicating with the processing center are divided into four groups which access the processing center through their own interface. While the users and the end-nodes are represented by two types, in reality, there could be many more of them.

The following subsections list various attack surface areas that should be considered to prevent attackers from compromising the IoT platform and to better understand the need for adequate security mechanisms. As with the end-nodes, the identified attack surface for the processing center is based on the OWASP IoT project [Org16], Daniel Messier's talk at DEFCON [Mie15], as well as Jan Audestad's compendium on network security [Aud12]. However, as this is an assessment of the security of a processing center within an IoT platform, multiple additional sources have been added and refinements have been done. This is to give a more precise presentation of the particular attack surface.

[1] A service level agreement (SLA) is a part of a contract where the service is formally defined. In the SLA, particular aspects of the service – scope, quality, responsibilities – are agreed between the service provider and the service user [Con16m]

[2] Such consequences only would be speculative in the sense that no actual system has been tested for vulnerabilities and attacks in this assessment

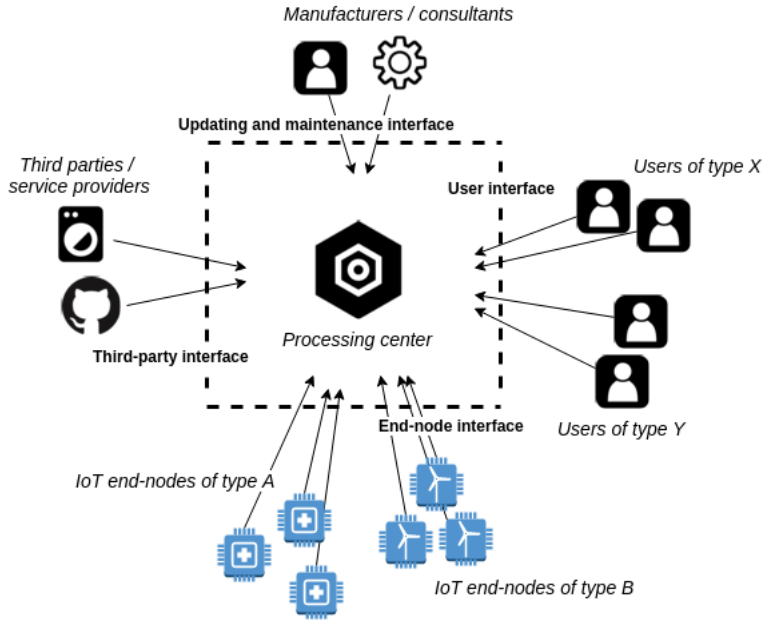

Figure 6.1: An overview of the four main groups that are in need of an interface for communicating with the processing center.

6.2.1 Overview of the interfaces

Through the four interfaces in Figure 6.1, multiple types of attacks can be attempted, as vulnerabilities might be found in many different technologies that are being used to provide access for the actors. While each interface might utilize individual resources that are separated from the other interfaces, the same types of services might often be used by the various actors to access these resources. Therefore, if one of these services is vulnerable, it could be that the same service used by the other actors also is vulnerable. Hence, although the actors are separated in the figure, they could be utilizing services that are based on the same technology and might contain the same vulnerabilities. Furthermore, the actors of the interfaces have different privileges that should be adjusted accordingly to their needs. The privilege to *write*, *read* or *execute* are all a part of defining how actors can interact with the resources of the system. They constitute the actions an attacker would like to be able to perform. Additionally, actors might not have any privileges if they are unable to authenticate, are suspended, or are in a registration process, for example.

End-node interface By infecting an end-node, an attacker can attempt to compromise the processing center in multiple ways. For example; by crafting and sending malicious data, errors could be caused in the system. Requests for resources might

compromise confidentiality. And sending attack vectors through a legitimate device could increase the privileges of the attack. Additionally, an attacker could try to enumerate the network services being used by the end-nodes, and directly access the interface without being in possession of an end-node. Ecosystem communication and network services are relevant to this attack surface area. This is discussed more in Sections 6.2.4 and 6.2.5.

User interface The user interface is regarded as the interface used by all human actors that access the processing center for application-specific reasons. Possibly, a partitioning could include *administrators* of the applications – people who configure parameters and correct errors in the state of the application, *privileged users* – people who can access and use every function of the application, and *basic users* – people who can access and use limited functions of the application.

To attack the processing center, an attacker could attempt exploit vulnerabilities in the user interface directly, for example by impersonating a legit actor, bypass authentication, or elevate privileges. Additionally, the attacker could also try to attack this interface indirectly, for example by compromising the computer, mobile phone or another device owned by a user. Internally within a company, fairly simple access procedures are often used. If an attacker is able to compromise a regular workstation within the company, access to other and more important machines could also be achieved. Eventually, the attacker can attempt to access the processing center through machines or user profiles that provide a certain access level.

The people accessing the processing center does so through technologies such as web interfaces, mobile applications, and possibly various network services. The attack surface and possible vulnerabilities of these are discussed in Sections 6.2.2, 6.2.3 and 6.2.5.

