TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

OLLI PENTTILÄ

CYBER THREATS IN MARITIME CONTAINER TERMINAL AUTOMATION SYSTEMS

Master of Science Thesis

Examiner: Professor Jarmo Harju
The examiner and topic of the thesis were approved by the Council of the Faculty of Computing and Electrical Engineering on 3.2.2016

ABSTRACT

OLLI PENTTILÄ: Cyber Threats in Maritime Container Terminal Automation Systems
Tampere University of Technology
Master of Science Thesis, 91 pages, 10 Appendix pages
February 2016
Master's Degree Programme in Information Technology
Major: Information Security
Examiner: Professor Jarmo Harju

Keywords: Information Security, cyber security, cyber threat, Industrial Control System, attack tree

The rapid development in connectivity of Industrial Control Systems has created a new security threat in all industrial sectors, and the maritime sector is no exception. Therefore this thesis explores cyber threats in a container terminal automation system using two methods: literature review and attack tree analysis.

In this thesis, cyber threats in Industrial Control Systems were first studied in general by the means of a literature review. Then, the identified threats were applied to a software component of a terminal automation system using attack trees. Attack trees are a tool that helps in visualizing different cyber attacks. Based on the results, threats were classified in risk categories and the most problematic areas were identified. Finally, suggestions were made on how to improve cyber security of the component assessed and of the terminal automation system in general.

Based on the literature review, ten different risk categories were identified. The categories cover various attacks ranging from malware and Denial-of-Service attacks all the way to physical and social attacks. When assessing the software component, three problem areas were identified: susceptibility to Denial-of-Service attacks, weak protection of communication and vulnerability of a certain software sub-component. The suggested security improvements include changes to the network design, use of stronger authentication and better management of the process automation network.

Based on the study, container terminal automation systems aren't very different from other Industrial Control Systems in terms of cyber security, as they are susceptible to the same threats. The current cyber security posture of Industrial Control Systems is considered relatively poor, and container terminal automation systems are no exception. Therefore it is important that companies involved in the industry commit to improving cyber security either voluntarily or as obligated by laws and regulations.

TIIVISTELMÄ

OLLI PENTTILÄ: Kyberuhat konttisataman automaatiojärjestelmässä
Tampereen teknillinen yliopisto
Diplomityö, 91 sivua, 10 liitesivua
Helmikuu 2016
Tietotekniikan diplomi-insinöörin tutkinto-ohjelma
Pääaine: Tietoturvallisuus
Tarkastaja: professori Jarmo Harju

Avainsanat: tietoturvallisuus, kyberturvallisuus, kyberuhka, teollisuusautomaatio, hyökkäyspuu

Kommunikointitarpeiden nopea kasvaminen automaatiojärjestelmissä on luonut uuden turvallisuusuhan kaikilla teollisuuden aloilla, eikä merisektori tee tähän poikkeusta. Siksi tässä diplomityössä tarkastellaan kyberuhkia satama-automaatiojärjestelmässä kahta tutkimusmenetelmää käyttäen.

Tässä työssä kartoitettiin aluksi automaatiojärjestelmien kyberuhkia yleisellä tasolla kirjallisuuskatsauksen muodossa. Tämän jälkeen tunnistettuja uhkia sovellettiin yksittäiseen satama-automaatiojärjestelmän ohjelmistokomponenttiin hyökkäyspuita apuna käyttäen. Hyökkäyspuuanalyysi on työkalu, jota käytetään kyberhyökkäysten rakenteen visualisoimiseen. Puihin perustuen uhat lajiteltiin riskiluokkiin ja järjestelmän ongelma-alueet tunnistettiin. Lopuksi annettiin suosituksia siitä, miten tarkastellun komponentin ja koko järjestelmän kyberturvallisuutta voitaisiin parantaa.

Kirjallisuuskatsauksen perusteella tunnistettiin kymmenen erilaista riskikategoriaa, jotka kattavat hyvin erityyppisiä hyökkäyksiä aina haittaohjelmista ja palvelunestoista fyysisiin ja sosiaalisiin menetelmiin asti. Työssä tarkemmin tarkastellussa automaatiojärjestelmän ohjelmistokomponentissa havaittiin kolme ongelma-aluetta: heikko kyky sietää palvelunestohyökkäyksiä, kommunikaation vähäinen suojaus sekä yksittäisen ohjelmistokomponentin haavoittuvuudet. Kyberturvallisuuden parantamiseksi suositeltiin esimerkiksi muutoksia verkkoarkkitehtuuriin, vahvempien autentikointimenetelmien käyttöä sekä automaatioverkon hallinnan kehittämistä.

Tutkimuksen perusteella satamien automaatiojärjestelmät eivät kyberturvallisuuden suhteen juurikaan poikkea muista teollisuuden automaatiojärjestelmistä, vaan altistuvat samoille uhille. Teollisuusautomaation kyberturvallisuuden nykytasoa pidetään yleisesti heikkona, eikä satama-automaatio tee tässä poikkeusta. Tästä syystä on tärkeää, että alan toimijat kehittävät kyberturvallisuutta joko oma-aloitteisesti tai lakien ja säädösten velvoittamina.

PREFACE

The topic of this thesis was given by Kalmar Global. I would like to thank Tommi Pettersson and Pekka Yli-Paunu of Kalmar for giving me this opportunity and a very interesting topic. I would also like to thank them for their insights on the topic and for trusting me with all the information about their terminal automation system. A big thank you also to everyone at Kalmar who kindly took the time to discuss with me and gave answers to all my questions.

I warmly thank Professor Jarmo Harju of Tampere University of Technology, who first brought this opportunity into my knowledge and then on multiple occasions provided excellent feedback on both the structure and the contents of this thesis.

Tampere, 14th of February 2016

Olli Penttilä

CONTENTS

1. INTRODUCTION 1
2. BACKGROUND 3
2.1 Cyber threats 3
2.1.1 Vulnerabilities and exploits 4
2.1.2 Cyber attacks 5
2.1.3 Cyber attackers 6
2.2 Cyber security in Industrial Control Systems 7
2.2.1 ICS networks as targets 7
2.2.2 IACS cyber security standards and certificates 9
2.3 Automated container handling 10
2.4 Related work 12
3. THREAT LANDSCAPE 14
3.1 Malware 14
3.1.1 Viruses and worms 15
3.1.2 Trojan horses 15
3.1.3 Watering holes 15
3.1.4 Implications for ICS networks 16
3.2 Denial of Service 16
3.2.1 Flooding and nuking 17
3.2.2 Jamming 17
3.2.3 Implications for ICS networks 18
3.3 Spoofing 19
3.3.1 Packet spoofing 19
3.3.2 Implications for ICS networks 19
3.4 Unauthorized access 20
3.4.1 Passwords 20
3.4.2 Authentication systems 21
3.4.3 Privilege escalation and lateral movement 21
3.4.4 Physical access 22
3.4.5 Remote access 22
3.5 Software vulnerabilities 23
3.5.1 Different types of software 23
3.5.2 Buffer overflows 24
3.5.3 Reverse engineering 25
3.5.4 Implications for ICS networks 25
3.6 Hardware vulnerabilities 26
3.6.1 Programmable Logic Controllers 26
3.6.2 Tampering 26
3.6.3 Implications for ICS networks 27

3.7 Networking vulnerabilities 27
3.7.1 Routers and switches 27
3.7.2 Firewalls 28
3.7.3 Network security tools 29
3.7.4 Wireless networks 29
3.7.5 Network protocols 30
3.7.6 Man in the Middle 30
3.8 Misuse of process automation data 31
3.9 Data breach 32
3.9.1 Complicated attacks 32
3.9.2 Browser-based attacks 32
3.9.3 Injections 32
3.9.4 Phishing 33
3.9.5 Data breach in reconnaissance phase 33
3.10 System users 34
3.10.1 Portable devices 35
3.10.2 Social engineering 36
3.11 Cyber threats related to cloud services 36
4. DETAILED THREAT ANALYSIS OF EIS 38
4.1 External Interface Service 38
4.2 Motivation for attacking EIS 40
4.2.1 Interruption of container handling process 40
4.2.2 Stealing process-related data 40
4.2.3 Stealing a container 41
4.2.4 Moving a container through the terminal undetected 41
4.3 Cyber security concerns in EIS 41
4.3.1 Software-related 42
4.3.2 Network-related 46
4.3.3 Physical and human-related 48
4.4 Threat assessment tools 49
4.5 Threat assessment 52
4.5.1 Attack trees 52
4.5.2 Threat classification 52
4.5.3 Attack scenarios 54
4.6 Countermeasures 58
4.6.1 System-wide countermeasures 59
4.6.2 EIS-specific countermeasures 69
4.7 Summary 73
5. CONCLUSIONS 78
REFERENCES 80

APPENDIX A: ATTACK TREES

LIST OF FIGURES

Figure 1. The difference between information security and cyber security. Adapted from von Solms & van Niekerk [6]. 4
Figure 2. Timeline of a vulnerability lifecycle. Adapted from Dumitras [7]. 5
Figure 3. The phases of a cyber attack. Based on Hutchins et al. [10]. 5
Figure 4. Location of a container handling system in the logistic chain of a port. 10
Figure 5. Hierarchy of the four core layers in the automated container handling system of the case example. 11
Figure 6. Location of EIS within the architecture. 38
Figure 7. Components of the OSGi framework. Adapted from OSGi Alliance [74]. 39
Figure 8. Example of an attack tree. 50
Figure 9. An attack vector for a DoS attack 55
Figure 10. An attack vector for data leakage 56
Figure 11. An attack vector for moving Container Handling Equipment 58
Figure 12. ICS system security program framework. Adapted from Kilman & Stamp [109] 67
Figure 13. Threat landscape of a terminal automation system. 74

Figure A-1. Attack tree: gaining knowledge on target system. 92
Figure A-2. Attack tree: unauthorized remote access. 93
Figure A-3. Attack tree: unauthorized physical access. 94
Figure A-4. Attack tree: flooding attacks and communication interference. 95
Figure A-5. Attack tree: physical damage and editing container map. 96
Figure A-6. Attack tree: nuking attacks. 97
Figure A-7. Attack tree: move CHE by injecting instructions to the ICS network. 98
Figure A-8. Attack tree: move CHE through OSGi exploit. 99
Figure A-9. Attack tree: stealing container information and process data. 100
Figure A-10. Attack tree: more ways to steal container information. 101

LIST OF TABLES

Table 1. Difficulty rating for attack tree nodes. Based on Byres et al. [94] *50*
Table 2. Description of different impact levels for attack vectors. *51*
Table 3. Attack vector enumeration and scoring. *53*
Table 4. Risk matrix of the attack vectors. *54*
Table 6. Prioritization of attack vectors for mitigation. *75*
Table 7. Suggested measures to improve cyber security *76*

TERMS AND ABBREVIATIONS

Cryptography	The science of writing and reading of secret messages
Cyber	Related to computers or computer networks
Exploit	A method to take advantage of a vulnerability
Hardening	The act of disabling unnecessary functions from software or hardware in order to improve security
Malware	Malicious software, designed to damage a computer or a computer network
Maritime sector	Covers shipping, port, and maritime leisure industries
Phishing	The act of asking for personal or secret information for malicious purposes, usually involves impersonation
Port	The harbor area where vessels are docked
Process automation	Application of IT to eliminate human interaction from a process
Spoofing	Impersonation of another device or person in the context of computer networking
Terminal	An area inside a port where cargo is handled, a port may include multiple terminals
Vulnerability	A flaw in security

ACM	Access Control Mechanism
CHE	Container Handling Equipment
CS	Control System
DCS	Distributed Control System
DoS	Denial of Service
EIS	External Interface Service
IACS	Industrial Automation Control System
ICS	Industrial Control System
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IPS	Intrusion Prevention System
ISO	International Organization for Standardization
JMS	Java Message Service
JVM	Java Virtual Machine
MitM	Man in the Middle
OS	Operating System
PLC	Programmable Logic Controller
SCADA	Supervisory Control And Data Acquisition
TOS	Terminal Operating System
TLS	Terminal Logistics System
XML	Extensible Markup Language

1. INTRODUCTION

It is well known in the field of cyber security that the recent trend of increasing connectivity between Industrial Control Systems (ICS) and the Internet, or other networks, broadens the range of cyber threats these systems are exposed to. The following incidents illustrate how control systems on different industries are equally susceptible to cyber threats:

- In 2010, a computer worm known as Stuxnet was discovered. The worm was designed to access and alter the Programmable Logic Controllers (PLC) of the Natanz nuclear fuel enrichment plant in Iran, but it later infected other systems as well [1].
- In 2013, it was disclosed that the Port of Antwerp had been under a continuous cyber-physical attack for several years and contraband worth hundreds of millions of euros had been moved through the port repeatedly. To facilitate the attacks, information systems of the port authorities were compromised and information was stolen. [2]
- In 2014, a cyber attack on a German steel mill prevented the operators of a blast furnace from shutting the furnace down controllably [3]. The resulting damage to the machinery was described as massive.

As proven by the case of Antwerp, ports, and maritime industry in general, are potential targets for cyber attacks. Although there is plenty of information available about cyber security of ICS systems in general, there hasn't been much research done on how to apply this knowledge on systems specific to maritime industry. In an attempt to work towards filling this void, this thesis collects and combines information from various sources to offer a comprehensive coverage of different cyber threats in a maritime terminal environment.

Before applying any security strategies, policies or technologies, one must recognize the assets that require protection, and the threats that endanger the security of these assets. Therefore this thesis concentrates on cyber threats and vulnerabilities, not on strategy or policy establishment. Only when the threats, and the risks they impose, have been covered, can the required countermeasures be designed and implemented.

This thesis was conducted at the request of Kalmar Global, a worldwide provider of cargo handling solutions for ports, terminals and distribution centers. Therefore, maritime container terminal automation systems are used as a subject of study. Most of the

content can still be applied to other cargo handling systems and even to ICS systems in general.

The structure of this thesis is twofold: first, cyber threats in terminal automation systems are explored in general, and then a cyber threat analysis is conducted on a single component of the terminal automation system of Kalmar. The component chosen for analysis is the External Interface Service (EIS), which provides a dedicated interface for all external communication to and from the system. Cyber security of this component is vital when considering confidentiality, integrity and availability of the entire system.

The research problems considered in this thesis are as follows:

1. What kind of cyber threats exist in terminal automation systems?
2. What kind of cyber threats does the External Interface Service face?
    a. What can be achieved by attacking EIS?
    b. How can EIS be compromised?
    c. How can cyber security of EIS be improved?

To answer these questions, a literature review is conducted. The review includes literature from the fields of cyber security, process automation and industrial computer networking in order to cover the different views on cyber security. In addition, interviews are conducted with process automation professionals of Kalmar to gain insight on how process automation is applied on container handling systems and how different cyber attacks might affect the process. To discover cyber threats specific to EIS, attack trees are constructed. Different attacks are also evaluated to provide a risk assessment.

As is reflected by the research problems, the focus of this study is primarily on cyber threats and their characteristics. Due to the limited timeframe and resources of the study, cyber security measures and solutions are only briefly examined. As will be explained in Section 2.1, there are certain information related threats that are outside of the scope of cyber security, and therefore outside of the scope of this study as well. In some cases technical failures and software bugs can cause effects similar to cyber attacks, but in this study they are not considered unless they are exploited with malicious intentions.

The structure of this thesis is as follows: in Chapter 2, various terms and the container terminal environment are defined to provide a basic understanding of the context of this study. Then, in Chapter 3, cyber threats of terminal automation systems are explored, thus answering the first research problem. In Chapter 4, EIS is analyzed and the second research problem including its sub-questions is answered. Finally, conclusions of the thesis are presented in Chapter 5.

2. BACKGROUND

In this chapter relevant terminology is explained. It is assumed that the reader has basic understanding of Information Technology (IT) and computer networking. Extensive knowledge on cyber security or ICS systems, such as a terminal automation system, is not required.

2.1 Cyber threats

To understand what a cyber threat is, cyberspace needs to be defined first. Lujiif [4] defines it as a digital world of computers, computer networks and people, where information is created, transmitted, received, stored, processed and deleted. In cyberspace, cyber attacks take place. IEC standard 62443 [5] defines a cyber attack as a successful exploitation of vulnerabilities in software, hardware or firmware of Internet application components or IT components. Here, unsuccessful attempts are considered attacks as well, as they may still have undesired effects on the target system. In this thesis *cyber threat is defined as a potential cyber attack.* Therefore cyber security can be defined as the implementation of protection against cyber threats. In this thesis the word 'threat' generally refers to a cyber threat.

According to von Solms and van Niekerk [6] cyber security and information security are not interchangeable as terms. They note that while all security is about protecting certain assets from certain threats, the set of assets protected by information security is not the same as the set protected by cyber security. This is illustrated by Figure 1 on page 4. Generally, if an information asset is not stored or transmitted using ICT, it is not covered by cyber security, but it isn't susceptible to cyber attacks either.

For example notes written on a piece of paper or undocumented knowledge in an employee's mind are not protected by cyber security as this information is not stored or transmitted in the cyberspace. On the other hand, gaining control of a physical device over the Internet is not considered as a threat on information security if the device doesn't contain any valuable information assets. This applies even if it was possible to cause major financial or physical damage by attacking that particular device. What is considered as a valuable asset depends on the security policy of the asset owner. The focus of this thesis is on cyber security and specifically on cyber threats.

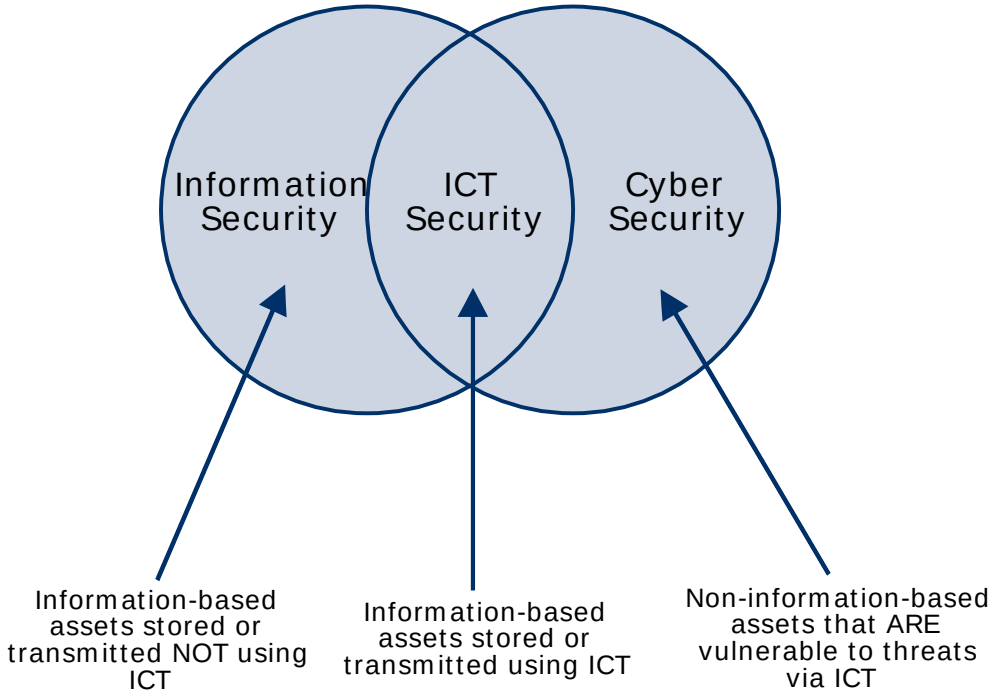

Figure 1. *The difference between information security and cyber security. Adapted from von Solms & van Niekerk [6].*

2.1.1 Vulnerabilities and exploits

In the context of security, a vulnerability is a security flaw that under certain circumstances causes behavior that was not intended. A method that makes use of a vulnerability is called an exploit. According to Dumitras [7] a typical vulnerability lifecycle begins with the release of a vulnerable application version (see Figure 2 on page 5). Once the vulnerability is found, an exploit is created and released "in the wild", for instance on a hacking community forum. Once attacks exploiting the vulnerability start emerging, the vulnerability becomes publicly disclosed, and soon a patch is released to remove the vulnerability.

The lifecycle ends, when all instances of the application have been patched. Depending on the type and distribution of the vulnerable application, the time between patch release and end of lifecycle can become very long. This means that even though the patch has already been released, all unpatched instances of the application still remain vulnerable to the initial exploit and other exploits using the same vulnerability [8].

When an exploit is released before the vulnerability becomes publicly disclosed, a term zero-day exploit is often used. This means that the vulnerability is exploited from the day zero of its known existence [8]. If the vulnerability is discovered by the application manufacturer or it is reported directly to the manufacturer instead of releasing it "in the wild", it can be patched before an exploit is created.

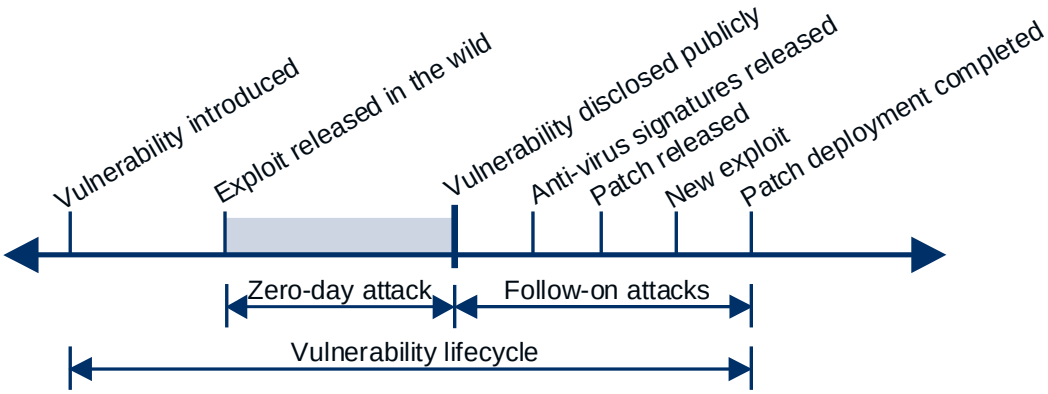

Figure 2. *Timeline of a vulnerability lifecycle. Adapted from Dumitras [7].*

2.1.2 Cyber attacks

A cyber attack can be divided in several phases [9-12]. Depending on the source used, the names and amount of phases vary slightly, but most of the models are based on the Cyber Kill Chain developed by Lockheed Martin. The Cyber Kill Chain consists of seven phases: reconnaissance, weaponization, delivery, exploitation, installation, command & control and actions on objectives [10]. To simplify the model, three main phases can be recognized: preparation, compromise and action (see Figure 3). All attacks don't go through all of the seven phases, but the three main phases are always present in a successful attack.

In the preparation phase an attacker finds a suitable target and acquires as much information about the target as is required (*reconnaissance*) [12]. Once the attacker finds a vulnerability, he can craft or obtain an attack tool to exploit the vulnerability (*weaponization*) [11]. The 'weapon' used can be a highly complex computer virus, a simple string of SQL commands or anything in between depending on the vulnerability found and the motivation of the attacker.

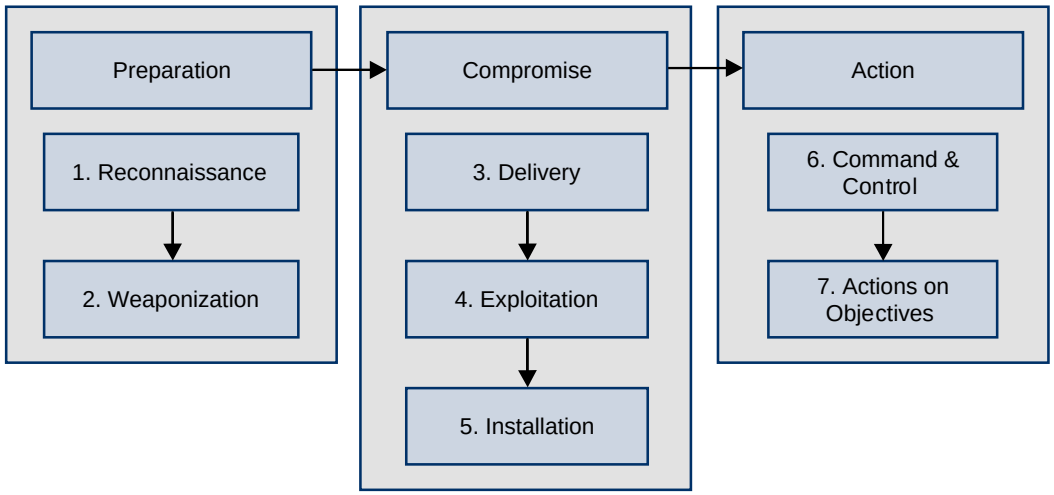

Figure 3. *The phases of a cyber attack. Based on Hutchins et al. [10].*

To compromise the target, the attacker first *delivers* the weapon to the target environment [10]. Delivery can happen through a website, in an e-mail, on a USB memory or in any other imaginable way. After a successful delivery the attacker *exploits* the vulnerability, usually to run a malicious program on the target system [10] or to gain unauthorized access of some form [9]. In *installation* phase the attacker installs a backdoor to provide easy access for later use [10].

Once the target system has been compromised, the attacker may want to gain control of the target system. To achieve this, a connection between the attacker and the target system is established *(command & control)* [12]. The connection provides a "hands on keyboard" access for the attacker [10]. At this point the attacker has gained access to whatever asset he was targeting, and depending on the intention the attacker can steal, destroy, modify or forge that asset to complete the attack (*actions on objectives*) [9].

According to a survey conducted by Kaspersky Lab [13] in 2014 the most common cyber threats faced by companies include spam e-mails, malware, phishing, network intrusion, theft of mobile devices and DoS (Denial-of-Service) attacks. According to another report by Kaspersky Lab [14] that focuses on ICS systems, 35% of malicious code in ICS networks propagates from the corporate office network, 29% from remote access connections and 9% directly from the Internet, while in rest of the incidents the ICS network is accessed directly. This means that roughly three out of four attacks utilize connections to other networks in the delivery phase.

2.1.3 Cyber attackers

There are many different motives for cyber attacks. Sometimes attackers are individuals willing to prove their abilities, or so called script-kiddies trying out pre-made attack tools [4]. However, a recent trend has been that attackers are criminal groups with financial profit in mind [15]. Criminal groups primarily target easily monetizable targets such as credit card or online banking information, but espionage, blackmailing and theft of physical goods are viable examples of other ways of making profit [15]. Sometimes attackers may also have a political agenda (e.g. activists or terrorists) [4].

Involvement of organized groups and money leads to more capable attackers and more efficient attack tools. Instead of attacking anything they can, organized attackers have fixed targets, so attacks also become more persistent. Even nations can use cyber attack tools as a part of their warfare actions, leading to the development of highly sophisticated and expensive attack methods [16]. This increase in resources needs to be considered when security measures are designed and implemented.

Business competitors may be interested in obtaining information for example on pricing, product development or contracting by the means of a cyber attack. Attacks can also come from the inside. The attacker can just as well be an employee as someone

from the outside. In fact, a survey by the Cyber Edge Group [17] shows, that one of five IT professionals in European and North-American companies are more worried about internal cyber threats than they are about external threats.

2.2 Cyber security in Industrial Control Systems

Industrial Control Systems (ICS) are systems that collect information from endpoint devices about the status of a production process and present it in an organized fashion [18]. According to Macaulay and Singer [18] the process controlled by an ICS can be manual, partly automated or fully automated. To avoid confusion, in this thesis systems controlling partly and fully automated processes are referred to as Industrial Automation Control Systems (IACS). IACS systems are considered as a subset of ICS systems.

Macaulay and Singer [18] divide ICS systems in three categories:

- Process control systems (PCS) allow system operators make control decisions, manipulating the course of the process.
- Distributed control systems (DCS) are PCS systems, where the controller elements are geographically wide-spread.
- Supervisory control and data acquisition (SCADA) systems are used to control large infrastructures as opposed to other ICS systems that control smaller elements.