Third-party interface The processing center could utilize third-party software for numerous reasons. In some cases it could be software that is allowed to read and/or write data to and from the processing center, it could be software that provides a specially customized interface to an application, or even software that is installed in the processing center to perform some particular task or service. Either way, by identifying third-parties interacting with the processing center, an attack has an even greater attack surface area.

While third-party software in many cases is an easy, fast, and cheap way to add new functionality or increase the availability of a product, it can also introduce serious vulnerabilities to the system. When giving a third-party a set of permissions, the administrators of the application or the processing center should be extremely careful. An attacker could identify vulnerabilities directly in the third-party software

or in the network service it uses to communicate with the processing center. Also, as with the user interface, an indirect attack can be launched towards the processing center by attempting to compromise computers or machines within the third-party organization. More about these attack surface areas can be found in Sections 6.2.5.

Maintenance interface Because the processing center itself needs reconfigurations, patches, or new software from time to time, an interface for performing this should also be available. While the other interfaces mostly relate to the applications running in the processing center, this is an interface that interacts both physically and remotely, directly with the server or servers that are running the applications. Usage of this interface could be done by manufacturers of the hardware components that constitute the processing center, by consultants on behalf of manufacturers, or by IoT platform employees, for example.

As with the third-party interface and the user interface, an attacker could attempt to compromise insecure regular computers or machines that are used by the actors of the maintenance interface, and use these computers to access the processing center.

Notably, this interface could provide system access to privileged actors that are not directly under control by the IoT platform. As this could be uncomfortable to platform providers that are uncertain about the security of the privileged and ”uncontrolled“ actors, the option of disabling the entire interface is a possibility. However, as one of the purposes of the interface is to increase system security through allowing remote patching and upgrading of various platform components, disabling the interface is in a way counter-intuitive and the platform providers should be able to find a middle ground for this.

For this interface the relevant surface areas are related to physical access, web interfaces, and network services. Respectively, discussion on these subjects can be found in Sections 5.2.1, 6.2.2 and 6.2.5.

6.2.2 Web interfaces

As applications or system configuration modules are likely to have individual interfaces which are customized to the application’s or module’s functions and requirements, it is probable that the processing center hosts multiple sites that could contain various vulnerabilities.

The different users of the applications, such as administrators, customer support, and customers, might all be utilizing the same web interface when they interact with the application. Therefore, an inadequate distinction between user privileges could be a potential security breach. For example, actions that are available to a user of

type *administrator* should not only be hidden from a user of type *customer* but be blocked or unavailable through proper logical configurations.

In general, the same types of vulnerabilities as those identified in Section 5.2.2 about device web interfaces, can be applied to the web interfaces of the processing center. While the web interfaces of the end-nodes often are outdated, the web interfaces of the processing center are likely to have been developed particularly for the applications and could, therefore, be more recent and up-to-date with respect to common vulnerabilities. Thus, an attacker could need to invest more effort into discovering vulnerabilities that are specific to each web interface and are caused by insecure coding or choosing bad architectural design during development.

6.2.3 Mobile application

It is known that multiple IoT applications come with a customized mobile application as well as a web interface. Many times, the two can be equal, or at least similar, in terms of functionality, design, and security. Also, both can be used for reading data, performing configurations, executing procedures, for example. However, the mobile application could also relate entirely different to the processing center compared to the web interface and introduce a new attack area for an attacker.

Disclosure of services An attacker is able to download the mobile application from a mobile application store, such as *Google Play* or Apple's *App Store*, and investigate its functions and behavior. Through this, potential vulnerabilities could be searched for in the attacker's personal environment. The attacker can investigate traffic generated by the mobile phone the application is running on, try to decompile the application, and investigate what APIs it is using. If the application utilizes hardcoded passwords or unprotected services, the attacker could launch an attack towards the processing center by using the discovered information.

Relations to the ecosystem As with web interfaces, a mobile application is also vulnerable to use of weak passwords, lack of end-to-end encryption, lack of account lockout after failed login attempts, for example. If an attacker is able to exploit this, the available actions of the mobile applications become highly relevant to the security of the processing center. For example, if the mobile application is allowed to hand out permissions it can introduce new and potentially malicious devices into the ecosystem. Furthermore, the mobile application could potentially be used to control and configure the IoT application's behavior and it is therefore equally important to ensure proper security of the mobile application as any other part of the ecosystem.

6.2.4 Ecosystem communication

From being an IoT application, most of the communication to and from the processing center is generated and processed autonomously. Therefore, it is important that the entities within the IoT application behave as expected, such that the processing center is able to maintain the IoT applications and ensure that their functions are preserved. For example, if the processing center wants to request a ”status update“ or a ”heartbeat“ from the end-nodes, the application has defined procedures for how the request should look like, and how the end-nodes should reply.

Given that an attacker controls a legitimate end-node within an application, the attacker can investigate how the processing center responds to variation and modifications of the expected responses. If the developers of the application have offered too little attention to how the ”fundamental“ messages within the ecosystem are handled, potential attacks might exist when implicit trust in the ecosystem is exploited.

Furthermore, an attacker does not only need to modify responses to the request but might also try to invoke the processing center with malicious requests, uploads, error messages, etc. In such events, the processing center must be sure not to allow remote code execution, information leakage, or other security breaches due to not handling data with unexpected format properly.

6.2.5 Network services

There could be multiple network services running at the processing center, where some might be crucial for ensuring that the IoT application operates fully. First and foremost, the IoT end-nodes should have somewhere to transmit data continuously such that the processing center can keep the application’s status up-to-date. Moreover, service providers and other third-parties which are used by the processing center could also be in need of interacting with the some network service. This creates an attack surface area for an attacker.

In Section 5.2.3, possible weaknesses related to network services are identified and while these are presented in the context of IoT end-nodes, they are just as applicable to the processing center. Unintended open ports, lack of end-to-end encryption, etc. must be prevented in the processing center to avoid attackers from intruding the system.

Denial of service As Chapter 4 concludes, the availability of the processing center is essential for the IoT applications to operate. It is a very natural target for an attacker that wishes to cause as most damage as possible to the businesses using the IoT platform. Because the threshold for performing a DoS attack is very low, this is

perhaps one of the major threats and concerns for connecting multiple businesses together through a single IoT platform.

In addition to the DoS attack described in Section 5.2.3, distributed DoS (or DDoS) attacks are also commonly used by attackers to take out a site or service. There are multiple ways a DDoS attack can by performed. One of them simply requires that attackers organize themselves at forums or groups, and agree to perform individual DoS attacks at a particular time. Another way of conducting a DDoS attack is if an attacker is in control of a *botnet*. This is a network of infected, hacked, or controllable computers that are connected to the Internet. The attacker uses the botnet to attack the site or service by instructing all of them to perform a DoS attack. The effect is similar to when attackers coordinate an attack, but it is different in the sense that the owners of the computers which are performing the attack are likely to be unaware that they are participating in a DDoS attack.

Malicious third-parties and service providers While the previous section, Section 6.2.4, discuss that the processing center should be able to handle end-nodes that are behaving unexpectedly and potentially are in control of an attacker, this also goes for the third-parties and services utilized by the processing center. The processing center must take into account that an attacker could try to compromise these third-parties first and afterward launch an attack towards the processing center. Potentially, an attack could be performed through creating malicious versions of the resources or services being used by the processing center or the attack can try to exploit permissions to access data which actually should have been unavailable to the third-party.

Additionally, an attacker could try to identify the network services being used by third-parties and mimic a particular third-party's behavior. This does not require that the attacker compromises the third-party, but would rely on that the authentication between the third-party and the processing center is weak or non-existing.

6.2.6 Outsourcing

Although outsourcing can be used by the IoT platform to lower the cost for development, operations, and maintenance, it could also inflict a reduced overview of the state of the system. Potentially, outsourcing could, for example, cause leakage of sensitive information, data manipulation, and information theft.

When any of the mentioned actions are performed by an external party, which could be foreign or domestic, instead of the IoT platform itself, the overall security becomes significantly harder to manage. Agreements regarding this issue can be made with the external party, but as outsourcing moves the entire process of, for

example, development to an entirely different location, it is rather difficult to monitor the procedures closely enough.

Furthermore, outsourcing could lead to both intended or unintended security breaches. Employees within the external party might leak or sell information to attackers, which further use the information to compromise the processing center directly or indirectly. Additionally, the external party could have own security issues that can be identified by an attacker. If exploiting these and gaining access to the processing center, an attacker has effectively compromised the IoT platform due to the uncertainties that comes with outsourcing. The principle is similar to vulnerabilities introduced by using third-party software, but with outsourcing, this happens at a much larger scale and with even less control of the system's security.

6.3 Attack scenarios

As with the attack scenarios of Chapter 5, also here some scenarios are provided to give a brief illustration of how the attack surface might be used by an attacker that has an aim of compromising the application.

6.3.1 Accessing the processing center through a smart meter

In this scenario, a smart meter from the example application in Section 3.3.1 is used by an attacker to exploit a vulnerability in the processing center's end-node interface. Effectively, a smart meter is assumed to have been compromised by an attacker, and the attacker is able to control data sent to the processing center.

Properly secured API To enable the end-nodes to read and upload data to the processing center, the end-nodes have a dedicated API for this purpose. Through authentication mechanisms, this API is only accessible to legitimate end-nodes, and if attempting to access the API without an end-node, the attacker only receives a "forbidden" message. Furthermore, the attacker could attempt to identify the authentication procedure, and perform attacks on this to gain access without controlling an end-node, but in this case, it is assumed that adequate authentication mechanisms are in place and that the attacker cannot proceed.

Ecosystem trust and lack access control As the attacker is in control of an end-node, the application is already compromised to some degree, but the attacker can attempt to utilize the foothold in the IoT platform to elevate privileges in the processing center. Given that the API is designed to receive requests and provide responses based on URLs, the attacker could attempt to exploit lack of access control

and proper sanitation of the requests by hiding a *path traversal attack*[3], for example, within the request parameters.