The use of terminology is not established among the field and for example SCADA is often used interchangeably with PCS or ICS [18; 19], but the definitions given above are used throughout this thesis. To clarify differences between the terms, consider a drinking water distribution system. A SCADA system would be used to supervise the entire infrastructure to ensure the availability and quality of drinking water. A DCS would be used to control for example a set of valves spread over a section of the network and a PCS would be used to control a similar set of valves inside a purification plant. SCADA systems can be considered to be even more wide-spread than DCS systems [20; 21]. The terminal automation system considered in this thesis can be seen as a PCS, as it operates in a closed area, where access is controlled.

2.2.1 ICS networks as targets

Initially ICS networks were isolated and based on primitive serial technology that supported only minimal functionality [21; 22]. While these solutions provided little to no security at all, ICS networks were difficult targets for cyber attacks, as they were physically isolated from all other networks [23]. These days though, business operations require ICS networks to have more and more compatibility and connections to other networks, and to the Internet in particular [20; 23]. This connectivity exposes them to the

same threats other networks face, even though the devices and protocols in use may not have originally been designed with such threats in mind [22].

ICS systems have an average lifespan of 15 to 30 years [18; 23], which is considerably longer than the intended lifespan of most IT solutions. This means that ICS systems often have outdated IT technology in them, mostly because solutions covering the entire lifespan do not exist [24]. Outdated solutions naturally do not receive any security updates or technical support so they have no protection against the most recent threats.

The development towards Internet connectivity makes today's ICS networks rather intriguing targets for cyber attackers, especially when appropriate security measures haven't been deployed. This is also illustrated by the emergence of malware specifically designed to attack ICS networks. Examples of such malware are Stuxnet [1], Sasser [20] and Havex [25]. A recent trend in the context of malware has been targeted attacks [15; 26]. A targeted attack means that an attacker designs a piece of malware just to attack a specific system. As a result attacks are getting more creative and persistent, and therefore more difficult to defend against.

As a subset of ICS networks IACS networks face the same threats as all other ICS networks, but there are also threats that are specific to automation. These threats stem for example from limited processing resources, real-time systems and physical safety [20; 24]. To provide uninterrupted production, IACS systems need to be constantly up and running. Therefore they tend to be rarely updated and running on outdated software versions [24]. Common cyber security measures are still applicable in industrial automation networks, but they need to be adapted to take automation specific threats into account [20].

ICS networks, as all networks, are exposed to attacks where data is stolen or destroyed and to attacks where functionality is severed. As industrial networks, they are also exposed to attacks where production is interrupted. Specific to ICS networks are attacks where an attacker gains ability to somehow modify the process. Depending on the skills and objective of the attacker, changes made to the system may be random or calculated, resulting in controlled or uncontrolled reactions in the process. Here lays the main difference between regular IT systems and ICS systems: attacks on ICS systems can, and often will, cause physical consequences [18]. The added layer of physical safety makes ICS systems more complex to secure, and since mechanical failures and cyber attacks often have similar consequences, attack detection may be more difficult than in regular IT systems.

One of the few publicly disclosed cases where a cyber attack has been used specifically to cause physical damage was reported in 2014 by the Federal Office for Information Security [3] in Germany. Although the report doesn't go into detail about the progress of the attack, it was disclosed that the attackers used phishing attacks to gain access to

the corporate network and were able to propagate to the ICS network from there. Exploiting vulnerabilities in the control components, the attackers were able to drive the control system of a blast furnace to an “undefined state” [3]. As a result the operators weren’t able to shut the furnace down controllably, which, according to the report, lead to “massive damage to the machinery”. While there is very little information available about similar incidents, it is difficult to tell whether it is because such incidents are very rare, because they are difficult to distinguish from other failures or because companies do not report them in order to protect their reputation.

2.2.2 IACS cyber security standards and certificates

Standardizing cyber security in IACS networks is not straightforward, because there are multiple applicable standards and good practice guides made by various organizations. They are all used variably and incoherently over the field [24]. ISO/IEC 27000 standard family is one of the more used standard families in the IT sector in general. It includes multiple standards for information security management and best practices for risk management [24]. The standard family has been written in a fairly high level of abstraction to be applicable in the entire IT sector and it isn’t process automation specific, so some adaptation will be necessary. In addition there is common criteria ISO/IEC 15408, which provides criteria for IT security evaluation [27]. Products certified with ISO/IEC 15408 are generally considered secure [24].

There are also standards specific to industrial control systems. ISO/IEC 62443 [5], which includes the older ISA99 standard, is a security standard for industrial automation and control systems. Also the National Institute of Standards and Technology (NIST) has a series of Special Publications (SP) regarding different aspects of SCADA and ICS security [28]. Especially NIST SP 800-82 [29] titled *Guide to Industrial Control Systems Security* is worth mentioning. IEEE is also developing a new standard, named P1711, for a cryptographic protocol to be used for improving security of serial communications which are often used in ICS environments [30].

Some standards are field specific, but are still partly applicable in the context of IACS networks as a whole [24]. Examples of such standards are IEEE P1686, ISO/IEC 62351, ISO/IEC 61850 and AGA-12. No standards specifically made for IT of the maritime sector were found at the time of writing.

As certification of process automation system security is not the primary subject of this study, extensive analysis on different certificates was not conducted. It seems though, that at the moment there are no major or widely accepted certificates available for ICS systems. Various consultation companies seem to offer their own certificates, but these rarely comply with any specific standards and therefore their actual quality and value can be questioned.

Even though standardization and certification of an ICS system proved to be challenging, it is still possible to train ICS cyber security specialists. The European Union Network and Information Security Agency (ENISA) has recently explored these certification programs and identified three ICS-specific certificates: IEC 62443 Cyber Security Certificate Program (CSCP), GIAC Global Industrial Cyber Security Professional (GICSP) and Certified SCADA Security Architect (CSSA) [31]. While CSCP and GICSP are targeted towards ICS professionals, the CSSA certificate is targeted for IT professionals.

2.3 Automated container handling

In this thesis the automated container handling system of Kalmar is used as a case example. A container handling system provides a solution to moving and storing containers in the container yard of a terminal. In this case, a maritime terminal is examined, but same basics apply to inland terminals as well.

The system acts between Ship-To-Shore (STS) cranes and container delivering trucks or trains as illustrated by Figure 4. The implementation of container handling varies depending on system provider, physical environment, equipment used and contracts between terminal operator and system provider. The case described in this study presents only one of the possible solutions.

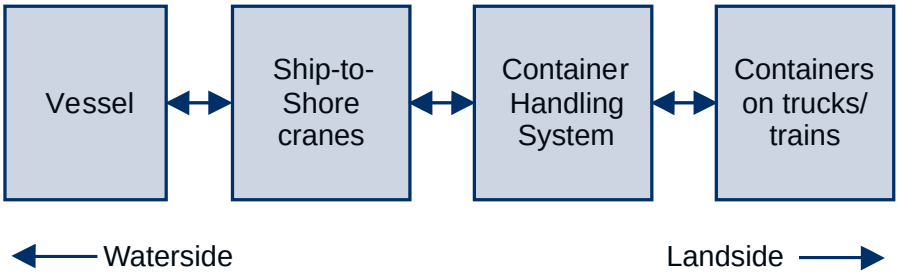

Figure 4. *Location of a container handling system in the logistic chain of a port.*

Container handling can be automated from two different perspectives: the equipment perspective and the process perspective. Automation from the equipment perspective means that there is no need for human operators in the handling equipment. The process perspective means automating the process of deciding which containers need to be moved and when in order to optimize loading and unloading of cargo. This thesis examines a system that has been automated from both perspectives.

The automation system consists of four core layers: Terminal Operating System (TOS), Terminal Logistics System (TLS), Control System (CS), and Container Handling Equipment (CHE). Figure 5 on page 11 presents the hierarchy and communication flows of the system.

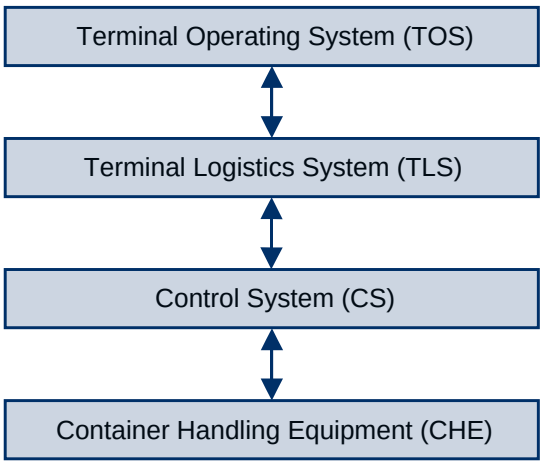

Figure 5. *Hierarchy of the four core layers in the automated container handling system of the case example.*

When a container arrives in from waterside or landside of the yard, information about the container and its location is delivered to the Terminal Operating System. TOS then requests Terminal Logistics System to store the container and TLS calculates where the container should be stacked. In addition, TLS finds the optimal Container Handling Equipment to carry out the move based on type and availability of CHEs. TLS then creates a job order for Control System, which converts it into simple instructions that the CHE can understand. CS instructs the CHE to complete the job and the container gets moved to a stack. The procedure for moving containers from stacks to waterside and landside loading areas is basically the same.

The physical environment in this case is the terminal area, including the container yard, where containers are stored in stacks and handled by CHEs. The TLS and CS software are running on a dedicated server that is physically located in a building somewhere in the terminal area. The server is usually mirrored and placed on two different physical locations in the yard to add redundancy. The TOS software is provided by a third party so it is located on a separate server in the terminal operator's network. The system has multiple connections to the Internet: TOS and TLS are connected to the Internet in order to extend their functionality and offer remote support, while some parts of the system below TLS are also remotely accessible for maintenance purposes.

CHEs are connected to the CS via wireless or wired connections, depending on their type. CHEs are either rubber-tyred or rail-mounted. Rubber-tyred equipment mostly requires wireless connections due to their high mobility compared to the rail-mounted equipment. The largest cranes require the highest data rate and therefore they often utilize wired connections to communicate with the CS. All other connections in the system are typically wired.

In terms of criminal activity ports and their terminals are desirable targets, as they act as hubs of maritime transport. As terminal and port operations become more and more

automated and the number and importance of information systems increases, cyber attacks are starting to emerge as a part of other criminal actions. A high profile case example occurred in the port of Antwerp in Belgium, where cyber attacks were used to facilitate drug smuggling. Drugs were loaded in containers with legitimate cargo, such as bananas or timber [2]. To steal information related to these containers as they arrived in the port, an organized crime group hired skilled hackers [2]. The hackers were able to penetrate the port authorities' network and provided the criminals with all the information required to forge documentation and pick the containers up before their legitimate owners [2; 32].

The port authorities eventually discovered the initial breach and a firewall was installed, but the hackers were able to physically break in to the premises to continue their activity [2]. They installed wireless devices to record keystrokes and screen captures on PCs, effectively bypassing the firewall [2; 32]. The attack is believed to have been ongoing for two years, but multiple raids by Belgian and Dutch police forces ended the activity in 2013 [2]. The total value of drugs trafficked through the port has remained unknown, but the street value of contraband seized in the raids alone totaled in around £260m (around EUR 307m at the time) [2].

As the example illustrates, vulnerabilities in cyber security can have very serious and concrete consequences. As the content of containers is what criminals would most probably be interested in, any information about containers is likely to be stolen. Another likely scenario in a terminal environment would be gaining control of one or more CHEs to either move containers or cause collisions and physical damage. A third likely scenario would be a situation where the attacker is somehow able to prevent the system from operating, resulting in financial losses.

2.4 Related work

The basics of ICS cyber security have been covered in various books, such as *Robust Control System Networks: How to Achieve Reliable Control After Stuxnet* [33] and *Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS* [18]. Several national authorities and other knowledgeable entities have produced books, whitepapers and good-practice guides to provide plenty of information about ICS cyber security as well. Centre for the Protection of National Infrastructure (CPNI) of UK has produced an ICS security good-practice guide and other cyber security material [34]. The U.S. Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) has a repository of ICS-related whitepapers, advisories and reviews [35]. In Finland the focus has been on cyber security of IACS systems [20; 24].

ENISA published a report [36] in 2011, studying cyber security aspects of the maritime sector in EU. The report states that the European economy is critically dependent on the maritime sector while the sector itself is increasingly dependent on ICT. According to

the report, the current security posture of the sector is weak: awareness, technical solutions, security policies and governance all call for improvement. In addition the report suggests more collaboration among the sector to create unified policies and good-practice guides, and to improve information exchange.

In 2014 U.S. Government Accountability Office (GAO) released a similar report [16], but focusing on the U.S. port facilities. The findings are parallel to the ones presented in the ENISA report: cyber attacks on maritime sector could have significant impacts to commerce; the overall security posture is poor; cyber security information is not shared and the current regulations do not adequately address cyber security. The report recommends that cyber threats are assessed, security guidance is created and a coordinating council is established. The report also briefly lists different threats based on both the source of the threat and the type of exploit.

A report by U.S. Coast Guard Commander Joseph Kramek [37] explores the current state of cyber security in U.S. port facilities. Again, the findings are very similar to the other reports: low security awareness, inadequate policies and technical solutions, and lack of standardization. Of the six ports studied in the report, only one had conducted a cyber security assessment and none had a cyber incident response plan.

Maritime cyber security company CyberKeel has produced a whitepaper [38] that discusses the motivations for attacking the maritime sector and cyber defenses to defend against attacks. The motivations according to the paper are stealing money or information, moving cargo and causing disruption or loss. When discussing cyber defenses, in addition to the technical solutions, the paper stresses the importance of “the human factor” and improvement in awareness about different cyber threats.

3. THREAT LANDSCAPE

To discover cyber threats in terminal automation systems, a literature review was conducted. Literature from the fields of cyber security, industrial networking and process automation was included to cover different aspects of the subject and to recognize qualities specific to the process automation environment. Based on the review, numerous possible attack vectors were recognized and they have been classified in 10 categories:

1. Malware
2. Denial of Service
3. Spoofing
4. Unauthorized access
5. Software vulnerabilities
6. Hardware vulnerabilities
7. Networking vulnerabilities
8. Misuse of process automation data
9. Data breach
10. Users.

Due to the constantly evolving nature of the threat landscape and the vast amount of threats, it is not possible to create an exhaustive list of all existing attacks or vulnerabilities. All threats recognized were found to fit in the 10 categories above, but the attacks described in the following sections are examples only, and their purpose is to illustrate the nature of that specific category. The categories were designed to cover all threats recognized during this study, but as new threats are discovered, the landscape changes and new categories may be required. To be able to defend against these emerging threats, it is important to reassess the threat landscape regularly.

3.1 Malware

Malicious software, or malware, is a piece of software that is specifically designed to damage or compromise a target system [20; 39]. Malware can be divided in three categories: viruses, worms, and Trojan horses. All malware consist of two main components: payload and propagation component. Between the three categories of malware, the payload can be very similar but the propagation component is what defines the type of a single piece of malware. [39] The effects of malware depend on the payload, but since they can do anything legitimate software can, possibilities are virtually endless. For example a Trojan known as Karagany is able to upload stolen data, download new

files, run executable files, collect passwords, and capture screenshots [40]. The source code of Karagany was leaked to public, so it is available for anyone to exploit [40].

3.1.1 Viruses and worms

Computer viruses, similarly to biological viruses, can't live on their own, but need a host instead. A computer virus is a piece of code that attaches to another file in the system. Once this host file gets executed, the malicious code gets executed as well. Once the malware has found a host file, it is also able to replicate itself in order to propagate to other systems. [39] The propagation usually happens when an infected file is downloaded or via some portable device, such as a USB-drive or a mobile phone that has been connected to the infected system.

Worms are pieces of malware specifically designed to propagate in computer networks. Unlike viruses, worms are independent of other files. Once a worm infects a system, it replicates itself and tries to send these replicas to other systems. The method of propagation makes them capable of infecting numerous systems in a short space of time without any interaction with a human user. [39] Due to the fast spreading, worms are effective in creating large networks of compromised systems. These networks can then be used to carry out more complicated attacks.

3.1.2 Trojan horses

Trojan horses enter a system inside a legitimate software (hence the name). They can either be coded directly into the source code of the software by the producer, or an attacker can embed malicious code to software made by someone else. [20; 39] A viable example of the latter would be taking a web browser, crafting a malicious add-on for it and then distributing this browser on your web page with the add-on pre-installed. The add-on could for example show ads or try to pick up any passwords or credit card information. Trojan horses are particularly difficult in terms of malware protection, because detecting malicious code in otherwise legitimate software is more complicated than detecting viruses or worms [20].

One significant form of Trojans is a Remote Access Trojan (RAT). A RAT gives the attacker complete control over the target system. Once a RAT has infected its target, it opens a remote connection between the target and the attacker. Opening the connection from the target's end allows an attacker to access targets even if they are behind a firewall and therefore invisible to the attacker. [41]

3.1.3 Watering holes

A watering hole attack begins by infecting a web site or other resource that is often accessed from inside the target network [42]. When someone from the target network ac-

cesses the infected resource, the malware is able to infect the target and propagate into the target network. While a watering hole isn't actually a type of malware, but rather a propagation method for malware, it is presented here because it illustrates a different approach on cyber attacks. Instead of going to the target, the attacker hides and waits for the target to come to him, which also explains the name. Watering hole attacks also illustrate the importance of collaboration between organizations in order to achieve a higher level of cyber security. In other words, the weakest link in the chain of security can sometimes be outside of the organization. Watering hole attacks are a relatively new phenomenon as they were first discovered in 2012 [42].

3.1.4 Implications for ICS networks

For all three types of malware (viruses, worms and Trojans) ICS systems, including terminal automation systems, are easy targets. Since these systems often are out of date in terms of software and security updates, there are also more possible entry points for malware than in a system that is up to date. Propagation without human interaction makes worms especially threatening for ICS networks [20]. Malware attacks can be either general, targeting any system they can reach, or targeted. While targeted attacks are high in efficiency but low in probability, general attacks are far more common. As general malware typically looks for bank account information, shows advertisements or causes some other minor harms, they may not be particularly harmful in ICS environments. On the other hand general malware can waste computing resources, cause unnecessary reboots or other unwanted side-effects that aren't even intended functionalities of the malware.

As was already mentioned, different applications of malware are endless and therefore actions should be taken to prevent any and all infections, be they general or targeted. It is also worth noting that although the Internet is the main source of malware, isolating a network entirely doesn't deny the threat. Infections can still enter the network through any portable devices with memory on them or through any new software installed on the system. Therefore it is important to develop and implement policies on how portable devices are handled, what software is installed on the system and who is allowed to install software. To further mitigate the risk, anti-virus software should always be activated, updated and properly configured.

3.2 Denial of Service

In a Denial-of-Service (DoS) attack the objective of the attacker is to prevent a system from serving its clients [20; 43]. The attack may be either intentional or unintentional [20], meaning that a DoS can also happen for example as a side effect of maintenance activities or when installing new devices or software that are improperly configured.

There are a few different ways to intentionally deny a service, for example *flooding*, *nuking* or *jamming*.

3.2.1 Flooding and nuking

To flood a system the attacker needs to send large amounts of messages, for example TCP packets, to the target [44]. When the target system receives more packets than it can handle, it will either stop working or slow down significantly [44]. In both cases the effect is noticeable for all users of the system. Successful flooding attacks originating from a single source require a significant amount of resources, especially if the target has been designed for commercial or industrial use. To have more capacity, an attacker needs multiple sources to attack from simultaneously. This kind of multi-source DoS attack is called a Distributed DoS (DDoS) attack. Typically DDoS attacks are carried out through *botnets*, i.e. networks of compromised devices [20]. In this case the attacker commands the botnet to send certain kind of packets to a certain destination.

From the target system's point of view a DDoS attack is observed as a significant increase in traffic from seemingly random sources. Since the sources seem to be random, it is difficult to filter the traffic generated by the attacker while allowing legitimate traffic to pass. A DDoS attack is analogous to a situation in a cellular mobile network where all the channels in a cell are already reserved and no new calls can be made. Similarly, if the target system is too busy handling all the requests from the attacker's botnet, connections from legitimate clients are denied or at least delayed.

One example of a DDoS attack is Blaster, which took down Microsoft's home page in 2003. In this attack a worm, later nicknamed Blaster, was created to compromise a large amount of Windows XP and Windows 2000 machines. Infected machines formed a botnet, which was instructed to collectively send messages to microsoft.com, successfully flooding the website. The attack kept the website down for two hours. [45]

Another form of DoS, called nuking, is an attack where the objective is to crash the target system so that it stops functioning completely [43]. While flooding attacks usually use well-formed, legitimate packets and rely purely on quantity of packets, nuke attacks use malformed packets to make use of a bug or vulnerability in the target system [43]. The CPNI technical note [43] only considers sending malformed packets as nuking, but it is worth noting, that a similar DoS situation can be achieved through malware as well. For example a virus, that repeatedly reboots the infected system, would have a similar effect.

3.2.2 Jamming

Signal jamming or radio jamming generally refers to deliberately interfering with a radio signal. A jammer transmits at the same frequency as the target and with enough

power the jammer is able to override the signal of the target receiver [46]. In essence, the receiver can't hear what the transmitter is saying, because the jammer is shouting over the conversation. For both the transmitter and the receiver it seems as if the connection has been lost.

More sophisticated jammers make use of the fact that certain wireless protocols listen to the transmission channel and only transmit when the channel is available. If a jammer reserves all channels, legitimate devices will have to wait for a channel to clear up. All wireless communication devices, such as Wi-Fi, cell phones and GPS receivers are susceptible to jamming. Jamming is a relatively simple as an attack vector, since jammers can be bought online with prices starting from less than a hundred euros and more powerful units selling for a couple of hundred euros.

3.2.3 Implications for ICS networks

In the context of ICS networks, an obvious motivation for a DoS attack would be blackmailing [20]. An attacker can try and demand a large amount of money or other profit to end the attack. In the worst case scenario a DoS attack would interrupt the entire production, which in turn might make it tempting to pay the attacker. This is not recommended though, as there is no reason for the attacker to actually stop the attack instead of continuing and demanding for more.

DoS attacks may also be used in conjunction with other attacks. For example nuking any security or monitoring systems would allow for easier or undetected access to other parts of the network. In a container handling environment a DoS attack would most likely stop one or more CHEs from operating and this way harm production.

Since DDoS attackers have virtually unlimited resources at their disposal, complete protection against them is not possible. Therefore it is important to be prepared and have predefined actions and roles in case of a (D)DoS attack. Network segmentation and load balancing are other ways to cope with such attacks. [20] It is worth noting that DDoS attacks are also harmful in a totally different way. If the system gets infected and becomes a part of a botnet, it'll start attacking other networks as commanded by the botnet manager [18]. This causes unnecessary use of network resources and might eventually even lead to legal actions.

Maritime ports are busy environments by nature, so most often used frequencies for wireless communication can easily become congested or even completely jammed. A cruise ship full of people arriving at the nearby harbor or the new Wi-Fi controlled system of the neighboring terminal can cause notable decrease in signal quality and channel availability. Therefore wireless systems should at least be capable of functioning on multiple frequencies, but a backup system using an entirely different technology would be ideal.

3.3 Spoofing

Spoofing in general refers to an attack, where the attacker impersonates another device or user. The aim of a spoofing attack usually is to deny access to the target system from other users, to steal information or to bypass security systems. [47] For example it is possible to capture network traffic of other users by setting up a rogue base station in a wireless network, which is known as base station spoofing. If this base station is configured to route traffic to legitimate base stations, the network remains functional and the attack might go unnoticed for a long time.

3.3.1 Packet spoofing

It is also possible to spoof network packets. Packet spoofing means sending forged packets into the network. For example sending large amounts of IP (Internet Protocol) packets with a spoofed source address to a single destination would cause loss of performance at the receiving end. The receiver would also try to reply to all these spoofed packets, which would mean that the device, whose address the attacker set as the sender address in order to protect his own identity, would also suffer from the attack. If the resources of the network are limited, this storm of packets would possibly consume most of the resources, harming other users as well. Results are similar to Denial-of-Service attacks and IP spoofing is indeed often used as a DoS attack method [47].

Another protocol prone to spoofing attacks is ARP (Address Resolution Protocol), which converts IP addresses to MAC (Medium Access Control) addresses and vice versa [48]. ARP is used to link a physical device (identified by MAC) to a certain IP address. By spoofing ARP messages, an attacker is able to alter the linking between IP and MAC addresses, which affects the routing of packets within the network [48]. This way the attacker can reroute traffic to go through his computer and capture the traffic for his own purposes. Spoofing ARP packets with random addresses would result in a DoS situation, since packets wouldn't be able to reach their intended destinations. It is possible to defend against spoofing attacks by filtering spoofed packets based on conflicting addresses in a packet [47]. A conflict means that for example the source address is inside the network, but the packet is actually coming from the outside. This of course won't work if the attacker is already inside the network.

3.3.2 Implications for ICS networks

Considering ICS networks, and IACS networks in particular, there are two types of spoofing attacks worth noting: control message spoofing and sensor spoofing. Control message spoofing means feeding forged process control messages to the network. This would lead to either controlled or uncontrolled changes in the process, which in terms of container handling would equal to the ability to move containers or cause CHEs to collide or even fall over. Sensor spoofing means sending forged sensor data to the devices

or software that use it. Sensor spoofing would allow the attacker to hide malicious actions by sending normal-looking data while another attack is in process or trip safety mechanisms by spoofing sensor data that is outside of the safe range.

These two attacks could be combined to move a container in a controlled way without it being visible to a supervisor. In this attack control messages are spoofed to instruct a CHE to move a desired container to a desired location while GPS data coming from the CHE is interrupted and replaced with data recorded earlier to mimic normal functionality. Data from other sensors can be also spoofed to avoid any logical conflicts that might trip any safety or security functions.

3.4 Unauthorized access

In terms of safety and physical security, access is the ability to enter a certain space or open a certain door. In the cyberspace access means the ability to view or modify data and execute programs. Access is considered unauthorized when someone gains access to something he shouldn't. To control access, an Access Control Mechanism (ACM) is required. To access a resource, a user requests access from an ACM that makes the decision on whether access is granted or denied [49]. The information needed to make the decision depends on the mechanism used. For example, if the mechanism is password protection, the decision is made based on the password a user has provided. If the mechanism is fingerprint recognition, the decision is based on the fingerprint of the user. Mechanisms can also be combined to form a multi-factor authentication method. A system can require both a password and a fingerprint or either one of them. When considering a process automation environment, both digital and physical access control measures are required in order to keep the system secure.

3.4.1 Passwords

One of the simplest and most common methods for digital access control is a password. They are widely used, because the concept is easy for users to understand and simple to implement. The problem is that passwords should either be easily memorable or written down somewhere [39]. A password that is easy to memorize, is susceptible to various password guessing attacks, such as brute force attacks and wordlist attacks [39]. In a brute force attack all possible character combinations are tried one by one. This obviously takes time, but is easy to execute, especially since pre-made software is easily accessible. Wordlist attacks are only slightly more intelligent. The attacker defines a list of words to be tried as the password. The wordlist can for example contain most common passwords or passwords stolen from another source. Again, pre-made software is available for anyone on the Internet. Guessing attacks can be slowed down drastically by limiting the amount of consecutive login attempts from a single source [39].

If passwords are strong enough to persist guessing attacks, they probably are difficult to remember as well, so the users are likely to write them down on small pieces of paper or store them in text files on their computers. When a piece of paper or a device with the text file in it gets lost or stolen, the risk of unauthorized access increases. The likelihood of this happening may not seem high when considering one person, but in a company with hundreds of employees the risk becomes notable. To mitigate the risk, a password policy should be created and employees should be trained to comply with that policy.

3.4.2 Authentication systems

A password can also be obtained directly from the authentication system. The username-password combinations are generally stored as hashes in the system [39]. Hashes are essentially encrypted and compressed versions of the combinations. This means that every time a user tries to log in, the hash is calculated based on the username and password, and the hashes are then compared. If an attacker is able to steal the hashes from the authentication system, he might try to reverse the hash algorithm to gain the username and password or try to come up with a combination that results in a hash that matches any of the stolen hashes. Hash algorithms are generally designed with these attacks in mind, but as computing power of processors increases, older algorithms become inadequate. When choosing access control solutions, the predicted lifetime of the system should be compared with the expected lifetime of the hashing algorithm to ensure that access control remains secure throughout the lifetime of the system.

3.4.3 Privilege escalation and lateral movement

As modern information systems and system networks often have multiple subsystems with different accounts and different user groups, obtaining a single username-password combination may not be of much use to an attacker. For example accounts in a system may be different for general users than they are for administrators. Obviously the number of administrator accounts is smaller than the number of general accounts, so for an attacker it would be easier to steal a general user's account. Compared to an admin, a general user has a more restricted ability to perform actions, so it would be beneficial to turn a general account into an admin account. This action is called privilege escalation [50]. Legitimate users can use escalation to temporarily elevate their user rights in order to perform an action [51]. Unauthorized privilege escalation usually happens through a software vulnerability.

While privilege escalation can be described as vertical movement, moving between different parts of a system or network is referred to as lateral movement. As some systems or databases may be accessible only from certain subnets, an attacker needs to be able to move laterally in the network after the initial breach [15]. An example of lateral move-

ment would be gaining access to the office network from a visitor Wi-Fi network. From office network it would then be possible to move further to process network and so on.

3.4.4 Physical access

In process automation environments physical access control is just as important as its digital counterpart, because attacks tend to become easier once you have a physical access to your target [8]. Physical access control could be viewed as a part of physical safety, but cyber security needs to be considered as well [8]. For example installing a water-spraying sprinkler system in the server room causes an electric hazard, which poses a threat to both physical safety and cyber security, but leaving the server room unlocked only poses a threat to cyber security.

There is a multitude of ways to apply access control, so they are not covered here, but the general policy of granting access for only those who need it is recommended. This includes revoking unnecessary access rights systematically. Before deciding on access control methods, it is important to recognize all devices, systems and data that require securing.

3.4.5 Remote access

Remote access is the ability to access organization's non-public resources from networks other than the one of the organization [51; 52]. To facilitate remote access, access points need to be opened for connections from other networks, which exposes the system to attacks coming from those networks. To control who is able to create a remote connection, an access control mechanism is required. This way an attacker can't directly connect to the target network. The attacker can still try phishing (covered later in Subsection 3.9.4) or password guessing attacks to gain the required credentials to establish a connection.

Another approach to attacking through a remote access point would be spoofing or Man-in-the-Middle attacks where the attacker steals data from remote connections of legitimate users somewhere in between the endpoints [52]. To tackle this threat, remote connections should always be encrypted. In a typical ICS network a strong authentication method, such as RSA SecurID card, is used for reliable authentication and a Virtual Private Network (VPN) for secure communication [20].

A third possibility would be attacking the other end of the remote connection, i.e. the remote user's device. This could happen for example by malware infection or physical theft [52]. In this context, the remote user's web browser is particularly vulnerable. There is a variety of different attacks starting with the user entering a malicious web site and resulting either in leaked login credentials or in data breach. Examples of such attacks are cross-site scripting (XSS), cross-site request forgery (CSRF) and cookie poi-

soning attacks. While anti-virus software may be able to protect against attacks through web browsers, it is important to train the users to be aware of such threats.

3.5 Software vulnerabilities

Software vulnerabilities are faults in the software (in design or in implementation) that allow an attacker to somehow modify the target system or shut it down completely [50]. Modifying might include executing code, moving files to or from the system, removing data, or spying on other users in the system. Basically software vulnerabilities act as enablers to cyber attacks. According to Schneier [39], faults in implementation of both software and hardware are far more common than faults in design. Finding faults (also called bugs) from software code is difficult and the idea of bug-free software is impossible if the size of the software is anything more than trivial. While applying good programming practices and testing software carefully is extremely important, mitigating software related risks can't rely solely on finding every single bug before releasing the software, as bug-free software doesn't really exist.

Since some vulnerabilities are almost certainly found after the software has been released, updates need to be made. In a regular office network this is not a problem, but in in a process automation network it would in most cases require interrupting the process, which obviously isn't desired, or in some cases even possible. Making changes to the system running the process may also cause unexpected changes in the process. Clearly, updating requires planning and comprehensive testing in a simulated environment to cause as little delay as possible. From the cyber security viewpoint updates should be made as often as possible.

3.5.1 Different types of software

In a process automation network there are three main types of software that need to be considered: operating systems, custom-made applications and Commercial Off-The-Shelf (COTS) applications. An operating system acts as a platform between the hardware and the software, and decides how computing resources are divided between software processes. An operating system is a complex system that is designed to run on a variety of different hardware, which makes it prone to unexpected behavior under certain conditions. Being a critical part of the system infrastructure and prone to vulnerabilities, operating systems are enticing targets for attackers.

Not all operating systems are the same though. Basically two types of operating systems are used: Windows or UNIX-based. While Windows systems are widely used and constantly updated, they also are the main target of cyber attacks. UNIX systems on the other hand are less consistently updated but have a smaller malware landscape. Being less used and tested, UNIX systems supposedly have more elusive vulnerabilities that, once found, may be used in zero-day attacks.

Custom-made applications are specifically made for a single purpose and can rarely be reused in other instances. The problem of custom made applications is that they typically aren't as extensively tested as COTS applications that may already be tried and tested by millions of other users. COTS applications, on the other hand, have a larger user base, and therefore they are targeted by a larger crowd of attackers as well. This means that even though there might be less vulnerabilities to find, there also are more people looking for them. For the same reason COTS applications are more susceptible to malware attacks, whereas custom-made applications usually are only susceptible to malware attacks that are specifically targeted to that particular application.

3.5.2 Buffer overflows

A buffer overflow is one of the most common software vulnerabilities [39]. It is easy to find and easy to exploit, and therefore it was ranked third on the list of most dangerous software errors [53]. *A buffer* is a predefined place in the computer's memory that is reserved for a user input [54]. The buffer has a limited size that has been set by the programmer. A buffer overflow occurs when the size of the input is larger than the buffer [8]. Whatever happens to be next to the buffer in the memory, will be overwritten by the excess of the input, in other words, the buffer is overflowed by the input [54].

For example if the buffer is reserved for user's first name, the programmer may assume, that no first name is longer than 20 characters, so he reserves a buffer of 40 characters just to be safe. If a user of the program, on purpose or by accident, enters an input of 41 characters, the last character won't fit in the buffer and will overflow to an area on the memory that shouldn't be overwritten at this point. The result of a buffer overflow depends on the data that has been overwritten [54]. If any of the program code has been overwritten, the program will crash or become unstable as soon as it tries to execute the part of the program that has been overwritten.

Accidental buffer overflows usually cause programs to crash, but cyber attackers can use buffer overflows in a more controlled way as well. If an attacker is able to find a user input that will potentially cause an overflow, he is able to inject arbitrary code into the memory of the target computer by entering the code as the input [8; 54]. As soon as the target computer attempts to execute whatever was on the memory before the overflow, the code of the attacker gets executed [8]. The consequences depend on the input code, but a knowledgeable attacker is able to even gain full control of the target [55].

To avoid buffer overflow vulnerabilities, all user input lengths should always be checked by the program before writing them into the memory. It is a simple procedure that programmers often overlook or forget during the programming phase [55]. There are also solutions that monitor the *buffer stack* (memory reserved for buffers) and stop the victim program in case of an overflow [8; 55]. This way they prevent execution of malicious code, but termination of the program probably has other undesired conse-

quences as well. It is also possible to use filters that check all inputs for any susceptible content before they are handed over to the program [55]. These kinds of intermediary software have their own flaws and writing secure code in the first place is the best way to defend against buffer overflows.

3.5.3 Reverse engineering

Reverse engineering is analysis of a system to identify its components and their relationships [56]. While the term originates from hardware analysis [56], it is here considered as a method for reverting software code from a machine-readable format back to a human-readable format. The motivation for reverse engineering software is to find out what the program does and how does it do it. This information is useful for example when creating a discrete malware that functions as the legitimate software, but performs malicious actions at the same time. Such malware presumably stays undetected longer as the target seems to be operating somewhat normally.

Reverse engineering is tedious work and while it used to be mainly manual, nowadays there are decompilers and other software that are able to do some of the work. Requiring a lot of work, reverse engineering isn't something that most casual attackers would be interested in, but for example competitors from the same industry might be willing to hire more motivated hackers or software specialists to uncover your software solutions.

3.5.4 Implications for ICS networks

To mitigate the risk of having vulnerable software in an ICS system, it is important to install only software that is necessary for the operation. Any pre-installed software that is not needed, should be removed, web browsers and e-mail clients are especially notorious for causing security issues. These measures are also steps in the process of hardening the operating system. In addition, all software should be updated as often as possible to patch any recently found vulnerabilities. The risk can be further mitigated by evaluating cyber security when sourcing software used in the system and including secure programming practices when producing software in-house.

To prevent programs from interfering with each other and especially to prevent malware from interfering with legitimate software, it is possible to separate different programs into their own execution environments. This method is known as sandboxing. When a program is running in a sandbox, it can only access resources that have been explicitly allowed, while use of all other resources is prohibited [57]. This means that even a malicious program won't be able to exploit memory or files that belong to other programs.

Software in ICS systems and especially in IACS systems can be very complicated and a result of a long development process, so it makes sense to protect the software from being copied by competitors. To make reverse engineering more difficult, software code

can be *obfuscated.* Obfuscation can be described as a method that transforms the code into a form where it is difficult to extract information from, while maintaining the functionality of the software. The trade-off of obfuscation is increase in the code length, which usually affects negatively on execution times. Therefore, obfuscation methods and their impacts need to be tested before application.

3.6 Hardware vulnerabilities

In this thesis hardware vulnerabilities are perceived as vulnerabilities that are related to a certain device, even when the actual fault is in the firmware running on the hardware. Firmware is a piece of software specifically designed to control the operation of the device and connections to other devices. In process automation networks most commonly used types of hardware are Programmable Logic Controller (PLC) units and networking devices, such as routers and wireless base stations, which will be covered in the next section.

3.6.1 Programmable Logic Controllers

PLC units are typically used to control processes or convert and deliver sensor data [20]. Being programmable, PLCs have a firmware that is responsible for providing the intended functionality. The firmware can usually be updated, so functionality of the PLC unit can be improved without physically replacing it. For an attacker who wants to somehow modify the process, PLCs are plausible targets, especially because even today many PLCs are insecure by design [23; 58]. This means that security wasn't considered in design or there are exploitable faults in the design that can't be corrected with firmware updates.

As PLCs are coming down in price, there is a trend emerging where attackers can buy PLCs they want to compromise, take them home to inspect and experiment with, and then design an attack based on a vulnerability they have found. This means, that attackers have virtually unlimited amount of time to find vulnerabilities as opposed to a situation where the attackers would first break in to a system and then start to look for any vulnerabilities. While in IACS networks attacks to PLC units are most probable, vulnerabilities can also be found in any other hardware with firmware in them.

3.6.2 Tampering

In the context of this thesis hardware tampering means physical modifications to alter functionality. The term is usually used when modifications are made with malicious intentions. One example of hardware tampering is a hardware Trojan. A device with a hardware Trojan on it acts just as a normal device would, but under certain circumstances malicious actions are performed [59]. The Trojan is either planted on the hard-

ware somewhere between design and implementation phases or the entire device can be counterfeited and sold as the legitimate product. It is possible to test hardware for Trojans, but as the physical size of components decreases and at the same time the complexity increases, comprehensive testing becomes more and more difficult and expensive [59]. Similar faulty behavior can also occur due to unintentional mistakes in the design and these vulnerabilities can be exploited as well, so careful testing of all hardware used is important not only for safety and reliability reasons, but for security reasons as well.

3.6.3 Implications for ICS networks

There are no direct ways to completely deny hardware attacks, but much like with software, to mitigate the risk it is advised to consider the cyber security perspective when choosing devices to be used. In addition secure network design mitigates the risk of attacks targeting PLC firmware coming from the outside. Attacks coming from inside the network are only slightly mitigated and attacks through physical access still remain, so PLC security can't be ignored even when other security measures are in place.

3.7 Networking vulnerabilities

Vulnerabilities related to networking are essentially vulnerabilities in software and hardware of networking devices, but since network security is a major part of securing ICS systems, it is covered in this separate section. The sole purpose of a network is to provide the possibility to send data packets between devices, so network-related attacks target the data sent over the network. The data can either be stolen, modified or discarded anywhere between the endpoints, i.e. the sender and the receiver(s).

3.7.1 Routers and switches

The purpose of routers and switches is to route packets towards their destination. This means that attacks on these devices will directly affect the way packets are routed inside the network and between other networks. As has already been discussed, they are vulnerable to flooding and packet spoofing attacks. Modern routers and switches are programmable and have updatable firmware in them, so, much like with PLCs, attackers may be able to install a maliciously modified firmware version [24]. A malicious firmware might for example allow an attacker to reroute or modify any packets that happen to go through the device.

Attacks mentioned above can be carried out remotely over the network, even from the Internet and can be mitigated by careful network design. Since routers and switches are physical devices, they can also be physically attacked. If an attacker can gain physical access to a device, the possibilities are virtually endless. To prevent any traffic passing

through the device, the attacker can unplug cables, switch the device off or physically damage it. To capture or modify data, the attacker can attach a separate device in between the cable and the switch or router. This is called a Man-in-the-Middle (MitM) attack, where the endpoints don't know that there is a third party in between listening to and possibly modifying any data sent. MitM attacks will be further discussed in Subsection 3.7.6.

To mitigate the risk of a physical attack, physical access control is needed. Devices should be placed in locked cabinets or at least out of sight. Things like cables coming unplugged can easily happen by accident as well, so to speed up recovery, clear documentation of the network and all connections is recommended even if security wasn't a major concern.

3.7.2 Firewalls

While switches and routers are designed to connect devices and networks, firewalls are designed to keep networks separated. They deny any unnecessary connections between networks [24; 60]. According to Schneier [39] a firewall is like a castle wall that keeps unwanted intruders outside, but allows authorized entities in and out through a gate. He identifies four fundamental problems with firewalls:

1. A firewall can keep attackers outside, but if they can get inside, firewall is useless.
2. The network behind a firewall can be starved by denying any connections going in and out.
3. A firewall needs to cover all connections to the network, or attackers will just go around it.
4. A firewall needs to somehow know which packets or connections are malicious and need to be blocked.

Based on these fundamental problems, Schneier [39] recognizes three basic ways to compromise a firewall. One way is to go around it. Just like all large networks in general, ICS networks have multiple access points for maintenance, remote access, monitoring or other similar purposes. If any one of these access points isn't covered by the firewall, circumventing the firewall is possible. Another way is to trick the firewall into thinking that your packets are not malicious. Typically this is achieved by creating a small piece of malicious code that the firewall allows through. The malicious code then opens a connection from behind the firewall to allow the attacker in. The third way is to gain control of the firewall, which is usually possible through vulnerabilities in the firewall software or in the operating system running the firewall.

A firewall is only effective when it is properly configured and managed. Proper configuration is not straightforward and depends on the network environment [60]. A good

practice is to use a *default deny* policy where all connections are denied unless they have been specifically allowed [20; 60]. A poorly configured firewall provides a false sense of security and therefore poses a threat to security [39].

3.7.3 Network security tools

Network security tools are designed to find vulnerabilities and collect statistics from the network. Although these tools are intended for network designers and implementers, they are equally useful for cyber attackers [61]. Hoque et al. [61] present a wide variety of different tools that can be used by both defenders and attackers to test the network or launch attacks. Most of the tools presented are available for anyone and some of them have graphical user interfaces to make them user-friendly as well. The tools can be used in all kinds of attacks, including Denial-of-Service attacks, spoofing, network traffic capturing, or network scanning.

Network scanning tools are used to find information about the target network by sending carefully chosen packets and analyzing the responses. Scanning allows an attacker to identify devices such as firewalls or servers in the network and find possible vulnerabilities in them. Network scanning is typically used in the reconnaissance phase of the attack to gain knowledge on the network structure.

3.7.4 Wireless networks

In a wired network the concern of an attacker attacking a physical wire is rather trivial, since attacking the endpoints of a connection or the intermediary devices is easier and far more convenient. In a wireless network though the data sent over a wireless channel can be captured by anyone within the transmission range. Therefore the basic principle of wireless network security is to make sure that only the intended receiver can make use of the data sent.

Wireless network standards used in industrial applications, such as WLAN, WiMAX and ZigBee, are generally based on the IEEE 802 series standards [20]. All of the aforementioned are capable of encrypting traffic, but if encryption has not been enabled, anyone within the coverage area can send and receive messages. Encryption provides security over the wireless channel by making messages unreadable to anyone who doesn’t have the *key* used to encrypt them.

Even if the traffic is encrypted, an attacker might be able to obtain the encryption key for example from one of the endpoints or through a vulnerability in the encryption algorithm. Just like password hashing algorithms, encryption algorithms get outdated as the calculation power of processors increases. For example WEP (Wired Equivalent Privacy) encryption method used in early WLAN applications is now considered inadequate,

because a regular laptop PC can test all possible encryption keys in a decent amount of time, thus rendering WEP practically useless.

Even if an attacker can't crack the encryption, there are still other possibilities. The attacker can for example try to disable the encryption by attacking a base station in the wireless network. Base stations are configured with a configuration file which is usually protected by a username and a password that can be attacked. An attacker can use a DoS attack causing unnecessary load and delay in the network. A network can also be jammed by reserving all available wireless channels so other devices will have to wait for a free channel, as was discussed in Sub-section 3.2.2.

To improve cyber security of wireless networks, encryption should always be used. The problem with encryption in ICS networks is that it introduces delay and therefore doesn't fit well in systems with strict latency requirements. Coverage of the network needs to be considered as well. The placement of base stations should be designed so that the coverage of the network isn't any larger than is required by the application. The larger the area covered is, the easier it is for an attacker to get inside it. For example directional antennas can be used to control the shape and size of the coverage area.

Physical access control is equally important. It is used to keep attackers outside of the coverage area and away from the devices themselves. To prevent any unintended or malicious changes to the configuration of base stations, strong password protection or other authentication methods should be used. In addition, any new devices joining the network should be authenticated. This will make it significantly harder to spoof a base station or any other network device, as was described in Section 3.3.

3.7.5 Network protocols

Network protocols set the rules for communication, thus making networking possible. Sometimes these protocols are faulty by design or one of their functions can be misused. Since ICS networks are moving towards better connectivity with other networks, including the Internet, the protocols used are more or less the same than in other networks [20]. There isn't much a network designer can do to improve the security of a protocol. Instead, threats caused by protocols need to be handled otherwise, for example with a firewall, anti-virus software or an Intrusion Detection System (IDS).

3.7.6 Man in the Middle

In a Man-in-the-Middle (MitM) attack the attacker intercepts and modifies communicated data between two or more entities [62]. It is characteristic to MitM attacks that from the communicating entities' viewpoint the connection seems to be working as usual. MitM attacks are analogous to wiretapping a telephone line in order to listen to a phone call, but with the distinction that a MitM attacker is able to modify the data as it

is transferred. The main prerequisite for a MitM attack is to redirect the communication through the attacker's computer

The lesson to be learned from MitM attacks is that other devices in any network or the data they send shouldn't be blindly trusted. The transmission path, whether it is wired or wireless, shouldn't be trusted either. To guarantee integrity and confidentiality of the data, it is necessary to authenticate devices and their users, and encrypt all data sent.

3.8 Misuse of process automation data

Vulnerabilities, as presented in previous sections, have often led to the possibility of modifying network packets. This section describes how modifying process automation data might affect the process. There are multiple desired outcomes an attacker might be trying to achieve, but the simplest attack would be to remove, or *drop*, packets before they reach their destination. If all packets are dropped, the entire process will come to a halt, but if packets are dropped selectively, for example security or safety systems could be bypassed. If the data traffic is unencrypted, all that is required is some knowledge about the network structure and the process itself.

If the attacker is able to gain more comprehensive knowledge about the process and how it is managed, he is also able to make controlled changes to the messages sent between components of the system. In the container handling automation case of this thesis container moves are requested by Terminal Operating System (TOS) as *job orders*. Job orders contain a container ID and the location where the container needs to be moved. By altering these fields before the order reaches Terminal Logistics System (TLS), an attacker can take control of the process. If the attacker wants to stay undetected longer, he can also modify confirmations coming from TLS to trick the TOS operator into thinking that jobs are completed just as they were requested.

The infamous computer worm Stuxnet behaves in a similar manner [1]. It isolates the control system (CS) completely from the sensors and starts mimicking the sensor data. As long as the sensor data sent by Stuxnet to the CS is the same as the data coming from the actual sensors, the process remains completely normal. As soon as the malicious part of Stuxnet is activated, control messages coming from the CS are dropped and Stuxnet takes control of the process. It also sends a previously recorded stream of sensor data to the CS, so for the system operators in the control room the process still seems normal. The case of Stuxnet illustrates how, with enough resources, knowledge and a suitable vulnerability, it is possible to create a highly automated attack with no human interaction required after the initial compromise.

In IACS environments communication often needs to happen as much in real time as possible. On the other hand, cyber security measures, like encryption, in many cases increase latency and therefore they are currently often omitted in the favor of better per-

formance and lower latency [20]. Instead of just discarding all security measures, the maximum tolerable latency of the system should be tested or calculated, and based on that information applicable security measures should be chosen.

3.9 Data breach

A data breach means that someone without the authority to do so, is able to access a certain data asset. The severity of a data breach depends on the value of the data. In industrial networks attackers are generally interested in assets like financial data, e-mail messages and classified documents. In process automation systems data related to the process and information about communication protocols between different components of the system would be useful for an attacker. In terminal automation systems any data related to containers and their contents is of particular interest, which was also the case in the Port of Antwerp example described earlier.

3.9.1 Complicated attacks

There are multiple attack vectors for a data breach depending on the capabilities of the attacker. If an attacker is able to access the physical premises of the target, he can for example steal a device, like a laptop, with valuable information in it, transfer data to a portable memory device or install a malicious device that gathers data. If physical access is not possible, malware can be used to capture data and send it to the attacker. Composing malware like this requires extensive knowledge and therefore isn't a choice for less experienced attackers, but pre-made attack tools that exploit certain vulnerabilities can make such efforts much easier.

3.9.2 Browser-based attacks

If the capabilities of an attacker aren't high enough for the attacks above, there are also simpler attacks that can be deployed over the Internet. Man-in-the-Middle attacks discussed in Sub-section 3.7.6, like cookie poisoning, Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) are typically used in data breach purposes. These attack methods are fairly easy to use, but probability of a successful attack is lower, because they rely on an inadvertent user to trigger the attack. These attacks also require a web browser, which shouldn't be necessary on all computers, especially in ICS environments.

3.9.3 Injections

Industrial networks often have dedicated databases for various data assets. Interaction with databases usually happens through *queries*. A query is used to request certain information from the database. For example a query requesting all database entries with

the name ‘John’ in the field ‘first name’ would result in a reply with all the people named ‘John’ in the database. Queries can also be used to add or remove entries in the database. From the users viewpoint query construction is usually automated for usability reasons. The user enters some variables into a user interface and the queries are constructed based on the variables [63; 64].

If an attacker gains access to the user interface, he may try to enter malicious inputs in order to perform queries that wouldn’t be possible when using the interface as was intended. If the attacker knows (or guesses) the query language used, he can *inject* valid query commands directly into the user interface [64]. If inputs aren’t checked when the query is constructed, the malicious commands are also executed by the database. The ultimate result is that the attacker can access all the assets stored in the database.

Execution of an injection attack only requires some knowledge about the query language used by the database. When the query language is not known, the best bet is to try SQL (Structured Query Language), which is often used in databases. SQL injections have been ranked as the most dangerous software error [53]. Since the vulnerability isn’t in the query language itself, but in the conversion from user input to query language, choosing a different language makes no difference security-wise. Injection attacks do not always target databases. For example operating system control commands can be injected as well, and sometimes triggering a buffer overflow can also be seen as an injection too. To avoid injections, the user inputs need to be analyzed before queries are formed [64].

3.9.4 Phishing

The target of a phishing attack is to gain information from businesses or individuals, usually by impersonating a legitimate authority [65]. Phishing can happen for example through forged web sites or e-mails. Phishers are usually interested in banking accounts, credit card information or something else with relatively high rewards, but phishing can also be used in reconnaissance phase of other attacks. The attacker can for example impersonate a system administrator by e-mail and ask for username and password in order to complete some maintenance actions. Phishing attacks are always automated and target large groups of individuals or businesses [66]. Similar attacks targeting a single person or a small group of people are called spear-phishing attacks.

3.9.5 Data breach in reconnaissance phase

Data breach methods can be used in the reconnaissance phase of an attack to retrieve information about the target network or about devices and users in it. In this case the attacker would be interested in for example [26]:

- File names and directory listings.

- Internal IP addresses, names and types of devices, especially databases, management servers or log servers.
- External IP addresses and subnet address ranges.
- URLs for Intranet pages.
- Information about any services that can be accessed remotely, such as VPN or e-mail.
- List of open ports in firewalls.
- Lists of usernames and/or password hashes.
- Anti-virus software and other security products used.

The list above is not exhaustive and targets always need to be assessed case-specifically, but shows that protecting information about the network is as important as protecting information in the network.

3.10 System users

In the previous sections a multitude of threats have been discovered and explained. Most, if not all, can be tackled with technological solutions. Even if all the necessary solutions were in place, properly configured and maintained, the system still remains vulnerable. It is often said that users are the weakest link in the chain of cyber security [13; 39; 67]. According to a recent study [17], “Low security awareness among employees is the greatest inhibitor to adequately defending against cyber threats”. This means that it is the system user, who often opens doors for attackers, both literally and figuratively speaking.

In some cases doors are opened by accident. In fact, 11% of all cyber security incidents in ICS systems are caused by mistakes made by operators [14]. Sometimes users bypass security measures deliberately, because they make work more difficult, prevent access to a certain website or cause some other inconveniences. This phenomenon is well-known not only in the field of security, but in safety as well.

Because security and safety measures tend to have a negative effect on usability of the system, users start to figure out ways around them, usually with little understanding on how it affects the security of the entire system [67]. Simple things like writing passwords down on a piece of paper or leaving authentication disabled on a wireless access point improve usability, but at the same time attacking becomes easier. For this reason it would be beneficial to take the usability viewpoint in consideration when comparing different security measures.

Centre for the Protection of National Infrastructure (CPNI) has put together a good practice guide [68] on how to improve cyber security skills and awareness of employees in ICS and SCADA environments. The guide describes three main principles:

1. Increase ongoing awareness.
2. Establish training frameworks.
3. Develop working relationships.

Increasing awareness helps employees to understand why security is needed and the business impact of security breaches. It is also important that the employees know what to do in the event of a security incident. Many attacks at some point require some form of interaction with a human target, which means that knowledgeable employees can prevent attacks.

Training provides the employees with the necessary skills to carry out their work without endangering cyber security. Relationship development refers to the fact that in ICS environments cyber security should be created collaboratively applying the best knowledge from both IT and process control fields. Effective collaboration between the two communities improves not only cyber security, but staff utilization and cost management as well. [68]

3.10.1 Portable devices

Portable devices are efficient tools in delivering malware to a target system, since over time they will most likely be attached to many different devices and connected to many different networks. A NIST good practice guide [52] actually states that it should be assumed that portable devices will become infected, so they should be treated accordingly upon connection. Infected devices such as laptops or USB-drives are significant threats in office networks, but they can just as well be utilized when attacking ICS networks too.