If the access controls of the system are misconfigured or the authentication procedure for the end-nodes' API hand out too high privileges, the attacker could attempt to disclose files that are critical to the application or the underlying system. Further, the attacker can use these to introduce backdoors or permanently elevate privileges. For example, by using path traversal attack, the attacker could gain access to modifying system files that contain lists of usernames and passwords. Then, new users with chosen passwords can be introduced in these files. While stricter access policies could solve this issue, it is also an issue of the processing center not expecting an authenticated end-node to behave differently from the standard and predefined actions. Thus, it fails to handle the malicious request securely.

6.3.2 Accessing database of connected health equipment

In this scenario, the connected health equipment application of Section 3.3.2 is compromised through improper implementations of access control in the maintenance interface. It is assumed that only maintenance-personal of the equipment's manufacturer are supposed to have access to a secondary system of the connected health equipment. This secondary system is hosted alongside the primary application. The system does not contain all of the operational features that can be found in the primary application, but only functions that are of value with regards to maintenance. For example, it would know the equipment' operating hours, their location, and the types of medicine they are used for.

Lack of restricted access As the maintenance system was designed to operate as an internal system, it lacks authentication mechanisms and it produces and displays internal error messages during execution. Therefore, the processing center must implement access control to avoid attackers or arbitrary people from locating and using the service unauthorized. To enforce this, the processing center has a blockade (typically a *firewall*, see Section 6.4.4) for incoming request to this system. However, to allow maintenance personnel access, the blockade has been configured to allow requests from a given range of IP-addresses that are supposed to correspond with the IP-addresses used by the manufacturer. Unfortunately, this configuration has been implemented wrongly, and an attacker is able to bypass the blockade without originating from any of the intended IP-addresses.

[3] A path traversal (or directory traversal) attack aims to access files and directories that are stored outside the web root folder by including "dot-dot-slash (../)" in variables that are interpreted by the application [Con15].

Locating the database After locating and accessing the system, the attacker is able to perform multiple actions that could affect the primary application and cause harm to people or material. But as the maintenance system also provides internal error messages, the attacker is able to determine the internal address of a database after intendedly causing an error to occur.

Crafting an exploit The attacker can continue to look for sensitive information in various error messages and possibly find a way to compromise the database or other components in the processing center. For every discovered resource, there exist multiple possible vulnerabilities which the attacker can attempt to exploit. As a significant error regarding access control already has been uncovered, it is not unlikely that similar mistakes have been made.

From this scenario, the importance of securing all the initial four interfaces is illustrated as an attacker almost is expected to be able to identify all of them when performing reconnaissance of the processing center. While the user interface and the end-node interface might be the easiest to locate, the others are just as likely to contain vulnerabilities and misconfigurations that are exploitable to an attacker.

6.4 Security mechanisms

From the sections about the attack surface and the attack scenarios, multiple requirements for security have been identified and possible vulnerabilities have been given as examples. In this section, security mechanisms that can prevent the relevant security breaches for the IoT processing center are presented. While most of the concepts presented are common components of other computer systems as well, here only the mechanisms that are of relevance to the IoT platform are included.

6.4.1 Access control

As with the IoT end-nodes, ensuring proper access control mechanisms is essential for making the processing center secure against attackers that investigate every possible entry. In Section 5.4.2, the requirements for a proper access control systems are described, and the same type of access control should be present at the processing center such that applications can be hosted securely.

Furthermore, it is not only the fine-grained access control between legitimate users with different levels of privilege that should be present, but also general restriction from public access needs to be preserved. Because the processing center exposes multiple interfaces and services to the various actors within the applications, there is a potential risk that some are misconfigured and expose entire or parts of services to the public. As Sections 5.2 and 6.2 describe, there are multiple ways for an attacker

to disclose APIs and resources being used by end-nodes, mobile applications, third-parties, for example. Once discovered, the attacker can use network enumeration, directory brute force, and other automated procedures to easily identify if there exist unprotected parts of the system that can be accessed. Consequently, it is of great importance to employ a minimum level of access control.

6.4.2 Authentication of end-nodes

Because the amount of individual messages received by the processing center is correlated with the number of end-nodes, this might be extremely high. It is difficult to filter out messages that appear to be falsified or incorrect simply by inspecting the data, as this would require sophisticated processing of all messages before accepting them. Consequently, authentication of the end-nodes could be a more accurate and efficient way of preventing attackers from sending malicious requests or data to the processing center.

By ensuring that each end-node has an individual set of credentials, the processing center can easily verify that the counterpart is a legitimate end-node and accept messages received from this origin. However, in many cases, the end-nodes will only transmit single messages periodically. To perform an authentication handshake, a procedure for authenticating and establishing a session key, for each message could be very inefficient because of the overhead it causes. Therefore, using authentication of messages, such as described in Section 5.4.6, is a possible security mechanism for ensuring that the messages sent to the processing center originate from a legitimate source.