One possible case would be a system operator who wants to install some program (for example a game) on his work desktop to make his work easier or just to pass time. The work desktop isn't most likely connected to the Internet so he installs the program to his USB-drive on his own computer (which happens to be infected with a worm) and plugs it in at work. Since the worm is programmed to attach to any new peripherals, it propagates first to the USB-drive and then to the work desktop. Soon the worm will infect all other computers on the ICS network. There are two noteworthy things in this example. First, a single ill-advised employee was able to infect all computers on the network. Second, even isolating the entire ICS network from the Internet wouldn't have prevented the infection.

In 2013 the entire computer network of the Cook County Department of Highway and Transport was shut down due to a virus infection. Apparently the virus was allowed in by an employee using a flash drive from home or surfing the Internet [69]. Around 200 computers were infected and it took five IT-technicians nine days to clean the system [69]. This example illustrates how seemingly small mistakes by users can lead to quite

significant consequences. In 2012 Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) reported [70] two separate but very similar incidents, where malware infected a power generation control system. In both cases the root cause turned out to be an infected USB-drive [70].

3.10.2 Social engineering

Even when the users are well trained and rarely make mistakes by themselves, a clever attacker can persuade or trick a user into a slip-up. A simple example would be a situation where an attacker pretends to have forgotten his keys inside and ask for a legitimate employee to open the door for him. If that doesn't work, he can pretend to be a repairman with hands full of tools and wait for someone to politely open the locked door for him. Attacks like this are referred to as social engineering. They typically rely on people's willingness to help [39].

Social engineering works over the Internet as well. All an attacker needs is e-mail addresses and some other personal information about one or two employees in his target company. He can then send e-mails impersonating the system administrator and ask for username and password to perform maintenance, or he can send a malicious software disguised as a software update and instruct the employees to install it immediately. In addition, Schneier [39] mentions multiple cases where social engineering has been used over the phone.

3.11 Cyber threats related to cloud services

There are two main concepts on how cloud services can be utilized in ICS environments. Either a cloud service is used for data acquisition and distribution purposes, or the entire ICS system is run on a cloud platform. The latter concept is better suited for highly distributed control systems, such as smart grids or water supply systems [71]. Accordingly, the first concept is more suitable in the context of terminal automation.

Although cloud computing is relatively new technology in the field of ICS, it has been suggested that system performance and availability could be improved while total cost of ownership could be reduced using cloud services [71]. When applying cloud technologies, there are some security issues to be considered. First concern is storage of the data: data of different users can be separated either physically or virtually. Virtual separation means that data of different users can be located on the same physical hard drive, but the virtualization software keeps one user's data separated from everyone else's data [72]. In theory this is equally secure to physical separation, but in practice virtual separation may sometimes fail due to a programming error or some kind of malfunction. In this regard it would be beneficial to assess the consequences of stored data leaking to the public before making a decision on moving that data to a cloud [73].

Another concern is the physical location where the service is hosted. Different countries have different laws about data management and the physical environment poses its own threats [71]. The maintenance and equipment contractors used by the service provider affect the security of the cloud service as well [72]. Connections in and out of the cloud need to be secure as well. Proper encryption and authentication provide secure connections, but if the purpose of the cloud is to distribute data to third parties, their endpoints need to be secure too. None of the aforementioned security features serve any purpose if an attacker is able to compromise a third party and use a legitimate account to access the cloud.

If a service provider is able to provide the required level of security and a contract is made, according to the Finnish National Cyber Security Centre [72] it is important to agree at least on the following three topics:

1. What is the level of availability provided by the service? What is the maximum latency tolerated?
2. What is the level of user support? What is the response time during weekdays? What about weekends and public holidays?
3. Where is the data stored? Who is allowed to modify it?

The agreement on these topics is called a Service Level Agreement (SLA). An SLA can also include agreement on how performance of the service is measured and what sanctions are imposed if the performance metrics are not met [72].

4. DETAILED THREAT ANALYSIS OF EIS

This chapter presents a threat analysis of External Interface Service (EIS). First, in Section 4.1, the purpose and structure of EIS are described. Then, in Section 4.2, the different motivations for attacking EIS are explained based on the functionality and data handled by EIS. In Section 4.3 cyber security concerns related to EIS are explored. Tools used to carry out the threat assessment are described in Section 4.4 and results of the assessment are presented in Section 4.5. Based on the findings in Sections 4.3 and 4.5, suggestions are made in Section 4.6 on how to improve cyber security. Finally, the findings of this study are summarized in Section 4.7.

4.1 External Interface Service

External Interface Service (EIS) is a software component of Kalmar's Terminal Logistics System (TLS). The purpose of EIS is to provide an interface for communication with external systems, such as a Terminal Operating System (TOS) or any other 3rd party systems. EIS also inspects and routes all messages coming to TLS from the third party systems. In practice, EIS is primarily used to send and receive container move requests and container yard status data between TLS and TOS. The general architecture is presented in Figure 6 below.

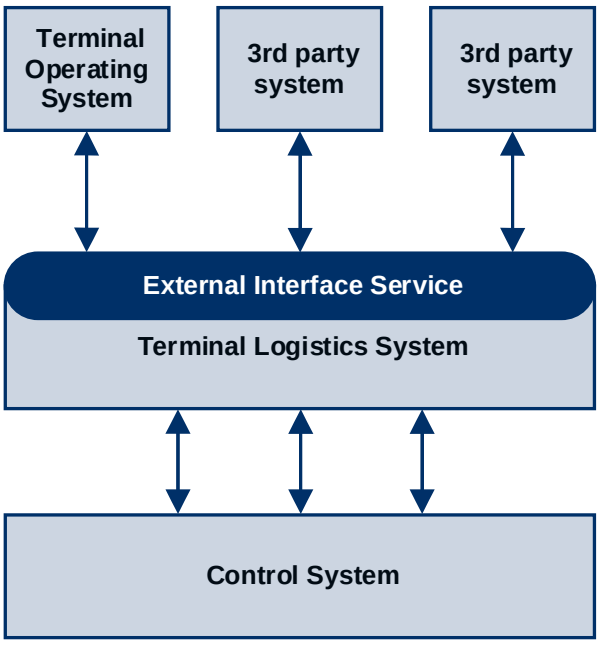

Figure 6. *Location of EIS within the architecture.*

EIS is built on the OSGi architecture. OSGi is a specification that defines a dynamic component system for Java [74]. OSGi runs on a Java Virtual Machine (JVM) to provide independency from the native operating system. In the OSGi architecture functionality is created with *bundles* that are essentially Java Archive (JAR) files [74]. Bundles can be installed, started, stopped and uninstalled without stopping the JVM, which means that bundles can be swapped or updated without interrupting the entire automation process.

In EIS there are separate bundles for example for TCP/IP communication handling, persistence handling and naming conversions [75]. EIS can also be adjusted to fit different customers' needs through project specific customization bundles [75]. In the scope of this study, there is no need to gain deep understanding on how OSGi works, but the framework is presented in Figure 7. OSGi framework is thoroughly described in the specification [74].

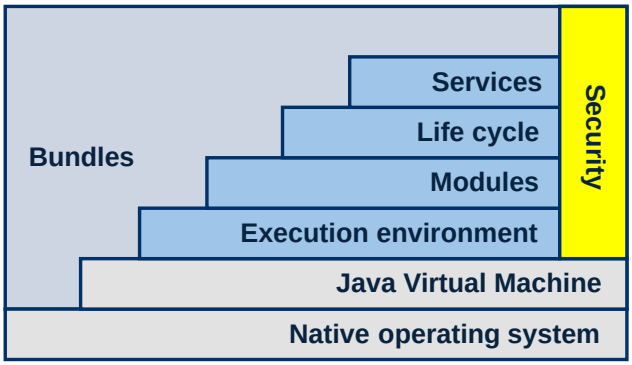

Figure 7. *Components of the OSGi framework. Adapted from OSGi Alliance [74].*

The JVM runs on a VMware virtualization product called vSphere, which allows multiple instances of JVM to be run on the same physical server machine [75]. This means, that all components of TLS, including EIS, are run on separate virtual environments, but on a single physical machine. While the components share their processing and memory resources, in theory the virtualization prevents them from interfering with each other. The components still can, and need to, communicate with each other over a virtual network as if they were on physically separate machines. To add redundancy, the physical machine is linked to an identical hardware setup in a different physical location, so in case of a failure, the execution can be moved over to the secondary hardware.

To communicate with other components, EIS uses a Java Message Service (JMS) broker called Apache ActiveMQ. Messages are encoded in Extensible Markup Language (XML) format, so they are practically human-readable. To provide Quality of Service (QoS), messages are queued until the receiver is available to receive them, i.e. the messages are persistent. Persistence is required to make sure all messages get delivered and stay in the correct order even if the receiver is not available at all times. On network level EIS utilizes TCP/IP protocol suite.

4.2 Motivation for attacking EIS

EIS acts between two major components (TOS and TLS) of Kalmar's terminal automation system and handles practically all of the communication between them. This means that most of the information interesting for an attacker either constantly flows through EIS or is accessible for it. In addition, the level of abstraction in messages handled by EIS is suitable for exploits. For example giving instructions to a CHE meter-by-meter directly from the Control System would be a tedious task, while a CHE can be instructed to move a container with a single XML-formatted message from EIS. On the other hand EIS doesn't have any long-term data storages, so it's not an ideal target for data breaches.

In terms of terminal automation systems in general, four significant attack motives can be recognized:

- Interruption of container handling process.
- Stealing process-related data.
- Stealing a container.
- Moving a container through the terminal undetected.

4.2.1 Interruption of container handling process

Motivation for interruption of container handling process can vary from random attacks coming from the Internet to cyber terrorism. Even hostile nations could be involved, which means that attacks can be highly sophisticated and persistent. Interrupting the process can be seen as a Denial-of-Service attack and indeed most attack vectors make use of typical DoS attack methods. Terminal automation systems are fairly susceptible to DoS attacks, because they typically rely on availability of all the components of the system.

In most DoS cases a well-chosen target component can bring the entire system to a halt. In this regard EIS is critical: being the dedicated component to communicate with TOS, preventing the communication would stop CHEs from getting new job orders, thus leaving them idle. Consequences of a DoS attack would be mostly financial, but physical damage due to collisions is possible as well. CHEs have advanced safety features to prevent any collisions, but cyber attackers may be able to disable them or cause unusual situations that haven't been taken into account when designing these features.

4.2.2 Stealing process-related data

Leakage of data related to container handling process would most likely involve an industry competitor who is interested in the performance of the system. Key Performance Indicators (KPI), such as fuel usage, utilization rate, number of container moves per

hour and similar data may not seem useful for a general hacker, but a dishonest competitor may be willing to pay for them. As was already mentioned, EIS doesn't store such data, but it is possible to capture the data as it passes through EIS using Man-in-the-Middle type of attacks. EIS doesn't handle any information related to the monetary flows of the terminal so attackers targeting bank transfers or similar data would have to find other targets.

4.2.3 Stealing a container

The motivation for stealing a container full of valuable cargo is obvious. To steal a container, an attacker would have to instruct the system to move the desired container to a desired location, for example on a trailer of a truck, and take it away before the legitimate owner arrives to pick it up. For the purpose of stealing a container both content and location information of containers might be useful for an attacker. The location can be obtained for example from messages flowing through EIS, but the exact content of containers isn't stored in the system, as it isn't really relevant from the viewpoint of container handling.

However, the ISO 6346 coded container type is obtainable and this code may give a hint on the content. For example code *S1* refers to an *automobile container* [76]. Most of the codes aren't as specific about the content, but with some experience and knowledge on industry practices it should be possible to tell which containers are potentially more valuable than the others. Additionally, container information is needed to bind a physical container with its digital representation in the system, which is necessary when exploiting EIS to move a certain container.

4.2.4 Moving a container through the terminal undetected

Moving a container through the terminal from a vessel to a truck trailer or vice versa is certainly the most demanding of the four goals, but it is also the most useful for criminal purposes. Therefore criminal groups may be willing to invest a lot of money and effort into such attacks, which leads to complicated and well-planned attacks. Moving a container through the terminal requires much more than just a cyber attack on EIS, so it is not possible to fully assess this threat without the assessment of the entire environment. In this study the threat is only considered to find out if exploiting EIS would have any contribution to an attack like this.

4.3 Cyber security concerns in EIS

When discussing with Kalmar employees about EIS, it became clear that reliability and performance have been the main objectives during design and implementation. Because cyber security often contradicts with these objectives, it hasn't been given much empha-

sis. Instead, it is assumed that the environment where EIS is used has been appropriately secured. While this kind of "hard shell protecting a soft core" mentality is valid as long as the outer shell remains impenetrable, it becomes useless as soon as an attacker is able to bypass that shell of security. Therefore a multi-layer approach, termed as "defense-in-depth", is often recommended [15; 19-21; 29]. This approach implies that getting past a single layer of security doesn't grant access to every asset and to access the most valuable assets, an attacker would have to be able to bypass multiple layers. Applying the layered approach means implementing security to EIS as well as to other components.

To find the most useful solutions for securing EIS, security concerns related to EIS need to be discussed first. Security concerns have been assessed from three viewpoints: software, network and physical security concerns. Because the network and physical aspects depend on the actual environment where the system is applied, a single container terminal was chosen to be assessed. As the environment can be different in other terminals, some concerns may not apply to them. Respectively, it is possible that other terminals have concerns that haven't been identified here.

4.3.1 Software-related

EIS utilizes different Java-based components, all of which have their own vulnerabilities, and the operating system also has its own security flaws. This sub-section discusses those vulnerabilities and possible outcomes if they are exploited. Of course there is also plenty of custom code on top of the OSGi platform and although a code review isn't included in this thesis, the importance of secure programming practices is discussed as well.

OSGi

The OSGi platform is a common Component-Based System (CBS) that is often used in embedded systems and application servers [77]. The wide usage means that the security of OSGi has already been discussed and a variety of vulnerabilities have been disclosed [77; 78]. The reports of Parrend & Frénot [77; 78] explore attacks against the Java/OSGi platform itself and attacks from a malicious bundle against a legitimate one. They recognize three possible consequences from these attacks: unavailability of the system, performance degradation or unauthorized access. When considering EIS, the first two of these directly contribute to the "interruption of container handling process" attack motive. The last one mostly contributes to the "stealing process-related data" motive, but could also be used as a reconnaissance method to acquire knowledge about EIS and its functions.

Attacks leading to unavailability try to stop execution of certain bundles or shut down the entire OSGi platform while attacks leading to performance degradation start infinite or redundant processes to waste processing resources [77]. Unauthorized access in

OSGi can mean access to private data of other bundles or even access to the underlying operating system [77]. If attackers are able to access the operating system through EIS, they can also try to compromise other software components running on the same operating system.

Attacking OSGi usually happens through a malicious bundle [77]. Creating the bundle itself is relatively simple and only requires basic Java programming skills, but to exploit other bundles, an attacker would have to possess extensive knowledge on how the system actually operates. Naturally such information is difficult to obtain and requires determination from the attacker. Some Denial-of-Service attacks, such as shutting down the OSGi platform, do not require knowledge about other bundles, but the outcome of such attacks is fairly limited in the sense that they can only be used to interrupt the process, which admittedly is a serious consequence in itself.

While it would be difficult to obtain the information required to create an effective malicious bundle, it would also be very useful for the attacker. Reflecting the attack motivations, it would be possible to capture process-related data or inject container move requests without having to breach encryption or any other network level security systems. In the worst case scenario an attacker would be able to completely take over the process, much like in the Stuxnet incident discussed in Section 3.8. Therefore it is justified to assume that highly capable instances, such as criminal or terrorist groups, would be willing to make the investments required to carry out such attacks.

Virtualization

Virtualization is implemented on two different layers, both of which have their own security concerns. The OSGi framework runs on a Java Virtual Machine (JVM), and the JVM runs on the vSphere environment. JVM is required, as it is a part of the OSGi framework, and the purpose of the vSphere environment is to make it possible to run multiple JVMs on a single server, so both layers of virtualization are needed.

JVM offers an isolated execution environment, *a sandbox*, where applications, such as EIS, can be executed without any interference from other software running outside the sandbox. Despite the security model included in JVM, under certain conditions it is still possible for example to freeze the JVM, leak data or prevent termination of bundles [79]. While a frozen JVM and data leakage have obvious consequences, preventing termination of bundles may cause either resource exhaustion or alternatively an attacker can exploit it to prevent the system administrator from terminating a malicious bundle. In such case the administrator would be forced to shut down the entire JVM to stop the malicious bundle [79].

While the aforementioned exploits happen inside the JVM, escaping the virtualization sandbox of vSphere poses a whole different threat. Since EIS is running on the same physical server than other software components, it is possible to find a vulnerability in

EIS, OSGi, JVM or vSphere that allows attackers to leave the sandbox and attack other software components. Similarly, attackers can find their way to EIS through those other components. This means that even though EIS in itself isn't accessible directly from the Internet, remote attacks are still possible if any other component on the server cluster has Internet access and the attacker is able to break out of that component's virtual environment.

JVM is also known to be susceptible to a phenomenon called *aging* [80]. Aging means that applications running long periods of time without interruptions cause for example memory depletion and decreasing of performance [80]. While aging is not something that could potentially be exploited by an attacker, it can still be considered as a threat because it can cause the system to hang or become unstable. This is especially true for EIS and other software components of the system that see minimal downtimes and are expected to be constantly running and available.

ActiveMQ

ActiveMQ is not a component of EIS, but rather a message broker that EIS among other components uses as a service for communication. To facilitate persistent messaging, ActiveMQ has *queues* for point-to-point messaging and *topics* for point-to-multipoint messaging [81]. The purpose of queues and topics is to store messages until the receiver is ready to consume them.

As was noted before, communication is an important part of EIS' functionality, so secure communication plays a major role when securing EIS. ActiveMQ offers two security measures: authentication and authorization [81]. Both are optional to use, and currently Kalmar's solution implements neither. Consequently, there is no control over which clients are allowed to read messages from a certain queue or topic. As Kalra [81] illustrates, it is possible for attackers to set up their own messaging clients and start consuming messages. Messages can be consumed with or without removing them from queues or topics, so it is possible to either disrupt messaging by removing messages or read messages without altering the process [81].

Kalra also presents a security assessment tool called JMSDigger, which is dedicated to testing ActiveMQ-based applications. Among other features JMSDigger is able to retrieve messages from queues and topics. The tool is available for anyone to download for free [82]. While the tool was designed for security professionals, it is just as useful for attackers.

Flooding attacks filling ActiveMQ queues or topics with unnecessary messages were also considered, but it was concluded that it would take a fairly significant amount of time and messages to cause any noticeable delay. This is due to the memory capacity of the queues and topics, and greater tolerance of delivery delay in EIS compared to some other components.

ActiveMQ also provides the possibility to assign different priority values for messages, but this feature is not used in this system. Therefore it is not possible to abuse prioritization by flooding the system with crafted messages assigned with the highest possible priority, thus preventing or slowing down delivery of any messages with lesser priority.

Operating System

The Kalmar Terminal Logistics System (including EIS) is currently specified to be run on a Windows Server 2008 R2 Operating System (OS) [75]. As all other OSs, this one comes with its own set of vulnerabilities [21]. According to Common Vulnerability Enumeration (CVE) [83] the OS has been affected by more than a hundred publicly disclosed vulnerabilities during its lifecycle. Patches are of course released to fix these vulnerabilities as they are disclosed, but the problem is that these updates aren't being installed regularly, but only during other maintenance operations, if even then. From the business viewpoint these maintenance breaks should be as few and far between as possible, so the intervals between update release and update installation tend to become lengthy. As was already discussed in Section 3.5, software updates may also cause unexpected changes to the behavior of the OS.

Being the underlying system, the OS controls all other software that is running on it. It also provides access to the configuration files of all the other software, such as the JVM and EIS. Even if an attacker doesn't have the knowledge to exploit these files, it is possible to remove them or some other important data. Therefore it is critical to ensure that OS can't be exploited.

Kalmar currently suggests the use of a client-server anti-malware product in order to protect the server [75]. It is also suggested that the program is only updated manually during other maintenance operations to avoid any excess network traffic [75]. Lengthy update intervals lower the detection capability of the anti-malware program and the system remains susceptible to the latest malware and other recently emerged exploits.

Software design

The last software-related security concern is the EIS software itself. In this case the software is in the form of bundles that are written in Java programming language. While Java is an inherently more secure programming language than for example commonly used C or C++, it has certain pitfalls that the programmer needs to be aware of [84]. Some of these are specific to Java, while some are common to all programming languages.

Howard et al. [85] present 24 common programming flaws that lead to security vulnerabilities. Eight of these were identified to be applicable to EIS:

1. *Buffer overflows*: while Java isn't as prone to overflows as for example C, they are still possible.

2. *Integer overflows*: integer arithmetic using binary representation will under certain conditions produce incorrect results. Incorrect results of course have an impact on further calculations.
3. *Format string flaws*: if inputs are not checked for appropriate length or syntax, an attacker may be able to read from or write to memory. EIS handles and restructures messages, so malicious changes to these messages could result in exploitation.
4. *Command injections*: similar to format string flaws, but command injection refers specifically to using control commands as the malicious input. When successful, allows execution of any control command, leading to takeover of the system.
5. *Failure to handle errors correctly*: if errors aren't handled correctly, the system may end up in insecure, possibly unstable state. Exploited by Denial-of-Service attacks.
6. *Information leakage*: Howard et al. use the term to describe a situation where information about the system is exposed to an attacker. For example software version numbers and detailed error messages help an attacker to identify vulnerabilities.
7. *Race conditions*: a situation where a variable is changed between time-of-check and time-of-use. Even if the variable is valid at the time-of-check, it may have been changed by a malicious actor before the time-of-use.
8. *Executing code with too much privilege*: using the lowest level of privilege possible mitigates the consequences in case something goes wrong. If an attacker is able to gain control of the execution process, the possible actions depend on the privileges that have been given to the process.

4.3.2 Network-related

When assessing the security of EIS, it is important to also assess the security of the network around EIS. Accessibility of a network and the devices in it often determine how easy or difficult the initial break-in is. This of course applies to remote attacks only, physical attacks, including physical access, are considered later in Sub-section 4.3.3.

Network design

The network where EIS is located includes TLS, CS and CHEs, so let's call it the process network. A firewall is used to control access to the process network from TOS and the Internet. According to network documentation, the process network can be accessed over the Internet through a VPN connection from Kalmar's Intranet, but direct access is not possible. This would make remote attacks more complicated, but eliminating direct access doesn't guarantee security, as attackers or malware may still propagate from neighboring networks or through portable devices. Upon closer inspection, the firewall was found to be misconfigured and reachable from the Internet, which makes it also visible to network scanners that systematically scan the Internet for vulnerable targets.

There is also a remote control desk, which allows controlling the Container Handling Equipment remotely. From the process network's viewpoint the desk is a workstation that is directly connected to the process server. There are also other workstations inside the process network to provide monitoring capabilities. These workstations can be used as an intermediary to attack the process server. The process server is the server that runs all software required by the automation process.

Virtual Local Area Networks (VLAN) are used to separate different functions to their own virtual networks. While VLANs make the network more manageable and can be thought of as a security feature, they also introduce new security concerns if they are not configured properly. In theory VLANs prevent devices of one VLAN from communicating with devices in another VLAN, which will also limit accessibility for an attacker. It has been shown that there are vulnerabilities that allow an attacker to "hop" between VLANs and send messages between them [86]. Nowadays though, these vulnerabilities are well known and they can be mitigated by correct configuration of the VLANs [87].

Encryption and authentication

Any communication inside or across the boundaries of the network is unencrypted, which means that messages can be captured from and injected to the network. The ability to capture and inject messages is a major part in achieving any of the four attack motivations, as Section 4.5 will demonstrate. Capturing messages could reveal information on how the system works or what the containers may carry, while injecting messages can alter the course of the container handling process or stop the process entirely.

Devices in the network are protected with a username-password combination to prevent unauthorized access, but reviewing internal documentation revealed that combinations are fairly generic and susceptible to guessing or bruteforce attacks. In addition, new devices joining the network are not authenticated, so adding a malicious device to the network is simply the matter of gaining physical access to the premises. The messages are not authenticated either. Although they are kept in sequence with a sequence number, it is trivial to calculate a valid sequence number as it is only checked that the number has not been used yet, meaning that the numbers don't even have to be consecutive.

Networking devices

There are a few different network switches from two manufacturers used in the network. Review of the configuration files revealed that the switches are remotely configurable and password protected. All switches allow configuration over an encrypted HTTPS or SSH connection, but many of them also allow insecure HTTP or Telnet connections. The problem with HTTP and Telnet protocols is that they deliver all communication, including passwords, unencrypted. Remote configuration allows network administrators to change settings of a switch without having to go to the physical location of the device. While this is a useful feature, it may also allow an attacker to alter the routing

rules in the switch, redirecting all packets through the attackers' computer or some other capturing device. The only perimeter protecting the switches is the username-password combination, which was already found to be susceptible.

As was already noted, the firewall between TOS and TLS is visible to the Internet due to a misconfiguration. This poses a significant threat, as the firewall is the only connection between TOS and TLS, which means that failure of that connection would stop the entire process. For example a simple flooding attack from a botnet could bring the firewall down and depending on the capabilities of the local employees it might take anything from minutes to days to locate and fix the problem.

Failure of the firewall has been taken into account in network design and there is a backup firewall, but there is no automatic switch-over system in the case of a failure, which makes the failure easy to fix, but only after it has been identified. Looking at the network documentation, it isn't apparent whether the backup firewall is supposed to be connected to the Internet in the event of a failure. A flooding attack from the Internet is likely to continue as soon as the backup firewall is put online, so instructions should be clear on how to configure the backup firewall in such incidents.

4.3.3 Physical and human-related

Based on discussions with Kalmar employees, physical security of the terminal in question has room for improvement. There are no strict access control policies in place and physical access control is susceptible to social engineering attacks. Process servers and networking equipment have been placed with easy maintenance in mind so they are not behind locked doors or away from plain sight. An attacker pretending to be a repairperson or Kalmar employee would have easy access to vital components of the system and plenty of time to fiddle with them.

It is well known that with a physical access to the device a skillful attacker is able to bypass any security measures that might be in place. Therefore such attack vectors need to be considered even though they might seem fairly unlikely to happen. Even if attackers weren't able to access the servers or any other devices, they can leave malware infected USB-drives lying around or look for any notes with passwords on them near the workstations.

Unlike to the site, access to the container yard is strictly controlled for safety reasons. Despite strict control, it is possible to exploit EIS to bypass access control. Access is denied or granted based on an identifier stored on an ID card and the identifier travels through EIS before the decision is made. The allow/deny response message travels through EIS as well. Manipulating either of these messages allows an attacker to control access to the container yard. Access to the yard could serve as an intermediary for a physical attack or for reconnaissance.

EIS has no user interface or human users so it's not susceptible to misuse or mistakes made by users, but human error is still possible during configuration or software development. Such errors might lead to unintended behavior of the system, which in most cases would lead to Denial of Service or degradation of performance.

As in any system, it is possible that a device or cable physically breaks. This can happen purposefully with malicious intentions, accidentally, or because of age or poor quality. The consequences of such incidents can be similar to cyber attacks even if they weren't actually attacks nor had anything to do with cyberspace.

4.4 Threat assessment tools

In today's networked world, it is not possible to achieve complete cyber security, and therefore cyber security should be viewed as risk assessment [20]. This thesis applies assessment methods and tools to find points of improvement for cyber security of EIS.