6.4.3 Two-factor authentication of users

To prevent an attacker from exploiting weak credentials, lack of account lockout, or possible brute force in the login procedure, two-factor authentication should be employed by web interfaces and mobile applications. When using two-factor authentication, the person which attempts to authenticate is often first asked to enter a long-term set of credentials, such as a username and a password. Afterward, the user is prompted to enter a one-time token, which is received through a *side-channel* or computed by a *dongle*. While the side-channel, for example, could be an email or an SMS, a dongle could be a hardware device or a piece of software that generates a new token on given intervals.

The intention with two-factor authentication is not to permanently remove the risk for an attacker being able to perform an unauthorized login, but rather to increase the amount of effort needed by the attacker to do so. There are no guarantees that a dedicated attacker would not be able to take control of the victim's mobile phone, email, or dongle, but two-factor authentication forces the attacker to target

individuals rather than a group and requires that the attacker compromises the side-channel or obtains the dongle in some unauthorized manner.

6.4.4 Firewall

In computing, a firewall is a network security system that monitors and controls the incoming and outgoing network traffic based on predetermined security rules [Bou10]. Typically, a firewall establishes a barrier between two networks, where the inside network is a trusted and safe environment that needs protection from the outside network, an untrusted environment. Firewalls can operate on various layers and in various location in the network, such that customized protection is ensured where it is needed. There might be default rules within the firewall that apply, otherwise, it is up to the firewall administrator to define the rules for how the firewall should operate. [Con16f, Aud12]

Network layer A firewall on the network layer, also called a packet sniffer, can filter packets based on many packet attributes. These attributes might be the IP-address and port of the source and destination, destination service, and many more. The firewall can be both stateful and stateless, which indicates whether or not it preserves information about ongoing sessions between the two networks. Although a stateless firewall uses less memory than a stateful firewall, it needs to compare all packets against all the rules. Therefore, the stateless could be slower than the stateful, which lets all packets within ongoing sessions past without comparing them to any rules at all.

Application layer A firewall on the application layer may inspect all traffic traveling to and from an application, such as browser traffic, FTP, SMTP, and scan for improper content. Consequently, an application layer firewall may help to prevent the spreading of computer worms, trojans, and similar malicious codes, but also be used for dropping traffic with misshaped headers, flaws, or illegal values such that correct behavior and unnecessary processing by the receiving process is assured.

Proxy By using a proxy server, a server that inspects and forwards all traffic to the actual server, filters can be established on the application layer such that only legitimate traffic, which is specified beforehand, reaches the actual server. The proxy server then acts as a client to the actual server, and as a server to the actual client.

Usage of the various types of firewalls is essential to ensure that the processing center is protected against attacks and that actors in the IoT platform do not step over the boundaries they should remain within.

6.4.5 Intrusion detection systems

An Intrusion Detection System (IDS) is used to monitor network or system activities for malicious activities or policy violation and notify system administrators about suspicious incidents or actual indications of security breaches. An IDS can monitor activities that both enter a system or originate from within a system, but does not react to a possible violation like an intrusion prevention system (IPS) does. Instead, logs are created and electronic reports are produced [Con16g].

Anomaly-based IDS A method for detecting possible breaches in security is to look for anomalies in the network or system traffic and use heuristic rules for determining if there is a reason for alerting the system administrators. Such an approach requires that the IDS learns the normal system behavior and adapts adequate methods for detecting actual anomalies. Often, artificial intelligence is used in the detection [WS04].

Signature based IDS With a signature based IDS, the IDS does not compare traffic against the normal behavior of the system or a generated set of rules that *could* indicate an intrusion. Instead, the IDS is regularly updated with a set of known patterns or signatures and reacts only if any of the monitored traffic matches a previously detected intrusion [WS04].

By using an IDS, the processing center has an increased probability of detecting a successful attack if there exist vulnerabilities that are unknown to the system administrators and exploitable by an attacker. An IDS could especially be of great importance if the attacker attempts to escalate privileges or overtake other systems than the one that was compromised initially, as this would require the use of known attack vectors or generate at least some irregular system traffic.

6.4.6 Encryption of data

To prevent attackers from monitoring the data sent to and from the applications, encryption should be employed by the processing center and the end-nodes. While an attacker could attempt to intercept messages at various locations within the infrastructure, the easiest and most natural location to perform such an attack would be close to the end-nodes, by acting as a false access point. Details regarding encryption of data is found in Section 5.4.5.

6.4.7 Data integrity and message authentication

Whenever an actor within the IoT platform interacts with the processing center, the integrity of the data should be preserved. This involves that both the actor and the processing center are assured that the message they receive has not been

altered in transit and that it originates from the claimed source. As this already was established for the end-nodes in Section 5.4.6, details on how to comply with the requirements are found there.

The integrity mechanisms are important to the processing center because they allow the applications to accept and process the content of data readings from the end-nodes without having to verify the correctness of their values. Furthermore, the security mechanisms also allow the processing center to authenticate and to integrity-protect its own messages, thus preventing that end-nodes are misled into sharing data or sensitive information with an attacker that tries to impersonate the processing center. Eventually, usage of these mechanisms strengthens the overall security of the entire IoT platform.

Chapter 7
Conclusion and Future Work

7.1 Conclusion

In the previous chapters, the concept of an IoT platform has been introduced and analyzed by using graph theory and traditional information security theory. Additionally, the chapters have pointed out particularities of the IoT that should be accounted when considering the security of an IoT system in general.

7.1.1 An IoT platform used by multiple applications

The concept of an IoT platform was introduced based on the increased emergence of IoT applications and the situation of TELCOs regarding M2M. Industries and businesses are realizing that the IoT not only can be used for light bulbs and refrigerators, but it can also improve operations in areas such as quality, efficiency, and cost. At the same time, TELCOs find that they already have the infrastructure needed to connect near and distant IoT components together through various network technologies. Furthermore, it is when a TELCO establishes a service like a processing center that an IoT platform takes shape. By hosting the central services of IoT applications, the IoT platform is able to provide connectivity, processing and distribution of information to the end-nodes and users of the applications.

An IoT platform can provide services to multiple applications simultaneously and independent of what industry or business area they operate within. Companies who seek out the possibility of digitizing operations and processes could find it tempting to outsource these to an IoT platform provider. By letting the platform manage operations and issues such as maintenance, security, and patching, companies can focus on their core activities instead of putting efforts into acquiring or developing the necessary competence internally.

While gathering multiple IoT applications on a single platform could benefit both the provider and the customers, some considerations and questions have been

raised regarding how this could affect the availability and overall security of the applications. For example, how does the platform prevent information leakage between the customers and their applications? Also, as there potentially could be tens or hundreds of millions of end-nodes connected to the platform, does the system scale to this and do the applications utilize mechanisms to prevent the end-nodes from overflowing the central processing? Are cascading failures handled such that an error in one application does not affect others?

7.1.2 Graph theory analysis

To identify how the availability of the IoT applications could be affected by connecting them to a common platform, an analysis was performed with the use of graph theory. By using some simplification, the analysis identified network topologies of the IoT platform, the Internet, and the redundant solutions of the processing center. Further, graph theory was used to describe what parts of the networks that were essential to prevent the availability of the IoT applications from being greatly impacted.

Through the analysis, it became clear that a simplified model of the IoT applications was highly vulnerable to targeted attacks as they have an obvious single point of failure. Additionally, the Internet, which can be regarded as the main component of an IoT platform's infrastructure, was analyzed as a graph and also found to be vulnerable to targeted attacks. However, while the Internet is a huge network which is scaled for an enormous amount of traffic, an attacker that attempts to cause damage to the IoT applications would be way more effective if targeting the IoT platform's processing center than targeting certain hubs in the Internet.

Given that the businesses that utilize a common IoT platform are unable to operate properly without their digital IoT systems, a common dependency between them is established through the platform. An attacker could attack all of the companies and businesses at the same time by striking their digital operations. Thus, even if they initially were entirely unrelated in terms of industry or business area, they can now be affected by a single incident.

To prevent targeted attacks from being able to easily take out the processing center, the analysis further suggests that redundant nodes can be introduced in the network. Demonstratively, a distributed solution and a solution using backup nodes were presented. Both showed to form a graph with improved robustness and reduced vulnerability compared to the graph of the IoT platform's structure without redundancy of the processing center.

7.1.3 Information security analyses

To identify how an attacker could attempt to compromise generic IoT systems or applications running on an IoT platform, the attack surface areas of such IoT applications were presented and discussed. Both the end-nodes and the processing center of an IoT platform were considered and potential vulnerabilities were introduced to clarify how the surface areas could be breached. Relevant security mechanisms were discussed as an illustration of possible measure against cyber attacks.

End-nodes Through the information security analysis of the end-nodes of the applications, multiple possible attack vectors were identified and discussed. Additionally, motivations for attacking an end-node were introduced. While ”traditional“ attack vectors, such as exploiting web interface vulnerabilities, lack of transport encryption, etc., were included in the analysis, more IoT-specific attack vectors were also identified, such as ecosystem communication, insecure update mechanisms and use of insecure access technologies.

Furthermore, factors such as that the end-nodes in many applications are physically accessible to attackers, have limited resources, or could have their security neglected during manufacturing were all accounted for while assessing potential threats and their corresponding security mechanism. It was found that an IoT end-node has a broad attack surface and involves multiple technologies. Eventually, possible security mechanisms were presented to all of the surface areas.

Processing center As the processing center and the end-nodes of an IoT platform share some common attack surface areas, they also share some potential weaknesses. However, the processing center was identified to be especially exposed to inadequate access control and misconfigurations of the multiple interfaces it exposes to actors within the platform. Because these actors require various privileges depending on their role in the system and also could be using different authentication services, an attacker has a huge area to explore for vulnerabilities and holes. This makes it difficult for system administrators and developers to ensure that no exploitable errors exist.