As was shown in Chapter 3, attacks targeting ICS systems are great in number, variety and complexity. Therefore it is absolutely necessary to have a tool capable of formalizing, phasing and visualizing the process of cyber threat assessment. Luckily, there already is a selection of tools specifically made for the purpose [22; 88]. These include attack trees [89], CORAS [90], EBIOS [91], INSAW [92] and OCTAVE [93].

Attack trees were chosen to be used in this thesis, because the concept appears to be already widely known in the field of process automation due to the extensive use of fault trees in safety analysis. The well-known concept supports communication among security, safety and automation professionals. In addition the relative simplicity of attack trees makes them suitable for the limited time frame of this study. Attack trees have also been previously used in the ICS field [94].

Attack trees were first introduced by Schneier [89] in 1999 as a formal way to present attacks targeting a system. Similar tree-like structures have been used earlier in fault management as fault trees, but Schneier introduced them to the world of cyber security [95]. Attack trees deconstruct an attack into more concrete attacks or goals, thus lowering the level of abstraction and making attacks easier to understand [95].

Construction of an attack tree begins by recognizing main goals of possible attackers. These are called *roots*. To achieve a root goal, an attacker needs to achieve one or more sub-goals first. These sub-goals are presented as *nodes* of the attack tree. If necessary, a sub-goal may again be divided into nodes. Nodes that can't be further divided are called *leaves* of the tree. If nodes are combined with an AND node, all of them need to be completed to achieve the goal. A path all the way from a leaf node to the root is called an attack vector. A sample attack tree is presented in Figure 8. The attack trees presented in this thesis have been drawn using a security modeling tool called SeaMonster [96].

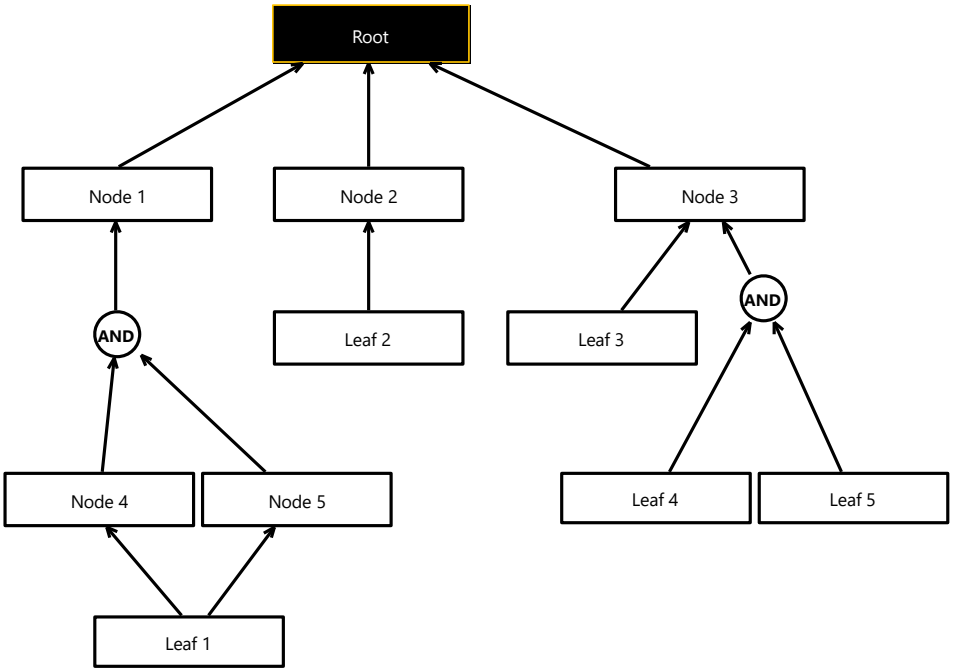

Figure 8. *Example of an attack tree.*

The description of attack trees given above is similar to the one originally given by Schneier [89], but more formal definitions exist as well. Both Mauw & Oostdijk [95] and Moore et al. [97] present a more mathematical definition through graph theory, but for the purpose of this thesis, it is not necessary to examine those definitions any further.

Once the attack trees have been constructed, nodes are rated according to the difficulty of their execution. The rating scale used here was adapted from Byres et al. [94], but other similar scales could be used as well. The scale has four different levels of difficulty: trivial, moderate, difficult and unlikely. The levels are described in Table 1. The benefit of a rating system is that once rated, different attack vectors become comparable with each other in terms of difficulty. In risk assessment difficulty is an important factor as it has a direct influence on probability.

Table 1. *Difficulty rating for attack tree nodes. Based on Byres et al. [94].*

Level	Level name	Level description	Capable attackers
1	Trivial	Little or no technical knowledge required	Anyone
2	Moderate	Average skills and knowledge of cyber attacks	Cyber enthusiast, criminal, industry competitor
3	Difficult	Significant amount of technical expertise	Industry competitor, organized crime
4	Unlikely	Beyond the known capacity of hackers	National level actors

The rating is done by estimating difficulty of each leaf node. Difficulty of other nodes is then derived by their sub-goals (also called *children*). If a node has multiple children that are connected through an AND node, it inherits the highest rating among the children. If a node has children that are connected without an AND node, it inherits the lowest rating among the children. For example if a node has two children (with difficulty ratings of 2 and 3) connected to it through an AND node, the parent node is given a rating of 3. If the same node also has a child node (with the rating of 1) that is not connected through the AND node, then the parent node is given a rating of 1.

Once all nodes have been rated, different attack vectors can be compared and the shortest and easiest paths to the root nodes are found. These results are useful when designing and implementing security measures or conducting risk assessment. Easiest path can be deemed as the most probable while more demanding paths are exclusive to more organized attackers. To complete the assessment, the impact of most critical paths is assessed on a scale of 1 to 4 (see Table 2). An attack qualifies to a certain level if it fulfills one or more of the level descriptors.

Finally, the probability and impact ratings are combined in a risk matrix, where different attack vectors are placed according to their ratings. Based on the risk matrix it is easier to decide which risks should be mitigated and which are acceptable. The structure of this method is compliant with ISO/IEC 27005 standard [98], covering the three phases of risk assessment: identification, analysis and evaluation.

Table 2. *Description of different impact levels for attack vectors.*

Level	Level name	Level description
1	Minimal	No interruption of process No physical damage No personal injuries Little to no financial damage No data leakage
2	Minor	Interruption of process (hours) Physical damage No personal injuries Small financial damage Small leakage of non-valuable data
3	Major	Interruption of process (days) Significant physical damage Personal injury Financial damage Leakage/loss of valuable data
4	Catastrophic	Lengthy interruption of process Expensive physical damage Life-threatening personal injury Significant financial damage Extensive leakage/loss of valuable data

4.5 Threat assessment

In this section, the process of forming attack trees from the security concerns of Section 4.3 is discussed. The scoring system presented in the previous chapter is then applied to rate the attack vectors. In addition, three of the attack vectors are further examined to provide more detailed examples of attacks.

4.5.1 Attack trees

The attack trees turned out to be large and complicated, so, for the sake of readability and clarity, most of them have been divided into smaller sections. There are also certain sections that are present in most of the attacks, so to avoid repetition, those sections are presented as separate trees and then referred to in other trees. Due to the size of the trees, they are only provided in Appendix A: Attack trees. In the appendix, the content of the trees has also been explained in greater detail.

All nodes in the attack trees have been rated using the difficulty rating of Table 1 and the score of each node is presented in brackets after the name the node. Like the security concerns in Section 4.3, the attack trees presented in this thesis are based on a single terminal, meaning that they represent threats to that specific terminal. While the threats are fairly similar in other terminals as the system itself isn't that different, changes especially in the physical environment and hardware in use have an impact on the scoring of various nodes.

In this particular case the physical access control was found to be insufficient, so the difficulty rating for physical attacks has been set low. Other terminals may have more effective control, which in turn would make physical attacks significantly more difficult. On the other hand, as the case of Antwerp suggests, physical access control might be a more common problem among terminals and ports in general. The misconfigured firewall also had an impact on the result.

The attack trees were constructed using three main goals as root nodes: denial of service, moving CHE and data leakage. These three goals reflect the attack motivations described in Section 4.2. In addition, gaining knowledge on the target system and unauthorized access were treated as separate trees, as they were a part of most other trees.

4.5.2 Threat classification

The attack trees reveal all the possible attack vectors, but presenting and scoring each vector separately would introduce a lot of redundancy, and thereby make the risk matrix difficult to read and sometimes even misleading. Therefore, attack vectors with similar scores have been bundled together. This allows for reduction of the number of attack vectors to 14. The bundled attack vectors are presented in Table 3 with their scores.

Table 3. *Attack vector enumeration and scoring.*

Goal	Attack vector	Matrix ID	Scores	
			Difficulty	Impact
Denial of Service	Flood devices accessible from the Internet	A	1	2
	Damage to CHE and / or infrastructure	B	2	3
	Remove vital system data	C	2	3
	Interfere with communications	D	2	2
	Shut down OS or JVM	E	2	2
	Edit the container map	F	2	2
	Flood network devices	G	2	1
	Freeze OSGi bundles	H	3	2
Moving CHE	Inject instructions to the ICS network	I	2	3
	Exploit an OSGi vulnerability to modify data	J	3	4
Data leakage	Steal process data	K	2	3
	Capture messages between EIS and TOS or CHE CS	L	2	2
	Go to the yard to see / access the containers	M	2	1
	Exploit an OSGi vulnerability to capture data	N	3	3

From the table it is clear to see that more than half of the attack vectors result in Denial of Service. These attacks are fairly easy to carry out, but also the impacts of such attacks remain generally lower than in other attacks. Causing a CHE to collide is an exception, because collisions potentially cause significant damage to the equipment and the infrastructure, and under rare conditions even personal injuries are possible. While there are safety features in place to prevent such incidents, attacking those simultaneously would allow an attacker to achieve the objective. Although moving CHE has only two attack vectors associated to it, it is noteworthy that the impacts of these attacks are significant. Attacks resulting in data leakage have various impacts depending on the data they are targeting.

Only one attack vector with a difficulty rating of one was found. This is due to the sensible network design, which prevents direct access to the process server from the Internet. The only exception was a misconfiguration, which allows access from the Internet. Also comparing to commercial off-the-shelf products, the system is rather obscure, and therefore finding vulnerabilities requires more understanding about the context. This doesn't mean that attacks would be any more difficult for more capable attackers, it just implies that the system isn't an ideal target for casual hackers, because it requires more examination initially.

There are also no attack vectors with the highest difficulty rating of four. This is explained by the fact that there are many other threats with lower difficulty ratings, so it is easy to ignore attacks that are very difficult to carry out. Once other threats have been mitigated, the assessment can be revised with the focus on more complicated attacks to include more unlikely incidents.

It is also worth noting that attack vectors have been rated as separate attacks. Using attack vectors in conjunction with each other might lead to a more significant impact without significant increase in difficulty. For example multiple DoS attacks could be deployed simultaneously, in which case recovery would take longer. Simpler attacks can also be used as a distraction to attract the attention of system operators while carrying out a more sophisticated attack.

For easy comparison attack vectors are also presented in the form of a risk matrix in Table 4. To see which letter corresponds to which attack vector, refer to the matrix ID column of Table 3. The risk matrix has been color coded so that green signifies low risk, yellow signifies mediocre risk and red signifies high risk. The threats on the red cells should be mitigated as soon as possible, whereas the threats on the green area can be accepted, at least while the other threats are being addressed.

Table 4. *Risk matrix of the attack vectors.*

Difficulty	Impact			
	Minimal	Minor	Major	Catastrophic
Trivial		A		
Moderate	G, M	D, E, F, L	B, C, I, K	
Difficult		H	N	J
Unlikely				

4.5.3 Attack scenarios

In this sub-section, three scenarios are presented based on the attack trees. The scenarios are hypothetical only, so they do not go into great technical detail, but they will offer some insight on how cyber attacks generally proceed. Some assumptions about preconditions and the environment are made based on findings of this thesis.

Scenario 1: Denial of Service

The first scenario describes an incident, where a cyber attack is used to cause damage by causing a collision between a CHE and a container (see Figure 9 on page 55). The incident will at least render the CHE inoperable, but it is likely that the entire process is suspended to figure out why the collision happened. This attack vector utilizes various kinds of premade attack tools to make the attack easier.

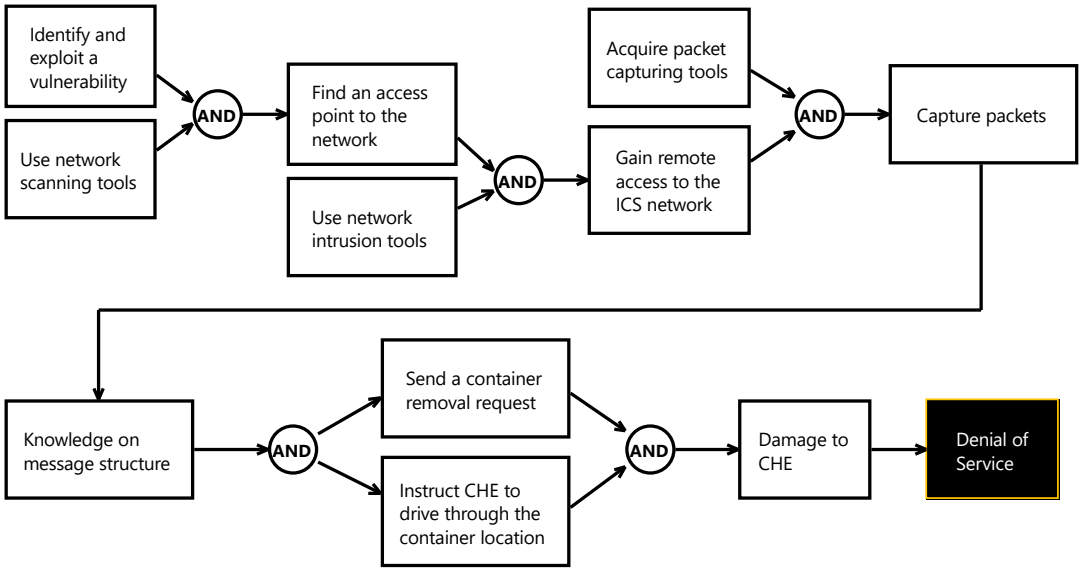

Figure 9. *An attack vector for a DoS attack*

Because the ICS network isn't accessible directly from the Internet, the attacker must first gain access to the office network, which is adjacent to the ICS network. To achieve this, the attacker uses a network scanning tool, such as Nmap, to scan access points to the office network. It is assumed that the attacker is able to guess or steal login credentials for a remote access service or find some other vulnerable point in the outer perimeter of the network. The attacker has now entered the office network.

In order to access the ICS network, the attacker needs to find a vulnerable target in the ICS network and gain control of it. The difficulty of this phase depends on how the two networks are separated from each other. If the networks are directly connected to each other, finding a target is easy, but if the networks are not connected at all, the attacker needs to take a step back and find another way to access the ICS network.

In this case, the networks are separated by a firewall. This means, that the attacker needs to either compromise the firewall, for example with malware, or try to find a way through the firewall. This usually isn't that difficult, because firewalls in ICS environments tend to be configured for accessibility, not security. Here, it is assumed that the attacker has eventually found a network intrusion tool that penetrates the firewall and is now able to capture all traffic between EIS and TOS.

By analyzing packets captured from the network, the attacker is able to figure out how containers are represented in the messages, how they are added and removed and how TOS instructs EIS to move containers. Based on this knowledge the attacker can now instruct EIS to remove containers from the container map. Once a container is removed from the container map, CHEs won't be aware of it. If CHEs relied solely on the map, they would collide with anything that isn't on the map, so the surroundings of a CHE are monitored by other safety features as well. In case any of these features fails, for example due to a software bug or another cyber attack, a collision will happen.

A less capable attacker can just randomly remove containers from the map and see what happens, but a more ambitious attacker can try to find containers that are already placed so that it is easy to cause a collision. It is even possible to instruct a CHE to move a container to an optimal location, which really would be a rather trivial task as all the required information is already available at this point. It would also be possible to trigger the incident by instructing a CHE to park on the location of a container that has been removed from the map.

Scenario 2: data leakage

In the second scenario it is assumed that an attacker wants to steal KPI data from the system (see Figure 10). The attacker is aiming for long-term presence in the system to collect data regularly, with the intent to sell the data to an industry competitor. Therefore the attacker modifies a piece of malware found online to fit the purpose. The piece of malware is designed to scan the network and look for any servers. Every time a server is found, the malware replicates itself and attempts to infect the server as well. Once a server is infected, the malware opens a remote control connection to receive further instructions from the attacker.

First, the attacker has to somehow deliver the piece of malware to the target network. To accomplish this, the attacker only needs to deliver the malware to the office network, which is adjacent to the targeted ICS network, as the malware is clever enough to get through the firewall separating the networks. So, the attacker sends the malware in an email attachment disguised as a Microsoft Word document to various email addresses that were found on the terminal operator's website and waits for the malware to start opening connections.

Eventually, after infecting some other servers in the office network, the malware finds its way to the process server of the ICS network and opens a remote control connection for the attacker. Having gained access to the process server, the attacker can now try to install a packet sniffing tool on the process server in order to capture packets going in and out of the server. Optionally, the attacker can exploit a vulnerability in a switch and use the server just as an intermediary to deliver captured packets.

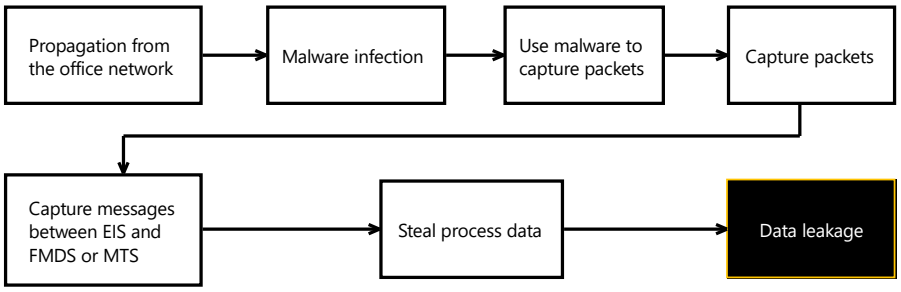

Figure 10. *An attack vector for data leakage*

The attacker can't just collect all possible packets all the time, because that would cause suspicious amounts of network traffic and load on the process server. Therefore it is necessary for the attacker to first only capture short sequences of packets to just a few IP addresses at the time. Once the attacker has figured out some characteristics of the packets containing KPI data, it is possible to start capturing the relevant packets only.

The difficulty of this attack vector is defined by three factors. First, if the malicious email is not opened or an anti-virus program detects the threat, the attack fails. Second, a firewall between the Internet and the ICS network may deny connection attempts of the malware, so a well-configured firewall would make the attack challenging. Third, a properly protected process server would make it significantly more difficult to both infect the server and to perform any further operations on it. In this case though, the process server is not regularly updated, so it is likely that the malware would be able to infect it.

Scenario 3: moving Container Handling Equipment

The third scenario describes an attack, where a container is moved by gaining control of a CHE (see Figure 11). As was discussed before, this type of attack would most likely be useful when stealing a container from the terminal. It is safe to assume that the attacker would already be in close proximity of the terminal, so this attack vector includes a physical component as well.

The purpose is to physically trespass into the premises of the terminal and install a malicious device that can capture and inject packets in the ICS network. The malicious device communicates with the attacker's laptop via a wireless connection of its own, so it is neither dependent on an Internet connection, nor affected by any firewalls or other security systems protecting the terminal operator's networks. The device could use for example Bluetooth or Wi-Fi, but if a larger range is required, the device could just as well have its own SIM card and phone number to facilitate communication over the telephone network.

Having built the malicious device, the attacker needs to access the terminal in order to install it. The attacker could for example pretend to be a Kalmar employee and ask for access to the building where the process server is located. Of course the attacker won't know beforehand where the malicious device should be plugged into, so it might be necessary to first do some network scanning disguised as maintenance operations. On the other hand it would also be possible to use methods from previous scenarios to remotely scan the network in advance. Nonetheless, it is assumed that the attacker is able find a suitable location for the malicious device to be installed.

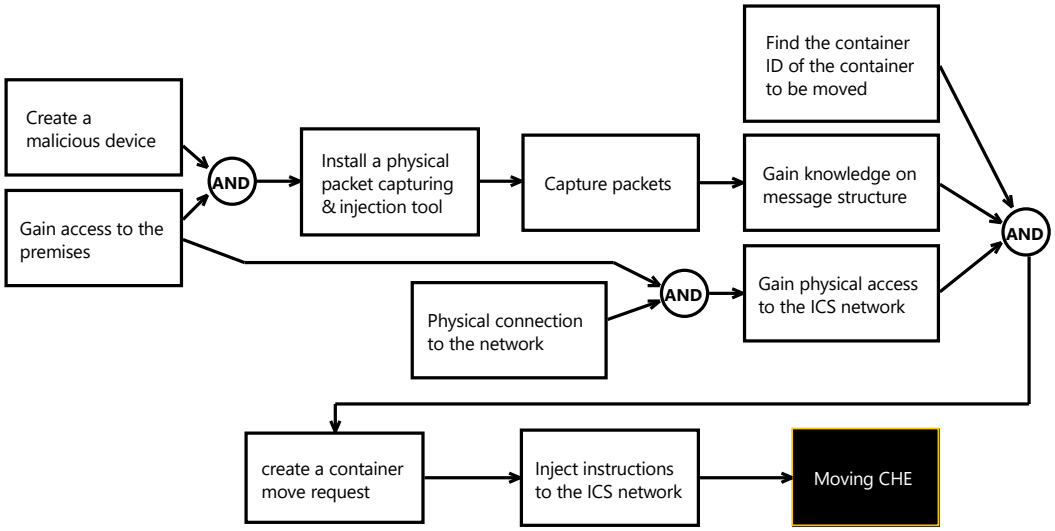

Figure 11. *An attack vector for moving Container Handling Equipment*

Next, the attacker will need to find out how the system operates. Analyzing packets captured from the network should offer enough information for the attacker to understand what is needed to form a container move request. Most importantly the attacker needs to acquire the ID of the container to be moved. The ID binds together an actual container and its digital representation. The ID can be obtained by observing messages that include data about containers. Those messages have the ID, but also another identification number that is also visible on the actual container and therefore known to the attacker as well. So finding a message with matching identification number from the actual container, the attacker is able to obtain the ID that the system uses to distinguish containers.

Once the container ID has been found, the attacker needs to craft a container move request, which at its simplest only includes the ID and coordinates of the target location. Based on that information the system finds out the current location of the container and schedules an appropriate CHE to carry out the move.

4.6 Countermeasures

In this section, suggestions are made on how to improve cyber security of Kalmar's container handling automation system in general (Sub-section 4.6.1) and cyber security of EIS in specific (Sub-section 4.6.2). General suggestions are made based on ICS-specific literature and good practices, and as such they are applicable in other ICS systems as well. EIS-specific suggestions have mostly been applied from more specific literature, which means that their application in other environments may not be straightforward.

When trying to achieve comprehensive cyber security that covers the entire ICS network, no single security solution is adequate, so individual solutions need to be applied to complement each other [20; 29]. As was discussed in Section 4.3, a defense-in-depth

approach is often recommended, and it will also be applied here. Depth of defense can be perceived on two different levels [20]. On one hand, the network itself can be divided in security zones to provide a layered structure. On the other hand, technical solutions, processes, training, policies and other measures can be viewed as layers, creating depth to the defense.

In the early days of cyber security, it was thought that not telling anyone how a system or a program works would provide sufficient security against hackers and other attackers. This kind of security-by-obscurity principle has been proven to be inefficient against modern attacks, such as reverse engineering. Therefore building security on the assumption that secrets will always remain as secrets is not recommended. For example writing passwords directly into software code (known as *hardcoding*) isn't recommended, as it won't be possible to change these passwords later if they get compromised by an attacker.

The limited space and timeframe of this thesis does not allow for an extensive look on countermeasures. Therefore it is advised to refer to the IEC standard 62443-3-1 [5], the cyber security guide by the French network and security agency (ANSSI) [99] and NIST SP 800-82 [29] for more detailed and comprehensive coverage. Additionally, references used in this chapter usually provide additional information.

4.6.1 System-wide countermeasures

Only countermeasures that provide direct or indirect benefits to the security of EIS have been covered in this section. While the system-wide measures may not protect EIS as such, they limit the amount of attack vectors that may lead to exploitation of EIS, and therefore have been included in this section. Choosing a focal point other than EIS would presumably result in a different set of countermeasures, and therefore the set of measures presented here can't be considered exhaustive when considering an entire ICS system. Still, implementing the measures presented here would be a considerable improvement and would result in an adequate baseline of security.

Secure network architecture

Practically all sources that discuss the architecture of ICS networks suggest some form of separation between the ICS network and other networks. As today's ICS networks most of the time require connections to other networks, it is not possible to isolate them completely. It is recommended though that the number of entry points to the ICS network is kept to its minimum [20; 29]. Ideally, all connections would pass through a single strictly controlled entry point [19; 29]. Limiting the amount of entry points decreases the size of attack landscape for the attacker, but also makes the network more manageable, as there are fewer devices that need to be configured. Entry points are usually protected with a firewall or a similar filtering solution [20]. According to Stouffer et al.

[29], network segregation and segmentation is among the most effective architectural concepts of ICS security.

To further strengthen the segregation between the ICS network and other networks, *a demilitarized zone* (DMZ) can be added between the networks [20; 100]. The purpose of a DMZ is to act as a buffer between the networks and eliminate all direct connections between them [100]. When using a DMZ, instead of having direct connections between devices from different networks, requests are made to the DMZ server, which then completes the requests [29; 100]. This way the two devices never actually communicate with each other, but only with the DMZ server. To filter out any malicious requests, the DMZ server can be configured to only allow through certain types of messages from certain IP addresses.

A DMZ is particularly useful when networks managed by different entities need to be interconnected. This is the case for example in maritime terminals, where typically the ICS network is owned by the system provider and the office network is owned by the terminal operator. As the container handling process relies on certain inputs from the Terminal Operating System, a connection between these networks is necessary.

The ICS network itself should also be divided into logical segments. Firewalls can be used to control communication between the network segments. For example treating the wireless part of a network as an individual segment and separating it from the wired part with a firewall would mitigate the risk of an attacker exploiting a vulnerability in a wireless device to attack a server in the wired part of the network.

In large networks, segments can consist of even smaller sub-segments that can be further divided, but ultimately they consist of individual devices that can be secured with various host-based security solutions, such as host-based firewalls, anti-malware software and Host Intrusion Detection Systems (HIDS) [19]. The most suitable solution depends on the device and should always be chosen case-specifically. VLANs can also be applied to keep traffic of one segment invisible to other segments [20], as has been done on Kalmar's solution.

Principally, connecting an ICS network to the Internet should be avoided, but if for any reason the ICS network needs to be accessed from the Internet, the connection should be authenticated and encrypted. To achieve this, most often a Virtual Private Network (VPN) solution is used [29]. On the case example of this thesis, access was only allowed from the Kalmar Intranet using VPN. This solution adds an additional layer of security as exploitation would require accessing the Intranet first. Unfortunately, the solution was rendered useless by the misconfigured firewall that, when exploited, would allow access past the VPN. Security could be improved by configuring the firewall so that it is visible to only certain IP addresses. Moving the VPN endpoint from the ICS

network to a DMZ would give better control over communication made over the VPN connection, thus providing further improvement in security.

Security monitoring

One important aspect of maintaining a secure state in a system is monitoring. Security monitoring tools are usually referred to as Security Information and Event Management (SIEM) tools. Two examples of SIEM tools are Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). Both of these systems analyze network packets to find signs of malicious actions, such as malware or exploits [19]. While IDS systems are only able to detect intrusions and generate alerts, IPS systems can also take actions to stop suspected attacks by dropping the malicious packets.