Because of the huge amount of possible entry points an attacker has to the system, it is also important to consider the usage of intrusion detection systems, or similar mechanisms, to look for indications of successful attacks having occurred. These systems can be used to uncover attempts, or successful attempts, of accessing the processing center unauthorized. They enable the possibility to identify breaches when they actually happen, not after the attacker decides to demonstrate to the world that the IoT platform has been compromised.

7.2 Future work

This study does not go in depth of the various attack surface areas that are identified to be relevant for IoT applications or empirically attempts to identify weaknesses or vulnerabilities in actual applications. By assessing actual applications and using this thesis as a framework, future studies could quantitatively explore vulnerabilities in particular surface areas, and contribute to establishing even more precise descriptions of possible attack vectors.

To be able to secure IoT applications better, future work could also take on subjects such as how to optimize encryption of sensor data, better prevention of traffic analysis, and the establishment of integrity combined with trust in the IoT.

Furthermore, to better understand how the failure of various nodes in an IoT platform could affect the applications running on it, simulations of both random and targeted attacks could be made. Topologies of the structures that are identified in this study could be used.

References

[AB16] Telenor Connexion AB. *Addressing a global health concern.* http://www.telenorconnexion.com/stories/srett-medical, 2016. Accessed: 2016-02-12.

[AGS09] Jan A. Audestad, Inge Grønbæk, and Stein Svaet. *Connected Objects Platform Specification.* Telenor, 2009.

[ASS09] Zahra Ahmadian, Somayeh Salimi, and Ahmad Salahi. *New attacks on UMTS network access.* IEEE, 2009.

[Aud11] Jan A. Audestad. *E-Bombs and E-Grenades: The Vulnerability of the Computerized Society, Lecture Notes: IMT4481 Information Society and Security.* Gjøvik University College, 2011.

[Aud12] Jan A. Audestad. *Network security.* Gjøvik University College, 2012.

[BB03] Albert-László Barabási and Eric Bonabeau. *Scale-free.* Scientific American, 2003.

[Bol98] Béla Bollobás. *Modern graph theory.* Springer, 1998.

[Bou10] Noureddine Boudriga. *Security of mobile communications.* CRC Press, 2010.

[CDKB11] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design.* Addison-Wesley Publishing Company, 5th edition, 2011.

[CH03] R. Cohen and S. Havlin. *Scale-Free Networks Are Ultrasmall.* Phys. Rev. Lett., 2003.

[CHWW03] Nancy Cam-Winget, Russell Housley, David Wagner, and Jesse Walker. *Security flaws in 802.11 data link protocols. Commun. ACM*, 2003.

[Com15] Computerworld. *Researchers hack a pacemaker.* http://www.computerworld.com/article/2981527/cybercrime-hacking/researchers-hack-a-pacemaker-kill-a-man-nequin.html, 2015. Accessed: 2016-05-22.

[Con15] OWASP Contributors. *Path traversal.* https://www.owasp.org/index.php/Path_Traversal, 2015. Accessed: 2016-05-10.

[Con16a] Wikipedia Contributors. *Access control*. https://en.wikipedia.org/wiki/Access_control, 2016. Accessed: 2016-04-22.

[Con16b] Wikipedia Contributors. *Albert-László Barabási*. https://en.wikipedia.org/wiki/Albert-L%C3%A1szl%C3%B3_Barab%C3%A1si, 2016. Accessed: 2016-02-25.

[Con16c] Wikipedia Contributors. *Authenticated encryption*. https://en.wikipedia.org/wiki/Authenticated_encryption, 2016. Accessed: 2016-04-26.

[Con16d] Wikipedia Contributors. *Client-server model*. https://en.wikipedia.org/wiki/Client%E2%80%93server_model, 2016. Accessed: 2016-03-10.

[Con16e] Wikipedia Contributors. *Encryption*. https://en.wikipedia.org/wiki/Encryption, 2016. Accessed: 2016-04-26.

[Con16f] Wikipedia Contributors. *Firewall*. https://en.wikipedia.org/wiki/Firewall_(computing), 2016. Accessed: 2016-05-06.

[Con16g] Wikipedia Contributors. *Intrusion detection system*. https://en.wikipedia.org/wiki/Intrusion_detection_system#Host_Intrusion_Detection_Systems, 2016. Accessed: 2016-05-02.

[Con16h] Wikipedia Contributors. *Machine to machine*. https://en.wikipedia.org/wiki/Machine_to_machine, 2016. Accessed: 2016-03-14.

[Con16i] Wikipedia Contributors. *Matthew effect*. https://en.wikipedia.org/wiki/Matthew_effect, 2016. Accessed: 2016-03-31.

[Con16j] Wikipedia Contributors. *Message authentication code*. https://en.wikipedia.org/wiki/Message_authentication_code, 2016. Accessed: 2016-04-26.

[Con16k] Wikipedia Contributors. *Network service*. https://en.wikipedia.org/wiki/Network_service, 2016. Accessed: 2016-05-27.

[Con16l] Wikipedia Contributors. *Scale-free network*. https://en.wikipedia.org/wiki/Scale-free_network, 2016. Accessed: 2016-03-01.