According to Knapp [19], IDS and IPS can both be installed in-line or out-of-line with other network devices. When installed in-line, the IDS or IPS stops each packet, inspects it and then lets it go. As a consequence, some delay is introduced. Installing either system out-of-line allows analysis without causing any delay as the packets are mirrored (duplicated) before inspection. As a downside, out-of-line systems can't drop any malicious packets they identify. This means that IPS systems can only utilize their full potential when installed in-line, while IDS systems do not benefit from in-line installation, so in time-critical systems they should be installed out-of-line to avoid delay.

IDS and IPS systems are best used in conjunction with a firewall, as it is able to filter packets based on their type, whereas IDS and IPS systems analyze the content of packets to allow or deny traffic [19; 20]. Using a firewall as a filter before an IDS or IPS also reduces the amount of traffic left for analysis, effectively reducing delay. IDS and IPS systems can be deployed either at the outer perimeter of the network to monitor traffic going in and out of the network, or at an individual device to monitor traffic in and out of the device [20].

IDS and IPS systems are based on a set of rules that define what kind of traffic is normal in the network, while everything else is considered malicious. As ICS networks usually are fairly static and the network traffic is limited to a small and strictly defined set of protocols, they are relatively easy in terms of configuring rulesets. Therefore, IDS and IPS systems are well suited for monitoring and securing ICS networks.

Authentication and encryption

Authentication has already been brought up on multiple occasions in this thesis, which highlights its significance to cyber security. Authentication provides the means to verify the identity of users and devices based on the credentials they provide [29]. Both human users and devices should always be authenticated to verify the claimed identity before any other communication takes place.

There are two general approaches to authentication: distributed and centralized [29]. Distributed authentication means that each system is responsible for its own authentication. Each system also stores its own user accounts and credentials, which means that users have separate accounts for each system. When authentication is centralized, all systems use the same authentication service and accounts and credentials are stored by that centralized service only.

The advantage of distributed authentication is that it doesn't require any additional infrastructure, but in large networks maintenance becomes tedious as accounts need to be created and removed separately in each system [29]. If a centralized authentication service is used, its security and constant availability must be guaranteed. Stouffer et al. [29] also note that some older products that are still used in ICS systems may not have any support for centralized authentication.

Regardless of the authentication approach, the credentials should be strong. As has been noted earlier, this was not the case in the assessed system. If passwords are used, they should be complex and be only used once. Default passwords should be changed. Using the same authentication credentials on multiple systems or devices is not recommended, unless a centralized authentication system is in use.

There are also other authentication methods that can be used instead of or in conjunction with passwords. In challenge/response authentication both entities have agreed upon a secret code beforehand. Authentication happens by one entity sending a random sequence (the challenge) and the other entity using the secret code to generate a unique sequence from the challenge (the response). Biometric authentication uses a unique biological characteristic, such as a fingerprint or facial geometry, as the credential to identify the human requesting access. A physical token, such as a smart card or a Radio Frequency Identification (RFID) tag, can also be used to store the password, meaning that it can be longer and more complex.

While authentication ensures that the endpoints are who they claim to be, encryption protects network traffic from being read or altered between endpoints. Encryption adds additional phases to sending and receiving packets, which means that latency of the network traffic is increased. The amount of latency added depends on multiple variables, including the encryption algorithm used and available processing resources, which means that the impact of individual solutions may vary. Being typically time-critical, ICS systems are susceptible to latency, and therefore encryption isn't always possible [29].

Considering EIS, most of its communication happens with other components running on the same process server, so encryption doesn't provide significant improvement there, as the data only travels short distances. EIS also handles connections that are time-critical, so applying encryption there would be problematic due to the introduced laten-

cy. Finally, EIS is connected to the Terminal Operating System and possibly to other third party systems that are located outside the ICS network. These connections usually can endure more latency, and as the endpoints are located in different networks, encryption, in conjunction with authentication, would provide an improvement in security. However, the most significant gains from encryption would be achieved in wireless networks and when accessing the network remotely from the Internet.

Change and configuration management

Over time, as configuration changes and system updates are made, the system slowly transforms. Changes are usually made to improve some aspect of the system, but if they are not traced and documented, the state of the system will slowly become unknown. Stouffer et al. [29] recommend the establishment of a formal change management program. Guidance can be found for example in NIST SP 800-100 [101] and NIST SP 800-128 [102].

As was pointed out in Section 3.5 and in Sub-section 4.3.1, patch management of an ICS network is a complicated task due to the conflict between security and availability of the system. Arguments can be made both for and against applying patches. It can be argued that updates shouldn't be made, because they introduce change to the ICS environment and therefore slowly erode the system, eventually causing it to malfunction. There is also the risk of something going wrong when applying a patch. On the other hand, leaving the system unpatched increases the attack landscape as time passes. For example, if a system has been unpatched for two years, the system is susceptible to all vulnerabilities discovered in that time.

From the security perspective application of critical patches is absolutely necessary. To mitigate the risk of unwanted process interruptions, a systematic approach to patch management is recommended [29]. The German federal energy association (Bundesverband der Energie und Wasserwirtschaft e.V., BDEV) [103] suggests five requirements for patch management in critical infrastructures:

1. The system shall allow patching of all components.
2. Patching should be possible without interrupting normal operation.
3. Preferentially, patches are installed on passive and redundant components first. Other components are updated after the switch-over process.
4. The system provider shall support a patch management process that manages testing, installation and documentation of patches. It is recommended that the staff operating the system installs the patches.
5. Installation and uninstallation of patches shall always be authorized by the system owner and must not be done automatically.

From the first three requirements it can be concluded that patch application needs to be considered already when designing the system. The first requirement suggests that all

components must be somehow accessible, so that they can be updated. The second and third requirements suggest that critical components of the system must be redundant, so that they can be updated without any interruptions to the operation. Fourth and fifth requirements suggest that there should be a patch management policy in place to ensure consistency of patch management.

To make sure updates do not break any functionality of the system, updates should be tested before they are applied to the production environment. Terminal operators typically have their own testing environments to test software and hardware, so regular testing of updates wouldn't be too difficult to facilitate. Installing patches usually requires a pause in the operation, but the OSGi framework used in EIS allows bundles being swapped without interruptions, which makes updating EIS easier.

To support controlled management, a patch management system can be used. A patch management system distributes patches to appropriate devices and monitors that all devices stay up-to-date [104]. Some systems are capable of regularly scanning for new updates and automatically distributing them to the devices, but using such features is not recommended in ICS systems, where patches need to be verified and tested before installation [104].

In ICS systems the benefit of using a patch management system is that devices do not need to be updated individually by hand. The system can also be used to ensure that all devices are running the correct software versions and to detect new devices with false software versions being added to the network. If the patch management system doesn't require any connections to the outside, it can be located in the ICS network, but it is recommended to place it in the DMZ, if there is one [29].

To improve reliability, the automation system of Kalmar has already been specified to be duplicated to a second process server [105]. This means that if the primary server fails for any reason, a manual switch-over can be performed to start running the process on the secondary server. If this redundant two-server structure was upgraded to a mirrored system with an automatic switch-over, it could also be utilized to update all software on the servers without interruptions to the automation process. This would be achieved by updating the secondary server first, then performing a switch-over and finally updating the primary server as well. Naturally, it would be necessary to test the patches first in a test environment to make sure that they don't change the automation process in any way. Otherwise the switch-over may not be successful.

Hardening

To reduce the size of the attack landscape, the software and devices should be hardened. Hardening means disabling or removing features, services or software that are not necessary for the intended operation of the system [20]. For example operating systems and firewalls used in ICS networks are generally made for a wide range of applications, so

they have plenty of functions that are not useful in a terminal automation system. These functions may have vulnerabilities or be exploited in attacks, so they should be disabled or, when possible, removed. Almost any part of a system can be hardened, but workstations, servers and networking devices are examples of the most typical items [20].

The general idea in hardening is to remove everything that isn't necessary, which is where the difficulty lies as well. To be able to harden an object properly, one must know exactly what is necessary and what isn't. Detailed documentation of the object is helpful, but over time the object might change making the hardened configuration either too tight or too loose. A too loose configuration means that unnecessary actions are allowed, giving more possibilities for attackers. On the other hand a too tight configuration blocks actions that are needed for normal operation. To help harden their products, major network device and OS manufacturers produce hardening guides with instructions on what to configure and how to do it.

Secure execution environment

Even when the ICS network and its connections have been carefully designed and all devices have been appropriately secured, it might happen that a malicious program finds its way on a PC or on the process server. To keep the most valuable data secure from the malicious program, a Trusted Execution Environment (TEE) can be established. A TEE is a secure processing environment that is isolated from the rest of the environment, sometimes called the Rich Execution Environment (REE) [106]. Isolation means that even if the malicious program infects the REE, it won't be able to access the TEE. Integrity of the TEE is verified when the system starts, and only authorized applications are allowed to access the environment to make sure that integrity is maintained [106].

Secure execution environments have been applied to different technologies where integrity of the system is important, such as smart cards [106], mobile phones [106] and cloud environments [107]. Development in smart cards and mobile devices suggests that secure execution environments could be applied in the mobile parts of the container handling process, i.e. in the CHEs. Weiqi et al. [107] present a method to create a TEE in a cloud computing environment that uses virtualization. The problem of ensuring isolation between virtual machines was also identified on the process server of Kalmar's container handling automation system, and use of TEE would provide a more reliable isolation there as well.

While some secure execution technologies already exist as complete products, not much research has been done on applying them in ICS environments. Secure execution environments seem to have the potential to produce an improvement in ICS security, but only when other parts of the network are equally secure. Securing the endpoints with TEEs won't offer a significant improvement in security if the communication channels are left insecure and susceptible to Man-in-the-Middle attacks.

Incident recovery

Even if all the recommended security measures are in place, the scenario of a security breach should still be considered. It is vital that employees know how to act in the event of a security incident in order to mitigate the consequences. An *incident response plan* is a document that describes at least responsibilities, predefined procedures, tools available and a communication plan for security incidents [108]. A specific Process Control Security Response Team (PCSRT) can also be assigned to manage response to security alerts and incidents [108].

In some cases incidents lead to loss of data and therefore backups should be made of all valuable data regularly to mitigate the risk. Such data also includes metadata, for example how devices are configured and what software versions are currently installed. Well-organized and easily recoverable backups also quicken recovery from the incident. Documenting the network structure and configurations used in devices makes incident recovery easier when broken devices or components need to be replaced. The general purpose of incident recovery plan is to re-establish the normal operation of the system as quickly and safely as possible [29].

Physical access control

The main concern regarding physical access in this case was someone with malicious intentions accessing the site and altering or damaging network devices or the process server. To mitigate this threat, access to the site should be more strictly controlled. Security-critical spaces should be identified and access to those spaces should be allowed for authorized personnel only [29]. Access control can be enforced using keys, Personal Identification Numbers (PIN), access cards or any other authentication methods discussed earlier. Access to servers and network cabinets should also be limited to maintenance personnel only [29].

Physical access control is typically the responsibility of the terminal operator, which makes management of this aspect of cyber security more difficult to manage. If the system provider wants to ensure security of its system, a contract between the provider and the operator must define the minimum level of physical access control.

Policies, contracts and training

Policies are an important part of establishing a desired level of cyber security [109]. Security policies define an organization’s approach to managing both information security and cyber security [110]. They also determine responsibilities, such as who is responsible for the security of the ICS network [111]. Organizations usually already have policies regarding IT security, and ICS security policies should be integrated with them where possible [29]. It should be noted though that not all organization-wide policies are directly applicable in ICS environments, so ICS-specific policies are required [20].

Kilman & Stamp [109] introduce a framework that presents a hierarchical classification of security policies, which covers different aspects of ICS security and what security policies should be created to cover those aspects. The authors use the term SCADA, but their definition of SCADA is actually closer to the definition of ICS as it was defined in this thesis. A redrawn version of the framework, where the term SCADA has been replaced with ICS to avoid confusion, is presented in Figure 12.

At the highest level of the framework is the system security program, which defines the context for all the other policies. It provides information about the ICS environment, but rarely contains any policy statements [109]. The different policies are divided into nine groups that represent different viewpoints. The reasoning behind this division is that multiple entities can work on different policies simultaneously with as little overlap as possible [109]. For more information on the categories and the policies included in them, refer to the report of Kilman & Stamp [109].

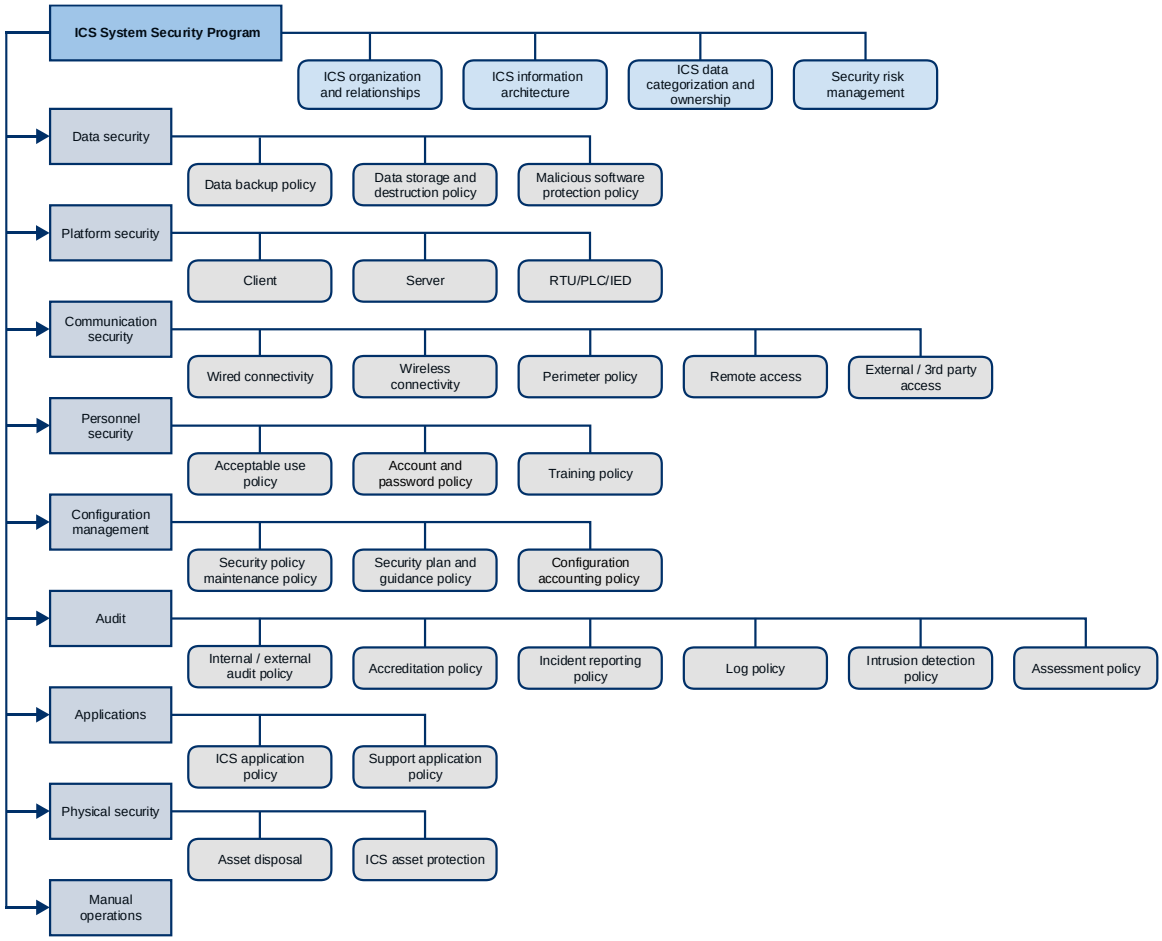

Figure 12. *ICS system security program framework. Adapted from Kilman & Stamp [109].*

While the framework may not be suitable for every organization's needs as such, it can be used as a reference to ensure that all aspects of ICS security have been considered. In some environments not all of the policies may be relevant, in which case they can be omitted. To complement the policies, procedures and guidelines can be used. Proce-

dures are detailed sequences of steps that need to be taken in order to provide a certain security measure [111]. Guidelines are suggestions on how to get something done [111]. The main difference between guidelines and procedures is that guidelines are not meant to be mandatory.

While policies define the internal conventions of an organization, in some cases it is necessary to establish similar rules with external entities as well. For example a container terminal typically has at least two actors: the system provider and the terminal operator. To create a secure environment, these actors must cooperate and divide responsibilities between each other. To make such agreements binding, they should be included in contracts or Service-Level Agreements (SLA).

It is particularly important to agree on which actor is responsible for which aspect of cyber security, in order to avoid confusion, overlap or gaps in security. Sanctions should also be set in the case that one of the actors neglects its responsibilities. From the system provider's viewpoint one option is to deny all responsibility. While this may seem intriguing at first, there are two problems in this approach. First, the system provider may have a legal obligation to bear some of the responsibility. Second, as customers become more security-aware, they will start demanding for a more security-minded approach.

Instead of denying all responsibility, it is recommended for a system provider to have premade contract templates that cover different levels of security and defined requirements on how to achieve these levels. The increasing security-awareness of customers should be viewed as a new business opportunity as security features can be sold as additional products or services. Security embedded in the software and system design can also be used as selling points in marketing.

For the customer it might be difficult to address cyber security in contract negotiations, if the organization doesn't possess the required security knowledge. To help in such situations, Department of Homeland Security (DHS) has produced a procurement language for cyber security in ICS systems [112]. This document offers insight on what should be considered when procuring an ICS system and how to formally express these requirements in a contract or in an SLA. The document is equally useful for system providers, as it can be used as reference when designing security features.

Implementation of a security program typically requires changes on how employees carry out their daily tasks [29]. To ensure that the employees understand why such changes are made and actually obey the new policies, security awareness must be raised and skills need to be developed. To achieve this, a NIST Special Publication [113] suggests the use of an awareness and training program, including four phases:

1. Awareness and training program design.
2. Awareness and training program material development.

3. Program implementation.
4. Post-implementation.

According to the Special Publication [113], in the program design phase the awareness and training needs are identified and an awareness and training plan is designed accordingly. Then, in the second phase, awareness and training material is produced to reflect the needs identified in the first phase. The third phase is where the actual awareness raising and training takes place. In the fourth phase success of the program is monitored and feedback is collected to improve the program. The Special Publication [113] remarks that even a small awareness campaign can make a big difference in the security posture of an organization.

Stouffer et al. [29] note that in an ICS environment a general IT awareness and training program is not adequate. Training must cover ICS-specific applications and the physical process controlled by the control system [29]. Training programs also demonstrate management's commitment to the security program [29].

4.6.2 EIS-specific countermeasures

Above, basic security fundamentals concerning IACS systems have been covered. In this sub-section security measures that are specific to EIS and the process server are discussed. Different building blocks of the system have been considered separately and concrete suggestions on improvement are given.

OSGi

In Sub-section 4.3.1, it was concluded that malicious bundles are the main concern of OSGi security. To control installation and resource usage of bundles, the OSGi framework has a *security layer* [74]. The security layer, use of which is optional, offers two security features: code authentication and permission control. Code authentication is achieved by signing the bundles (technically JAR-files) with a digital signature [74]. When only bundles signed by authorized entities are allowed to be executed in the OSGi environment, malicious bundles created by hackers are blocked. In the example case of this thesis all the bundles would be signed by Kalmar and all unsigned bundles or bundles signed by other entities would be blocked.

In addition to verifying the origin of the bundle, a digital signature also ensures integrity of the bundle [74]. This means that if the bundle has been modified after it has been signed (maliciously or due to an error), the signature check will fail and the bundle won't be executed. The signature is checked only when the bundle is started, so implementing digital signatures won't affect the runtime performance of the system. The signing process has been described in the OSGi specification [74].

While bundle authentication happens before bundles are started, permission control happens during execution. Permissions define what resources a bundle is allowed to access and what operations it can perform [74]. As with permissions in general, it is advised to limit the amount of permissions to its minimum. This way, even if a malicious bundle does get installed, it only has a limited access to resources.

In recent years, the Internet of Things (IoT) and cloud computing seem to have created a push on developing methods for reliable isolation of applications in virtualization environments, and various solutions have been suggested [79; 114; 115]. Similar concept could be applied to OSGi in order to prevent malicious bundles from interfering with legitimate bundles, but it seems that solutions directly applicable to ICS systems are not yet available.

Virtualization

There isn't much that can be done to improve the security of JVM apart from improving security of OSGi, which minimizes the possibility of a malicious actor escaping the JVM virtualization sandbox. Improving JVM security isn't necessarily needed either: if the OSGi environment inside the JVM has been secured and vSphere is able keep the JVM instances separated, it shouldn't be possible to find any exploits.

The vSphere platform makes several attempts to ensure security of the environment on virtualization layer, on individual virtual machines and on virtual networking layer [116]. These security features cover secure processing and storage of data inside a VM, secure internal communication between VMs and secure external communication in and out of the process server.

As vSphere in itself is a relatively large and complex system, it has its own security vulnerabilities and exploits [83]. The best way to cope with its vulnerabilities is to keep the software up-to-date and to harden the vSphere configuration. The vSphere platform has been designed to support a wide variety of use cases, so in most cases some unnecessary features need to be disabled. To achieve this and other security improvements, VMware has produced several documents [116-118] that offer hardening guidelines.

ActiveMQ

As was discussed in Sub-section 4.3.1, ActiveMQ offers the option of authenticating JMS clients. Authentication prevents attacks where an attacker sets up a new client to access message queues and topics of ActiveMQ. As Kalra [81] notes, implementing authentication is fairly simple, as the authentication schemes are plug-in based and require hardly any changes to the JMS client code.

ActiveMQ supports three authentication schemes: default, simple and JAAS-based [81]. The default authentication actually means no authentication at all. Connections from

any JMS client are accepted without any username and password, granting full access for anyone within the network. When using simple authentication, ActiveMQ doesn't accept connections without a username and a password. Kalra [81] claims that usernames and passwords for simple authentication are written to the configuration file as plaintext, but since the version 5.4.1 of ActiveMQ it is now possible to encrypt passwords [119]. The authorization credentials are still stored in the ActiveMQ configuration file, so maintenance of the user base requires accessing the configuration file.

The third scheme is based on Java Authentication and Authorization Service (JAAS). The JAAS plug-in allows implementation of custom authentication methods [81], which in practice would mean leveraging Active Directory (AD), Lightweight Directory Access Protocol (LDAP) or some other database dedicated to authentication. The benefit of using this scheme is that it frees ActiveMQ from the responsibility of managing authentication and allows for centralized management of authentication credentials directly from the authentication system. Usually an authentication database is already present for other purposes, so implementing JAAS-based authentication would only be the matter of writing the plug-in and adjusting the JMS clients to support the feature.

Without authorization all authenticated users have unlimited access to all resources, when clients should only have access to resources that they actually need. ActiveMQ offers authorization on destination level and on message level. Destination level authorization requires clients to have certain privileges in order to access a topic or a queue, whereas message level authorization controls access to individual messages in a queue or a topic [81]. Message level authorization provides fine-grained access control, but if it isn't actually necessary, the simpler destination level authorization is recommended, as it is easier to configure and maintain.

It is also possible to encrypt all messages by using Secure Sockets Layer (SSL) encryption protocol that is natively supported by ActiveMQ [120]. When using SSL, both endpoints are first authenticated with digital certificates and from then on all communication is encrypted to protect it from Man-in-the-Middle attacks. As was already discussed in the previous sub-section, EIS would mainly benefit from encrypting communication with TOS or other third party systems that reside in other networks. The downside of SSL is the same as it is with all other encryption protocols: establishing a secure connection takes more time and generates more network traffic. If the performance and stability of the system are not diminished by the use of SSL, improvement in security can be gained.

Operating System

The OS lies beneath all other software, so vulnerabilities in the OS threaten integrity of the entire system and it is therefore important to keep the OS updated with latest securi-

ty patches. Hardening and controlled change management, as they were described in Sub-Section 4.6.1, are important in OS security.

Security of OS can also be improved by using an Anti-Virus (AV) software. Ideally, virus definitions of the AV should be updated at least daily [29; 121]. The reason for the short update interval is the way an AV works: it compares files in the system with the database of virus definitions, also called signatures [121]. AV vendors constantly produce new signatures as they discover new malware. If the AV hasn't been updated with these latest signatures, it won't be able to detect the latest threats either [121]. In this case there already is an AV product in place, but according to documentation is not updated regularly. If it is not possible to update the AV signatures during production, updates should be scheduled to be performed during breaks.

Software design

In this chapter, the defense-in-depth approach has been applied on the outer perimeter of the ICS network, inside the ICS network, on the process server and other devices, on the processor level of the process server and on the software framework level. This is about as far as it is possible to go without considering security of the source code of the software itself. Therefore, the final countermeasure covered in this thesis is secure programming and how to avoid writing vulnerable code in the first place.

In an effort to improve the quality of Java code and to limit vulnerabilities, various entities have created guidelines or sets of rules that help programmers to avoid common mistakes or oversights. While McGraw & Felten [122] listed 12 rules to develop more secure code, Long et al. [123] took a more comprehensive approach giving 75 recommendations. Oracle [124] also has its own guidelines that are based on eight fundamentals:

1. Prefer to have obviously no flaws than no obvious flaws.
2. Design Application Programming Interfaces (API) to avoid security concerns.
3. Avoid duplication of code and data.
4. Restrict privileges.
5. Establish trust boundaries.
6. Minimize the number of permission checks.
7. Encapsulate.
8. Document security-related information.

In the scope of this thesis it is not possible to present all the guidelines and their justifications, but the guides of McGraw & Felten and Oracle are publicly available, while the book of Long et al. is available for purchase (see References).

To help programmers to verify the security of their code, it is possible to conduct code reviews, where a professional reviews the code to identify any flaws. This of course is a time consuming and expensive task, so there are also automated tools that attempt to do

the same. As the studies of Ware & Fox [84] and Díaz & Bermejo [125] show, current static analysis tools aren't perfect, but they are capable of identifying vulnerabilities.

While Ware & Fox [84] studied eight tools using specialized test cases written in Java, Díaz & Bermejo [125] present a similar test, but using nine tools and C programming language. Most of the tools tested by Díaz & Bermejo are also capable of analyzing Java code. The results of both studies are similar: Static analysis tools are capable of identifying vulnerabilities, but only to a certain extent. They aren't yet sophisticated enough to replace manual code review, but they can certainly be used in conjunction with it. Both studies also note that the detection rate can be improved slightly by using more than one tool simultaneously, as the tools identify different sets of flaws.

As concluded by the studies, manual code review can't be replaced by static analysis tools, so secure programming knowledge is needed. This can be acquired either by using external consultation or by training of employees. There are various secure programmer certificates, such as GIAC Secure Software Programmer (GSSP) [126], EC-Council Certified Secure Programmer (ECSP) [127] and Certified Professional for Secure Software Engineering (CPSSE) [128]. GSSP and ECSP both have Java-specific versions, whereas CPSSE takes a general approach without focusing on a single programming language. All these certification programs aim at training programmers to understand how secure code is made and how vulnerabilities and flaws are identified.

4.7 Summary

In this section the results found in Chapters 3 and 4 are summarized. The research problems considered in this thesis were as follows:

1. What kind of cyber threats exist in terminal automation systems?
2. What kind of cyber threats does the External Interface Service face?
    a. What can be achieved by attacking EIS?
    b. How can EIS be compromised?
    c. How can cyber security of EIS be improved?

To answer the first research problem, 10 different categories of threats were identified. They represent the attack landscape a typical terminal automation system faces. Some of the threats are similar to office networks, but there are also threats that are specific to ICS systems and terminal automation. While the attack methods themselves are mostly similar to the ones used in anywhere else, the consequences are usually unique to the terminal automation environment. Especially physical consequences are characteristic to terminal automation. The threat categories are summarized in Figure 13 on page 74.