[Con16m] Wikipedia Contributors. *Service level agreement*. https://en.wikipedia.org/wiki/Service-level_agreement, 2016. Accessed: 2016-05-09.

[Con16n] Wikipedia Contributors. *Telecommunications operator (TELCO)*. https://en.wikipedia.org/wiki/Telecommunications_operator, 2016. Accessed: 2016-05-24.

[Con16o] The Conversation. *Cyberattack on Ukraine grid*. http://theconversation.com/cyberattack-on-ukraine-grid-heres-how-it-worked-and-perhaps-why-it-was-done-52802, 2016. Accessed: 2016-03-15.

[Cor16a] EMC Corporation. *What is tamper-resistant hardware?* http://www.emc.me/emc-plus/rsa-labs/standards-initiatives/what-is-tamper-resistant-hardware.htm, 2016. Accessed: 2016-04-25.

[Cor16b] Microsoft Corporation. *Microsoft - Cloud Platform.* http://www.microsoft.com/en-us/server-cloud/, 2016. Accessed: 2016-01-28.

[CS16] Inc. Cisco Systems. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update.* http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, 2016. Accessed: 2016-02-05.

[CSN09] A. Clauset, C. R. Shalizi, and M. E. J. Newman. *Power-Law Distributions in Empirical Data. SIAM Review*, 2009.

[DR11] Thai Duong and Juliano Rizzo. *Here Come The XOR Ninjas.* 2011.

[Hen14] Mike Hendry. *Near Field Communications Technology and Applications.* Cambridge University Press, 2014.

[HHR+08] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. IEEE Computer Society, 2008.

[HTM+14] Jan Höller, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, and David Boyle. *From Machine-to-Machine to the Internet of Things - Introduction to a New Age of Intelligence.* Academic Press, 2014.

[Inc16] Amazon.com Inc. *Amazon Web Services.* https://aws.amazon.com/iot/, 2016. Accessed: 2016-01-28.

[Ins15a] EY Insights. *Cyber threat intelligence.* http://www.ey.com/Publication/vwLUAssets/EY-how-do-you-find-the-criminal-before-they-commit-the-cybercrime/$FILE/EY-how-do-you-find-the-criminal-before-they-commit-the-cybercrime.pdf, 2015. Accessed: 2016-05-02.

[Ins15b] EY Insights. *Cybersecurity and the Internet of Things.* http://www.ey.com/Publication/vwLUAssets/EY-cybersecurity-and-the-internet-of-things/$FILE/EY-cybersecurity-and-the-internet-of-things.pdf, 2015. Accessed: 2016-03-17.

[KSBB10] Christoph Kemetmüller, Mark M. Seeger, Harald Baier, and Christoph Busch. *Manipulating Mobile Devices with a Private GSM Base Station - A Case Study.* University of Plymouth, 2010.

[Lov11] Lovdata.no. *Regulations governing metering, settlement, billing of network services and electrical energy.* https://lovdata.no/dokument/SF/forskrift/1999-03-11-301/kapittel_4#kapittel_4, 2011. Accessed: 2016-02-09.

[Mie15] Daniel Miessler. *IoT Attack Surface Mapping.* https://www.owasp.org/images/3/36/IoTTestingMethodology.pdf at DEFCON 23, 2015. Accessed: 2016-03-31.

[Mil67] Stanley Milgram. *The Small World Problem.* Psychology Today, 1967.

[MvOV97] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[MW04] Ulrike Meyer and Susanne Wetzel. *A man-in-the-middle attack on UMTS*. ACM, 2004.

[Org16] OWASP Org. *Internet of Things Project*. https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project, 2016. Accessed: 2016-04-05.

[oST15] National Institute of Standards and Technology. *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar1.pdf, 2015. Accessed: 2016-04-28.

[Pat15] Al-Sakib Khan Pathan. *Securing Cyber-Physical Systems*. CRC Press, 2015.

[plc16] Laird plc. *What is the connected hospital?* http://www.lairdtech.com/solutions/embedded-wireless/what-connected-hospital, 2016. Accessed: 2016-02-12.

[Rya13] Mike Ryan. *Bluetooth: With Low Energy Comes Low Security*. USENIX Association, 2013.

[Shi] R. Shirey. *Internet Security Glossary*. RFC 2828.

[Sta11] William Stallings. *Cryptography and Network Security - Principles and practice (5.ed.)*. Prentice Hall, 2011.

[WAD09] Walter Willinger, David Alderson, and John C Doyle. *Mathematics and the internet: A source of enormous confusion and great potential*. Defense Technical Information Center, 2009.

[Wir15] Wired. *Hackers Can Disable a Sniper Rifle – Or Change Its Target*. https://www.wired.com/2015/07/hackers-can-disable-sniper-rifleor-change-target/, 2015. Accessed: 2016-05-22.

[WS04] Ke Wang and Salvatore J. Stolfo. *Anomalous Payload-Based Network Intrusion Detection*. Springer, 2004.

[WYY05] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. *Finding Collisions in the Full SHA-1*. Springer, 2005.