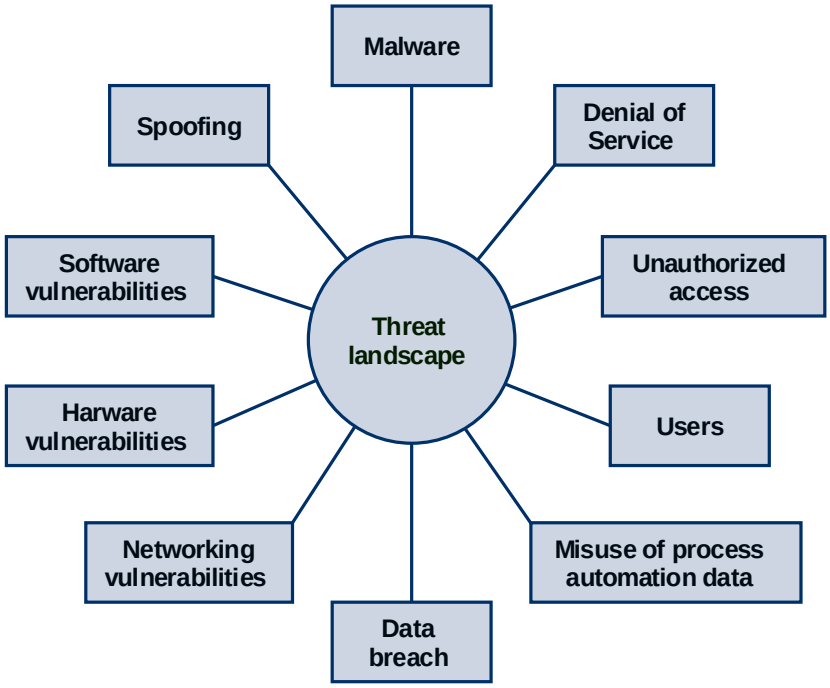

Figure 13. *Threat landscape of a terminal automation system.*

To find an answer to the second research problem, attack trees were constructed. In order to find the roots of those trees, it was necessary to recognize motivations for attacks by analyzing the environment and the process itself. Four motivations for attacks were identified:

1. interruption of process,
2. stealing process-related data,
3. stealing a container,
4. moving a container through the terminal undetected.

These four motivations also reveal what can be achieved by attacking EIS, as it was verified that EIS offers attack vectors for all of these motivations. The motivations recognized in this study are supported by a previous study by CyberKeel [38], where similar motivations were identified.

Once the motivations had been found, attack trees were constructed to explore how EIS can be compromised. The four attack motivations were mapped to three root goals for attack trees:

1. Denial of Service (DoS),
2. data leakage,
3. moving Container Handling Equipment (CHE).

While the motivations 1, 2 and 3 are directly mapped to root goals 1, 2 and 3, the motivation 4 is mapped to the root goal 3 as well. Moving a container through the terminal

undetected requires a far more complicated attack that the other motivations and it is not possible to carry that out by exploiting EIS only. In such attack EIS would be used to move containers on the container yard, so from the viewpoint of EIS the root goal would be the same as when stealing a container.

Once the root goals had been established, the attack trees were constructed by analyzing vulnerabilities of the system to reveal all possible attack vectors. The number of different attack vectors was very high, so some reduction was necessary. The attack vectors were grouped into 14 vectors that have clearly distinctive characteristics. The attack vectors were rated based on their difficulty and expected impact, and based on the rating the vectors were divided into three risk categories (see Table 5).

Table 5 shows that there are six of both high-risk and mediocre-risk attack vectors, whereas the number of low-risk attack vectors is only three. This suggests that there is room for improvement in cyber security of the system. In particular, three problem areas were identified: susceptibility to Denial-of-Service attacks, accessibility of human-readable communication and vulnerability of OSGi bundles.

In order to improve cyber security of EIS, different cyber security technologies, methods and principles were explored and their applicability to an IACS environment was considered. Table 6 summarizes the measures that were found to be suitable, how they improve security and which attack vectors they mitigate. As the number of attack vectors turned out to be high, it was necessary to find measures that mitigate as many attacks as possible, since it simply isn't feasible to mitigate the threats one by one.

Table 5. *Prioritization of attack vectors for mitigation.*

Attack vector	ID	Level of risk
Exploit an OSGi vulnerability to modify data	J	High risk
Damage to CHE and / or infrastructure	B	
Remove vital system data	C	
Inject instructions to the ICS network	I	
Steal process data	K	
Exploit an OSGi vulnerability to capture data	N	Mediocre risk
Flood devices accessible from the Internet	A	
Interfere with communications	D	
Shut down OS or JVM	E	
Edit the container map	F	
Capture messages between EIS and TOS or CHE CS	L	
Freeze OSGi bundles	H	Low risk
Flood network devices	G	
Go to the yard to see / access the containers	M	

In practice that means making changes to the fundamentals of how the system is designed, implemented and maintained, as opposed to retrofitting technical solutions into

the existing system design. Some of the countermeasures mitigate the risk associated to all of the attack vectors and some only a subset of vectors. For some of the countermeasures it is not possible to define attack vectors that they mitigate, as the improvement in security is indirect. For example a good cyber security program improves the skills and knowledge of employees in general, but how it improves the security of the system depends on the contents and objectives of the program.

Table 6. *Suggested measures to improve cyber security.*

Scope	Countermeasure	Improvement in	Associated attack vectors
System-wide	Network design (DMZ, network segregation, VPN, VLAN)	Access control, network security	All
	Avoiding obscurity	Transparency, maintenance	All
	Authentication	Access control, network security	All
	Encryption	Data security, access control, data integrity	B, D, F, I, K, L
	Monitoring (SIEM, IDS, IPS)	Intrusion detection, response time, recovery time	Indirect
	Physical access control	Access control, network security	All except A
	Secure execution environment	Data security & integrity	D, H, J, N
	Regular backups	Recovery time	Indirect
	Patch management	Fewer vulnerabilities	All except M
	Hardening and secure configurations	Fewer vulnerabilities	All except M
	User policies	Security awareness, fewer user errors	Indirect
	Cyber security in contracts	Clear responsibilities and roles	Indirect
	Security awareness program	Security awareness	Indirect
	Incident response plan	Response time, recovery time, responsibilities	Indirect
	Change control process	System stability, integrity	Indirect
EIS-specific	Digital signatures (OSGi)	Authenticity and integrity of bundles	H, J, N
	Permission control (OSGi)	Data security, access control	H, J, N
	vSphere hardening (virtualization)	Fewer vulnerabilities	C, E, H, J, K, N
	Client authentication and authorization (ActiveMQ)	Data security, access control	B, D, F, G, I, K, L
	SSL encryption (ActiveMQ)	Data security	B, D, F, G, I, K, L
	Regular updates of system and AV (OS)	Fewer vulnerabilities	Indirect
	Hardening (OS)	Fewer vulnerabilities	Indirect
	Secure programming practices	Fewer vulnerabilities, quality of code	H, J, N
	Secure programmer certificates	Quality of code, credibility	H, J, N

Table 6 offers a wide variety of different security measures. It is by no means necessary to implement all of the measures, but to study and test which solutions are functional and easily applicable. Finding a combination of solutions that complement each other

and cover the entire system is much more important than the sheer amount of measures used. Comprehensive security is created through quality, not through quantity. It is also important to work with all parties involved to achieve the optimal balance between security, reliability and usability.

5. CONCLUSIONS

The results of this thesis suggest that work on cyber security of terminal automation systems is in the beginning, as it is on ICS systems in general. Until recently, cyber security has not been a major concern during design of ICS systems, but as the number of cyber attacks is increasing and the tools used in attacks are improving, it is clear that the threat of a cyber attack has quickly become significant.

Most of the threats identified in this thesis are typical for ICS systems in the sense that they extend beyond the cyberspace and into the physical world. The threats were found to be focused on three problem areas: susceptibility to Denial-of-Service attacks, communication in unencrypted, human-readable formats and vulnerability of OSGi bundles. These issues are characteristic to systems with limited computing resources and little tolerance for delays or packet loss.

The conflict between cyber security, usability of the system and real-time requirements has been widely discussed on the field of ICS security and the same issue was identified in this thesis as well. The conflict seems to be one of the most often used justifications for not implementing security, so overcoming this issue would be a notable step forward. Part of the solution would be improvement on the performance of cyber security products. For example encryption protocols that are light enough to operate on a performance-limited PLC without significant delay are not available yet.

Another part of the solution would be improving the security aspect of the ICS components themselves. For example PLCs are often criticized by security professionals as some of them lack critical security functionality. Even the most secure components do not provide security if the system isn't well-designed and therefore the ultimate responsibility is in the hands of the system designers. In the life-cycle of a terminal automation system, security should be implemented as early as possible. The later in the life-cycle implementation happens, the harder it will be and consequently the costs go up.

Designing a terminal automation system usually is a joint venture between the terminal operator and multiple system or component providers. Division of responsibilities between parties is not implicit, and if responsibility for security has not been explicitly assigned, gaps in security can be expected. Even though this thesis has been written from the viewpoint of a system provider, the security measures presented shouldn't necessarily be the responsibility of the provider. What's important is that all parties involved are committed to providing a secure system and are able to collaborate in order to achieve that target.

This thesis covers a variety of different measures to improve cyber security. While from the security viewpoint it would be optimal to implement as many of the measures as possible, in reality it just isn't feasible because of the added cost and complexity. Instead, it is recommended to choose a few items that are relatively simple to implement and provide a good baseline for further security improvements.

In many ways introducing cyber security to terminal automation systems can be compared to implementing physical safety to the same systems. Both security and safety require compromises on efficiency and ease of use, and if either of them fails, consequences can be significant. It is typical for security incidents in terminal automation environments to have physical consequences, so it would make sense to develop both safety and security to complement each other. Currently, maturity of safety features is a long way ahead of security features, so to develop security, it would be useful to see what kind of steps were taken to establish the current status of safety. Implementing high-quality safety solutions has become an industry standard and cyber security should be brought up to that same level.

While the maritime sector has not been very active on the cyber security front, there has been significant advancement made for example in the sectors of gas and oil delivery, nuclear power and wastewater systems. These sectors are important components of the critical infrastructure and therefore they have been strictly regulated and supervised in terms of cyber security. Once these sectors have been brought up to a high level of security, the focus is likely to shift to other components of critical infrastructure, such as transport.

Some instances, such as GAO [16] and ENISA [36] have already stated that more regulation and collaboration is required to improve cyber security on the maritime sector. A recent report also raises the issue of cyber threats in transportation sector in general [129]. This suggests that in the near future stricter requirements will also be set for security of terminal automation systems and the ability to meet or exceed these requirements sooner than competitors can be seen as a competitive advantage.

REFERENCES

[1] R. Langner, To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve, 2013, 36 p. Available: http://www.langner.com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge.pdf.

[2] T. Bateman, Police Warning After Drug Traffickers' Cyber-attack, BBC News, web page. Available (accessed on 2 April, 2015): http://www.bbc.com/news/world-europe-24539417.

[3] Bundesamt für Sicherheit in der Informationstechnik (BSI), Die Lage der IT-Sicherheit in Deutschland 2014, 44 p. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2014.pdf.

[4] E. Luiijf, Understanding Cyber Threats and Vulnerabilities, in: J. Lopez, R. Setola, S.D. Wolthusen (ed.), Critical Infrastructure Protection, Springer Berlin Heidelberg, 2012, pp. 52-67.

[5] Industrial Communication Networks. Network and System Security. Part 3-1: Security Technologies for Industrial Automation and Control Systems, Suomen Standardoimisliitto SFS, IEC/TR 62443-3-1:fi, Helsinki, 2012, pp. 91.

[6] R. von Solms, J. van Niekerk, From Information Security to Cyber Security, Computers & Security, Vol. 38, 2013, pp. 97-102.

[7] T. Dumitras, Understanding the Vulnerability Lifecycle for Risk Assessment and Defense Against Sophisticated Cyber Attacks, Advances in Information Security, Vol. 56, 2015, pp. 265-285.

[8] R. Anderson, Security Engineering, Second edition ed. Wiley, 2008, 1080 p.

[9] CERT-UK, The Information Security Arm of GCHQ, Common Cyber Attacks: Reducing The Impact, 2015, 17 p. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/400106/Common_Cyber_Attacks-Reducing_The_Impact.pdf.

[10] E.M. Hutchins, M. Cloppert J., R.M. Amin, Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains, Proceedings of 6th Annual International Conference on Information Warfare and Security, 17-18 March 2011, Academic Publishing International Limited, Reading, UK, pp. 14.

[11] The MITRE Corporation, Active Defense Strategy for Cyber, 2012, 2 p. Available: http://www.mitre.org/sites/default/files/publications/active_defense_strategy.pdf.

[12] L. Myers, Cyber Kill Chain is a Great Idea, But is It Something Your Company Can Implement? Infosec Institute, web page. Available (accessed on 13 April, 2015): http://resources.infosecinstitute.com/cyber-kill-chain-is-a-great-idea-but-is-it-something-your-company-can-implement/.

[13] Kaspersky Lab, IT Security Risks Survey 2014: A Business Approach to Managing Data Security Threats, 2014, 25 p. Available: http://media.kaspersky.com/en/IT_Security_Risks_Survey_2014_Global_report.pdf.

[14] Kaspersky Lab, Cyber Threats to ICS Systems, Industrial Security 2014, 4 p. Available: http://media.kaspersky.com/en/business-security/critical-infrastructure-protection/Cyber_A4_Leaflet_eng_web.pdf.

[15] J. Gardiner, M. Cova, S. Nagaraja, Command & Control: Understanding, Denying and Detecting, 2014, 38 p. Available: http://c2report.org/report.pdf.

[16] United States Government Accountability Office, Maritime Critical Infrastructure Protection, GAO-14-459, 2014, 54 p. Available: http://www.gao.gov/assets/670/663828.pdf.

[17] Cyber Edge Group, 2014 Cyber Threat Defense Report, 2015, 38 p. Available: http://www.brightcloud.com/pdf/CyberEdge-2014-CDR.pdf.

[18] T. Macaulay, B. Singer, Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, and SIS, Auerbach Publications, 2012, 203 p.

[19] E.D. Knapp, Industrial Network Security, 1st ed. Syngress, 2011, 360 p.

[20] Suomen Automaatioseuran Turvallisuusjaosto, Teollisuusautomaation tietoturva, Suomen Automaatioseura ry., Helsinki, 2005, 160 p.

[21] M. Hentea, Improving Security for SCADA Control Systems, Interdisciplinary Journal of Information, Knowledge, and Management, Vol. 3, 2008, pp. 73-86.

[22] P.A.S. Ralston, J.H. Graham, J.L. Hieb, Cyber Security Risk Assessment for SCADA and DCS Networks, ISA transactions, Vol. 46, No. 4, 2007, pp. 583-594.

[23] E. Schweigert, SCADA Security Basics: Why are PLCs so Insecure? Tofino Security, web page. Available (accessed on 17 July, 2015): https://www.tofinosecurity.com/blog/scada-security-basics-why-are-plcs-so-insecure.

[24] P. Ahonen, TITAN-käsikirja: VTT:n päätuloksia Tekesin Turvallisuusohjelman TITAN-projektissa, Valtion Teknillinen Tutkimuskeskus, Helsinki, Finland, 2010, 152 p.

[25] F-Secure, Threat Description: Havex A, web page. Available (accessed on 11 March, 2015): https://www.f-secure.com/v-descs/backdoor_w32_havex.shtml.

[26] BAE Systems Applied Intelligence, PIANOS - Protecting Information About Networks, the Organisation and Its Systems, 2014, 54 p. Available: http://www.cpni.gov.uk/Documents/Publications/2014/2014-04-23-pianos_report.pdf.

[27] Information Technology - Security Techniques - Evaluation Criteria for IT Security - Part 1: Introduction and General Model, ISO/IEC 15408-1:1999, Geneve, Switzerland, 1999, pp. 62.

[28] ICS-CERT, Standards and References, web page. Available (accessed on 1 April, 2015): https://ics-cert.us-cert.gov/Standards-and-References.

[29] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, A. Hahn, Guide to Industrial Control Systems (ICS) Security, SP 800-82r2, National Institute of Standards and Technology (NIST), 2015, 247 p. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r2.pdf.

[30] IEEE Standards Association, P1711 - Standard for a Cryptographic Protocol for Cyber Security of Substation Serial Links, web page. Available (accessed on 1 April, 2015): http://standards.ieee.org/develop/project/1711.html.

[31] A. Pauna, Certification of Cyber Security Skills of ICS/SCADA Professionals, 2015, 46 p. Available: https://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-infrastructure-and-services/scada-industrial-control-systems/certification-of-cyber-security-skills-of-ics-scada-professionals.

[32] A. Grange, Ports at Risk of Cyber Attacks, Port Finance International, web page. Available (accessed on 2 April, 2015): http://www.portfinanceinternational.com/features/item/1629-ports-at-risk-of-cyber-attacks-by-aidan-grange.

[33] R. Langner, Robust Control System Networks: How to Achieve Reliable Control After Stuxnet, Momentum Press, 2012, 222 p.

[34] Centre for the Protection of National Infrastructure (CPNI), Cyber Security, web page. Available (accessed on 24 June, 2015): http://www.cpni.gov.uk/advice/cyber/.

[35] ICS-CERT, Information Products, web page. Available (accessed on 24 June, 2015): https://ics-cert.us-cert.gov/Information-Products.

[36] European Union Network and Information Security Agency (ENISA), Analysis of Cyber Security Aspects in the Maritime Sector, 2011, 31 p. Available: http://www.enisa.europa.eu/activities/Resilience-and-CIIP/critical-infrastructure-and-services/dependencies-of-maritime-transport-to-icts/cyber-security-aspects-in-the-maritime-sector-1.

[37] J. Kramek, The Critical Infrastructure Gap: U.S. Port Facilities and Cyber Vulnerabilities, Federal Executive Fellows Policy Papers 16, Brookings, 2013, 50 p. Available: http://www.brookings.edu/research/papers/2013/07/03-cyber-ports-security-kramek.

[38] CyberKeel, Maritime Cyber-Risks, 2014, 26 p. Available: http://www.cyberkeel.com/images/pdf-files/Whitepaper.pdf.

[39] B. Schneier, Secrets and Lies: Digital Security in a Networked World, Wiley Publishing Inc., Indianapolis, Indiana, 2000, 432 p.

[40] Symantec Security Response, Dragonfly: Western Energy Companies Under Sabotage Threat, Symantec, web page. Available (accessed on 1 June, 2015): http://www.symantec.com/connect/blogs/dragonfly-western-energy-companies-under-sabotage-threat.

[41] GFI Software, The Corporate Threat Posed by Email Trojans, 2011, 8 p. Available: https://www.gfi.com/whitepapers/network-protection-against-trojans.pdf.

[42] W. Gragido, Lions at the Watering Hole – The "VOHO" Affair, RSA Security LLC, web page. Available (accessed on 21 April, 2015): https://blogs.rsa.com/lions-at-the-watering-hole-the-voho-affair/.

[43] Centre for the Protection of National Infrastructure (CPNI), Response to Distributed Denial of Service (DDOS) Attacks, Technical Note 06/02, 2002, 6 p. Available: http://www.cpni.gov.uk/documents/publications/2002/2002005-tn0602_ddos_attacks.pdf.

[44] M. Jensen, N. Gruschka, N. Luttenberger, The Impact of Flooding Attacks on Network-based Services, Third International Conference on Availability, Reliability and Security, ARES 08, 4-7 March 2008, IEEE Computer Security, pp. 509-513.

[45] P. Roberts, Microsoft.com Falls to DOS Attack, web page. Available (accessed on 16 March, 2015): http://www.computerworld.com/article/2571559/malware-vulnerabilities/microsoft-com-falls-to-dos-attack.html.

[46] C. Won, J.H. Youn, H. Ali, Impact of High-Mobility Radio Jamming in Large-Scale Wireless Sensor Networks, Lecture Notes in Computer Science, Vol. 4097, 2006, pp. 244-251.

[47] N. DuPaul, Spoofing Attack: IP, DNS & ARP, Veracode, web page. Available (accessed on 18 March, 2015): http://www.veracode.com/security/spoofing-attack.

[48] V. Ramachandran, S. Nandi, Detecting ARP Spoofing: An Active Technique, Lecture Notes in Computer Science, Vol. 3803, 2005, pp. 239-250.

[49] E. Bertino, Policies, Access Control, and Formal Methods, in: S.K. Das, K. Kant, N. Zhang (ed.), Handbook on Securing Cyber-Physical Critical Infrastructure, Elsevier Inc., 2012, pp. 573-594.

[50] G. Hoglund, G. McGraw, Exploiting Software: How to Break Code, Addison-Wesley Professional, 2004, 512 p.

[51] Centre for the Protection of National Infrastructure (CPNI), Configuring & Managing Remote Access for Industrial Control Systems, 2010, 67 p. Available: https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/Recommended_Practice-Remote_Access_1-6-2011.pdf.

[52] K. Scarfone, P. Hoffman, M. Souppaya, Guide to Enterprise Telework and Remote Access Security, SP 800-46r1, National Institute of Standards and Technology (NIST), 2009, 46 p. Available: http://csrc.nist.gov/publications/nistpubs/800-46-rev1/sp800-46r1.pdf.

[53] S. Christey, Top 25 Most Dangerous Software Errors, 2011, 41 p. Available: http://cwe.mitre.org/top25/archive/2011/2011_cwe_sans_top25.pdf.

[54] J. Nelissen, Buffer Overflows for Dummies, 2002, 31 p. Available: http://www.sans.org/reading-room/whitepapers/threats/buffer-overflows-dummies-481.

[55] C. Sheppard, Buffer Overflows and Application Security, 2004, 12 p. Available: https://cyber-defense.sans.org/resources/papers/gsec/buffer-overflows-application-security-105917.

[56] E.J. Chikofsky, J.H. Cross, Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software, Vol. 7, No. 1, 1990, pp. 13-17.

[57] S. Bak, K. Manamcheri, S. Mitra, M. Caccamo, Sandboxing Controllers for Cyber-Physical Systems, IEEE/ACM International Conference on Cyber-Physical Systems (ICCPS), 12-14 April, 2011, IEEE, pp. 3-12.

[58] D.G. Peterson, Offensive Cyber Weapons: Construction, Development, and Employment, Journal of Strategic Studies, Vol. 36, No. 1, 2013, pp. 120-124.

[59] G. Bloom, E. Leontie, B. Narahari, R. Simha, Chapter 12 - Hardware and Security: Vulnerabilities and Solutions, in: S.K. Das, K. Kant, N. Zhang (ed.), Handbook on Securing Cyber-Physical Critical Infrastructure, Elsevier Inc., 2012, pp. 305-331.

[60] E. Byres, K. Savage, J. Karsch, J. Carter, Firewall Deployment for SCADA and Process Control Networks: Good Practice Guide, 2005, 42 p. Available: http://energy.gov/sites/prod/files/Good%20Practices%20Guide%20for%20Firewall%20Deployment.pdf.

[61] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya, J.K. Kalita, Network Attacks: Taxonomy, Tools and Systems, Journal of Network and Computer Applications, Vol. 40, 2014, pp. 307-324.

[62] R. Shirey, Internet Security Glossary, Request for Comments 2828, The Internet Engineering Task Force (IETF), 2000, 211 p.

[63] Y. Jang, J. Choi, Detecting SQL Injection Attacks Using Query Result Size, Computers & Security, Vol. 44, 2014, pp. 104-118.

[64] M. Bravenboer, E. Dolstra, E. Visser, Preventing Injection Attacks with Syntax Embeddings, Science of Computer Programming, Vol. 75, No. 7, 2010, pp. 473-495.

[65] B. Parno, C. Kuo, A. Perrig, Phoolproof Phishing Prevention, Lecture Notes in Computer Science, Vol. 4107, 2006, pp. 1-19.

[66] K. Olk, Something Phishy: How to Avoid Being Caught in the Net of Specialized Spam, 2005, 21 p.

[67] K. Smith, Security Awareness: Help the Users Understand, 2001, 14 p. Available: http://www.sans.org/reading-room/whitepapers/awareness/security-awareness-users-understand-414.

[68] Centre for the Protection of National Infrastructure (CPNI), Good Practice Guide: Process Control and SCADA Security - Guide 4. Improve Awareness and Skills, 2008, 22 p. Available: http://www.cpni.gov.uk/documents/publications/2008/2008027-gpg_scada_improve_awareness_and_skills.pdf.

[69] P. Rogers, K. Smyser, Virus Causes Massive Shutdown at Cook County Highway Department, NBC Chicago, web page. Available (accessed on 1 June, 2015): http://www.nbcchicago.com/investigations/Virus-Causes-Massive-Shutdown-at-Cook-County-Highway-Department-217025381.html.

[70] ICS-CERT, Monitor October-December 2012, ICS-CERT Monitor ICS-MM201212, U.S. Department of Homeland Security, 2013, 15 p. Available: https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2012.pdf.

[71] R.S. Piggin, Securing SCADA in the Cloud: Managing the Risks to Avoid the Perfect Storm, IET & ISA 60th International Instrument Symposium, 24-26 June, 2014, The Institution of Engineering and Technology, pp. 1-6.

[72] National Cyber Security Centre, Pilvipalveluiden turvallisuus, Finnish Communications Regulatory Authority, FICORA, 2014, 24 p. Available: https://www.viestintavirasto.fi/attachments/tietoturva/Pilvipalveluiden_tietoturva_organisaatioille.pdf.

[73] K. Wilhoit, SCADA in the Cloud - A Security Conundrum? 2013, 11 p. Available: http://www.trendmicro.co.uk/media/misc/scada-in-the-cloud-a-security-conundrum-en.pdf.

[74] OSGi Alliance, OSGi Core Release 6, 2014, 450 p.

[75] Kalmar Global, Kalmar Automation System Architecture, 2014, 53 p.

[76] W. Strauch, Container Handbook - Volume 1, The German Insurance Association, 2003, 475 p.

[77] P. Parrend, S. Frénot, Java Components Vulnerabilities - An Experimental Classification Targeted at the OSGi Platform, Projet ARES 6231, 2007, 87 p. Available: http://www.rzo.free.fr/publis/RR-6231.pdf.

[78] P. Parrend, S. Frénot, More Vulnerabilities in the Java/OSGi Platform: A Focus on Bundle Interactions, Project ARES 6649, 2008, 82 p. Available: https://hal.inria.fr/inria-00322138/PDF/RR-6649.pdf.

[79] N. Geoffray, G. Thomas, G. Muller, P. Parrend, S. Frénot, B. Folliot, I-JVM: a Java Virtual Machine for Component Isolation in OSGi, IEEE/IFIP International Conference on Dependable Systems & Networks (DSN '09), 29 June - 2 July, 2009, IEEE, pp. 544-553.

[80] P. Parrend, Security for Java Platforms, in: M. Hayes, I. Johansen (ed.), Java software and embedded systems, Nova Science Publishers, Inc., 2010, pp. 143-170.

[81] G.S. Kalra, A Pentesters Guide to Hacking ActiveMQ Based JMS Applications, 2014, 26 p. Available: http://www.mcafee.com/mx/resources/white-papers/foundstone/wp-pentesters-guide-hacking-activemq-jms-applications.pdf.

[82] McAfee, JMSDigger, web page. Available (accessed on 2 July, 2015): http://www.mcafee.com/sg/downloads/free-tools/jmsdigger.aspx.

[83] CVE Details, Common Vulnerability Enumeration, web page. Available (accessed on 3 July, 2015): http://www.cvedetails.com/.

[84] M.S. Ware, C.J. Fox, Securing Java Code: Heuristics and an Evaluation of Static Analysis Tools, Proceedings of the Static Analysis Workshop (SAW 2008), 12 June, 2008, Association for Computing Machinery, New York, NY, USA, pp. 12-21.

[85] M. Howard, D. Leblanc, J. Viega, 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them, McGraw-Hill, 2009, 433 p.

[86] S. Rouiller, Virtual LAN Security: Weaknesses and Countermeasures, 2003, 51 p. Available: https://www.sans.org/reading-room/whitepapers/networkdevs/virtual-lan-security-weaknesses-countermeasures-1090.

[87] K. Colo, Using VLAN's in Network Design, 2012, 5 p. Available: http://www.spidercloud.com/assets/pdfs/VLANSecurityNexusWP1212.pdf.

[88] I.N. Fovino, L. Guidi, M. Masera, A. Stefanini, Cyber Security Assessment of a Power Plant, Electric Power Systems Research, Vol. 81, No. 2, 2011, pp. 518-526.

[89] B. Schneier, Attack Trees, Dr Dobb's Journal, 1999, http://www.drdobbs.com/attack-trees/184411129.

[90] D. Raptis, T. Dimitrakos, B.A. Gran, K. Stølen, The CORAS Approach for Model-based Risk Analysis Applied to the e-commerce Domain, Proceedings of the Communication and Multimedia Security, 26-27 September, 2002, Kluwer, pp. 169-181.

[91] J. McDonald, N. Oualha, A. Puccetti, A. Hecker, F. Planchon, Application of EBIOS for the Risk Assessment of ICT use in Electrical Distribution Substations, PowerTech (POWERTECH), 2013 IEEE Grenoble, 16-20 June, 2013, IEEE, pp. 1-6.

[92] I.N. Fovino, M. Masera, A Service Oriented Approach to the Assessment of Infrastructure Security, in: E. Goetz, S. Shenoi (ed.), Critical Infrastructure Protection, Springer US, 2008, pp. 367-379.

[93] C. Alberts, A. Dorofee, Managing Information Security Risks: The OCTAVE (SM) Approach, Addison-Wesley Professional, 2002, 512 p.

[94] E. Byres, M. Franz, D. Miller, The Use of Attack Trees in Assessing Vulnerabilities in SCADA Systems, International Infrastructure Survivability Workshop (IISW'04), 4 December, 2004, IEEE, pp. 9.

[95] S. Mauw, M. Oostdijk, Foundations of Attack Trees, 8th International Conference on Information Security and Cryptology, 1-2 December, 2005, Springer Berlin Heidelberg, pp. 186-198.

[96] SeaMonster, SeaMonster - Security Modeling Software, web page. Available (accessed on 20 July, 2015): http://sourceforge.net/projects/seamonster/.

[97] A.P. Moore, R.J. Ellison, R.C. Linger, Attack Modeling for Information Security and Survivability, CMU/SEI-2001-TN-001, Software Engineering Institute, Pittsburgh, PA, 2001, 30 p. Available: http://resources.sei.cmu.edu/asset_files/TechnicalNote/2001_004_001_13793.pdf.

[98] Information Technology - Security Techniques - Information Security Risk Management, ISO/IEC 27005:2011, Geneve, Switzerland, 2011, pp. 127.

[99] Agence nationale de la sécurité des systèmes d'information (ANSSI), Cybersecurity for Industrial Control Systems - Detailed Measures, 2014, 97 p. Available: http://www.ssi.gouv.fr/uploads/2014/01/industrial_security_WG_detailed_measures.pdf.

[100] ICS-CERT, Monitor January-April 2014, ICS-CERT Monitor ICS-MM201404, U.S. Department of Homeland Security, 2014, 18 p. Available: https://ics-cert.us-cert.gov/sites/default/files/Monitors/ICS-CERT_Monitor_Oct-Dec2012.pdf.

[101] P. Bowen, J. Hash, M. Wilson, Information Security Handbook: A Guide for Managers, SP 800-100, National Institute of Standards and Technology (NIST), 2006, 178 p. Available: http://csrc.nist.gov/publications/nistpubs/800-100/SP800-100-Mar07-2007.pdf.

[102] A. Johnson, K. Dempsey, R. Ross, S. Gupta, D. Bailey, Guide for Security-Focused Configuration Management of Information Systems, SP 800-128, National Institute of Standards and Technology (NIST), 2011, 88 p. Available: http://csrc.nist.gov/publications/nistpubs/800-128/sp800-128.pdf.

[103] Bundesverband der Energie und Wasserwirtschaft e.V., BDEV, Requirements for Secure Control and Telecommunication Systems, 2014, 36 p. Available: http://www.insys-icom.com/bausteine.net/f/10809/BDEW_Whitepaper_v1-1_Okt2014.pdf.

[104] K. Scarfone, M. Souppaya, Guide to Enterprise Patch Management Technologies, SP 800-40r3, National Institute of Standards and Technology (NIST), 2013, 26 p. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-40r3.pdf.

[105] Kalmar Global, Kalmar Automation IT Specification, 2014, 24 p.

[106] J.-E. Ekberg, K. Kostiainen, N. Asokan, The Untapped Potential of Trusted Execution Environments on Mobile Devices, Security & Privacy, IEEE, No. 4, 2014, pp. 29-37.

[107] W. Dai, H. Jin, D. Zou, S. Xu, W. Zheng, L. Shi, TEE: A Virtual DRTM Based Execution Environment for Secure Cloud-End Computing, Proceedings of the 17th ACM Conference on Computer and Communications Security, 4-8 October, 2010, Association for Computing Machinery, ACM, pp. 663-665.

[108] Centre for the Protection of National Infrastructure (CPNI), Good Practice Guide: Process Control and SCADA Security - Guide 3. Establish Response Capabilities, 2008, 21 p. Available: http://www.cpni.gov.uk/documents/publications/2008/2008026-gpg_scada_response_capabilities.pdf?epslanguage=en-gb.

[109] D. Kilman, J. Stamp, Framework for SCADA Security Policy, Sandia National Laboratories report SAND2005-1002C, 2005, 6 p. Available: http://energy.sandia.gov/wp-content//gallery/uploads/sand_2005_1002C.pdf.

[110] Information Technology - Security Techniques - Code of Practice for Information Security Controls, ISO/IEC 27002:2005, Geneve, Switzerland, 2005, pp. 183.

[111] Industrial Communication Networks. Network and System Security. Part 1-1: Terminology, Concepts and Models, Suomen Standardoimisliitto SFS, IEC/TS 62443-1-1:fi, Helsinki, 2012, pp. 153.

[112] Department of Homeland Security (DHS), Cyber Security Procurement Language for Control Systems, 2009, 145 p. Available: https://ics-cert.us-cert.gov /sites/default/files/documents/Procurement_Language_Rev4_100809.pdf.

[113] M. Wilson, J. Hash, Building an Information Technology Security Awareness and Training Program, SP 800-50, National Institute of Standards and Technology (NIST), 2003, 39 p. Available: http://csrc.nist.gov/publications/nistpubs/800-50/NIST-SP800-50.pdf.

[114] T. Aho, J. Koskinen, A. Nieminen, A Secure OSGi Environment for Untrusted Web Applications, Proceedings of the Second Nordic Symposium on Cloud Computing & Internet Technologies, 2-3 September 2013, Association for Computing Machinery, pp. 15-21.

[115] M. Anne, K. Attouchi, D. Henry-de-Villeneuve, J. Pulou, Jasmin: an Alternative for Secure Modularity Inside the Digital Home, Proceedings of the 15th ACM SIGSOFT Symposium on Component Based Software Engineering, 25-28 June, 2012, Association for Computing Machinery, pp. 145-150.

[116] VMware, vSphere Security, 2011, 98 p. Available: https://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-security-guide.pdf.

[117] VMware, Java in Virtual Machines on VMware ESX: Best Practices, 2009, 19 p. Available: http://www.vmware.com/files/pdf/Java_in_Virtual_Machines_on_ESX-FINAL-Jan-15-2009.pdf.

[118] VMware, VMware Security Hardening Guides, web page. Available (accessed on 25 August, 2015): http://www.vmware.com/security/hardening-guides.html.

[119] The Apache Software Foundation, Encrypted Passwords, web page. Available (accessed on 18 August, 2015): http://activemq.apache.org/encrypted-passwords.html.

[120] The Apache Software Foundation, URI Protocols, web page. Available (accessed on 18 August, 2015): http://activemq.apache.org/uri-protocols.html.

[121] J. Falco, S. Hurd, D. Teumim, Using Host-Based Antivirus Software on Industrial Control Systems: Integration Guidance and a Test Methodology for Assessing Performance Impact, SP 1058, 2006, 50 p. Available: http://www.nist.gov/customcf/get_pdf.cfm?pub_id=823596.

[122] G. McGraw, E. Felten, Twelve Rules for Developing More Secure Java Code, web page. Available (accessed on 27 July, 2015): http://www.javaworld.com/article/2076837/mobile-java/twelve-rules-for-developing-more-secure-java-code.html.

[123] F. Long, D. Mohindra, R.C. Seacord, D.F. Sutherland, D. Svoboda, Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs, Addison-Wesley Professional, 2013, 304 p.

[124] Oracle, Secure Coding Guidelines for Java SE, Version 5.0, web page. Available (accessed on 22 July, 2015): http://www.oracle.com/technetwork/java/seccodeguide-139067.html.

[125] G. Díaz, J.R. Bermejo, Static Analysis of Source Code Security: Assessment of Tools Against SAMATE Tests, Information and Software Technology, Vol. 55, No. 8, 2013, pp. 1462-1476.

[126] Global Information Assurance Certification, (GIAC), GIAC Secure Software Programmer - Java (GSSP-JAVA), web page. Available (accessed on 27 August, 2015): http://www.giac.org/certification/secure-software-programmer-java-gssp-java.

[127] The International Council of Electronic Commerce Consultants, (EC-Council), EC-Council Certified Secure Programmer (ECSP) Java, web page. Available (accessed on 27 August, 2015): http://www.eccouncil.org/Certification/ec-council-certified-secure-programmer-java.

[128] SecurityInnovation, ILT - ISSECO CPSSE Certification, web page. Available (accessed on 27 August, 2015): https://www.securityinnovation.com/training/application-security/instructor-led/courses/ilt-isseco-cpsse-certification.html.

[129] Office of Cyber and Infrastructure Analysis (OCIA), The Future of Smart Cities: Cyber-Physical Infrastructure Risk, 2015, 50 p. Available: https://ics-cert.us-cert.gov/Future-Smart-Cities-Cyber-Physical-Infrastructure-Risk.

APPENDIX A: ATTACK TREES

The attack trees below describe different attack vectors that can be used to attack the External Interface Service (EIS) of Kalmar's terminal automation system. There are three main goals (roots) for attacks: Denial of Service (DoS), moving Container Handling Equipment (CHE) and data leakage. The trees have been broken up to smaller sections so that they can be presented. Certain parts of attacks, such as gaining knowledge on the target system and gaining access to the system are a part of most attacks, so to avoid repetition they are presented as separate trees as well. A gray box denotes that the vector is expanded in another tree. For example if a gray box says "access to the ICS network", see the unauthorized access trees for the rest of the vector.

Gaining knowledge on the target system is the beginning of most attacks, especially when the attack target has been chosen, but attack method still needs to be found. Knowledge on the network structure helps an attacker to find vulnerable devices and possible routes to whatever device or software the attacker is targeting. Knowledge on system operation and message structures helps the attacker figure out what actually needs to be attacked in order to achieve the goal. Figure A-1 shows how knowledge can be acquired.

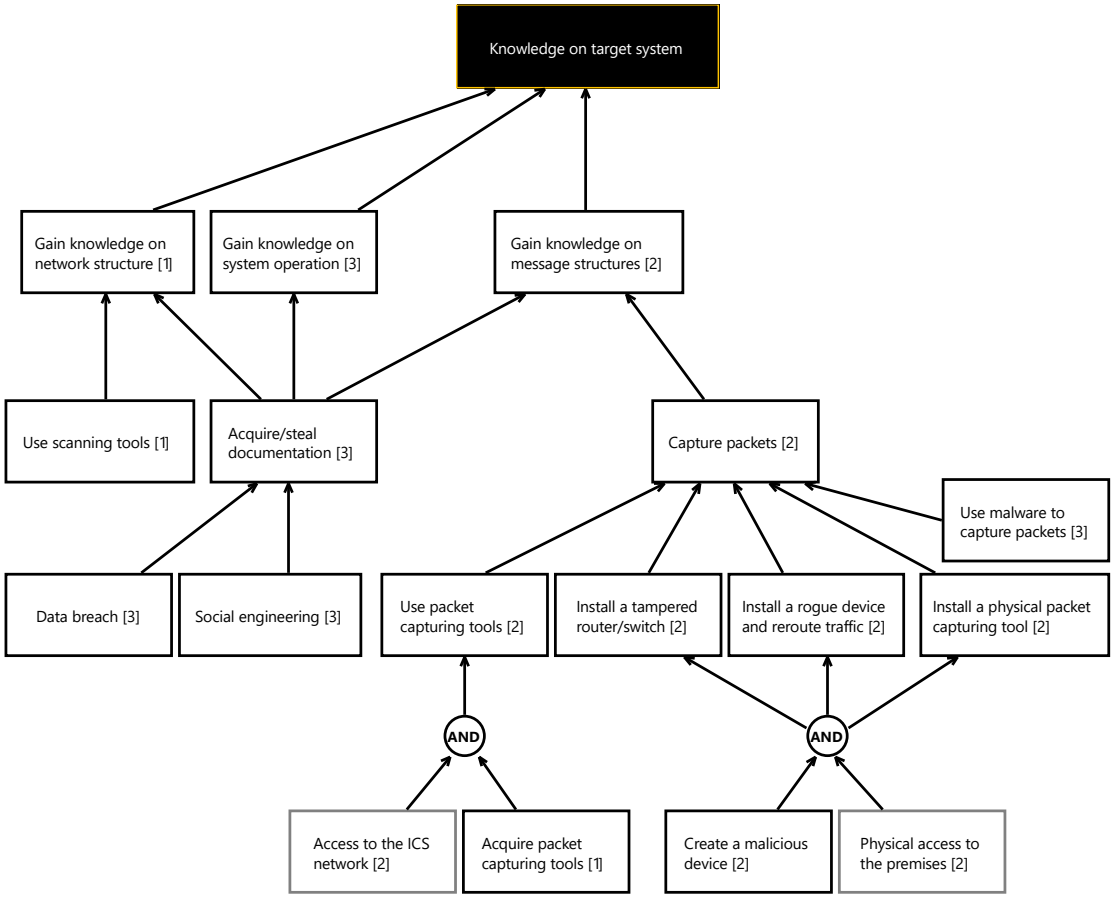

Figure A-1. *Attack tree: gaining knowledge on target system.*

Practically all attacks require unauthorized access to some part of the system. The access can be either remote (Figure A-2) or physical (Figure A-3). Remote access can come from the Internet or from networks adjacent to the target network. Here remote access has been divided into two cases: access to the ICS network and access to the process server. While access to the ICS network grants access to the data moving between devices in that network, access to the process server grants access also to data that never leaves the server.

The easiest way to access the ICS network from the Internet would be by using one or more intrusion tools that are specifically made to find vulnerable points on the outside perimeter of the network. Nowadays such tools have easy-to-use interfaces and are publicly available, so such attacks are likely to occur. Another possibility would be malware infection: a Remote Access Trojan (RAT) could be used to open a command and control connection to the ICS network. The RAT could be delivered to the system for example in a USB-drive or through an Internet browser. It is also possible for malware to propagate from the office network, which might be an easier target for an attacker to compromise first.

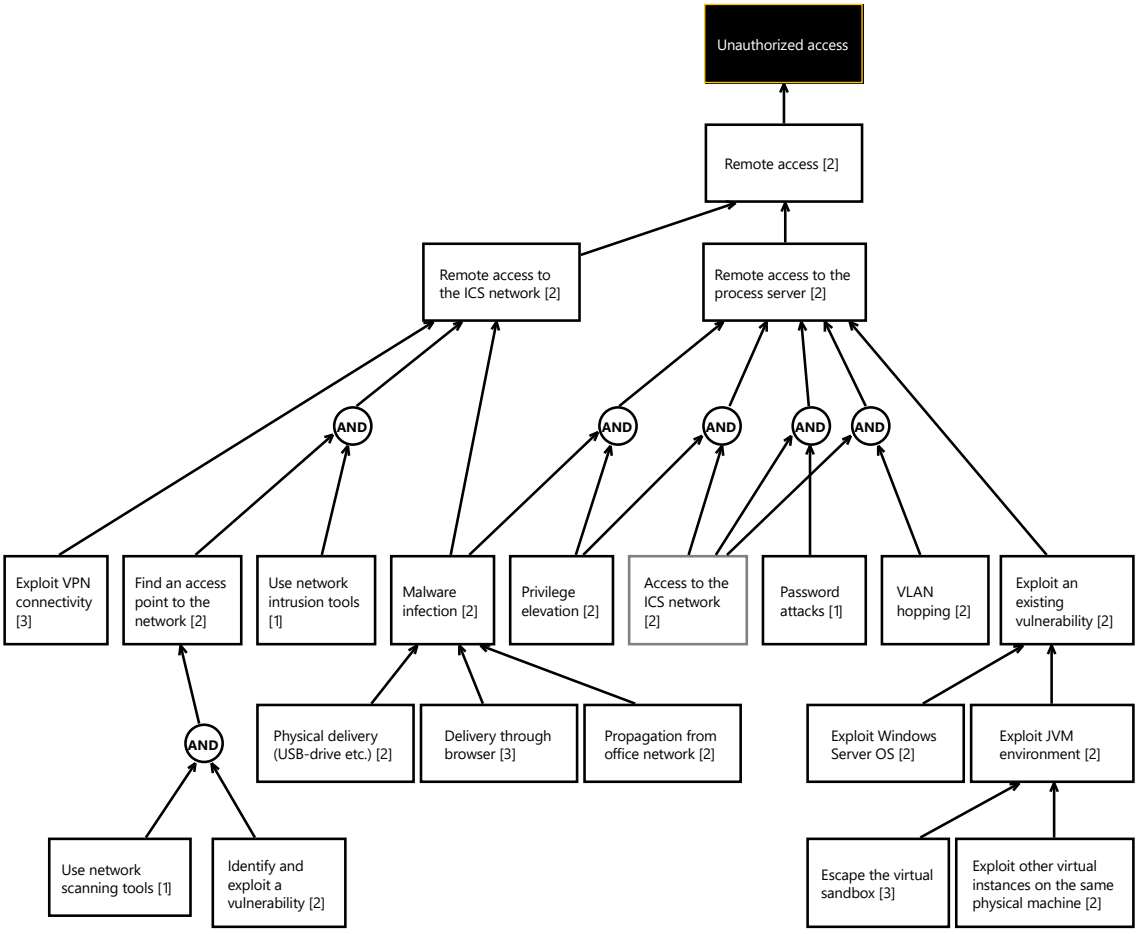

Figure A-2. *Attack tree: unauthorized remote access.*

Physical access means that the attacker is able to physically touch the target. Physically accessing the ICS network practically means plugging in to a switch or a router to capture and inject messages or using one of the desktops connected to the network. This of course means that the attacker must first be able to access the premises where these devices are located. Physical access to the process server room would mean access to the process server and to all data on the server. For example social engineering, bribing or intimidation could be used to access the server room.

Access to the container yard would make it possible to open containers and find out what is inside them, which of course is useful information for example when attempting to steal containers. The access control system managing access to the yard could be exploited by forging access control messages, or by stealing or forging an ID card that is used to identify anyone entering the yard. All of the physical attacks would be significantly easier for employees who already have the required access credentials.

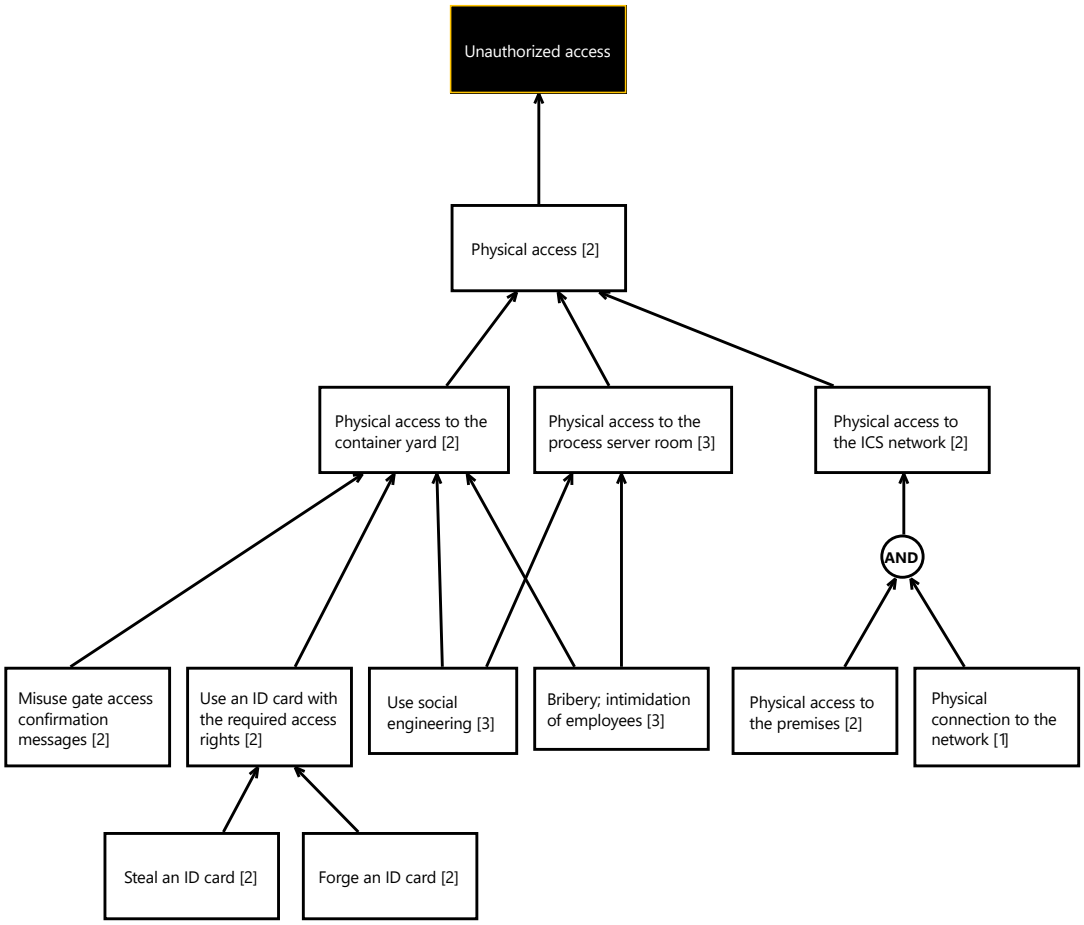

Figure A-3. *Attack tree: unauthorized physical access.*

First of the three identified main goals of an attacker is Denial of Service (DoS). DoS attacks are generally easier to carry out than other attacks, so there are quite a few attack vectors. DoS attacks can be carried out on purpose, but they can also be launched by accident for example by incompetent employees. In flooding attacks (see Figure A-4) the attacker sends more messages than the receiver can handle, which makes the receiv-

er incapable of responding to other messages in time. Such attacks are simple to execute as ready-made tools exist. All devices that are visible to the Internet are susceptible to random attackers, who scan the Internet for suitable targets. Therefore devices that need to be accessed from the Internet should be configured so that they can only be accessed from certain IP addresses, effectively making them invisible to others.

Once inside the ICS network, it is possible to interfere with communications of the network. Network devices can be attacked in different ways to change how network packets are routed. Physically damaging a network device naturally stops the traffic through that device entirely, but misconfiguration, spoofing and malware attacks can result in packets delivered to wrong places either constantly or intermittently. Especially intermittent issues are difficult to track down and solve, so such attacks would last longer than problems caused by a broken device.

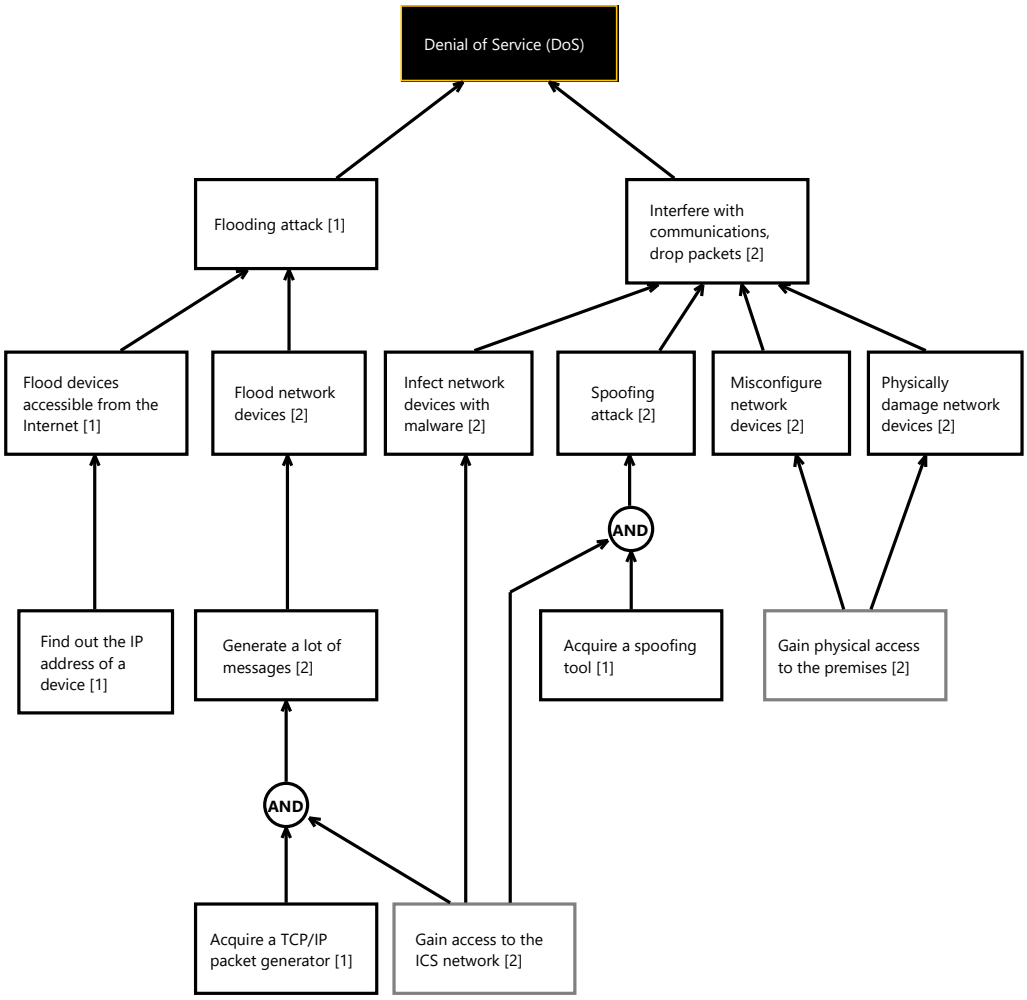

Figure A-4. *Attack tree: flooding attacks and communication interference.*

Container Handling Equipment (CHE) colliding with other objects will force the terminal operator to halt the automated container handling, so collisions can be seen as DoS attacks, even though the motive might actually be causing financial harm (Figure A-5).

As the container move requests and other high-level instructions to CHE are given by EIS, manipulating these instructions may result in collisions, even when safety mechanisms are activated. In theory, the system is aware of all static obstacles on the container yard, but if there are any other obstacles on the yard, a CHE might collide with them. For example moving a container to a place where it wouldn't normally be and then removing it from the container map of CHEs would make it "invisible".

Editing the container map also offers attack vectors other than collisions (Figure A-5). If an attacker adds randomly new container instances to the system without their physical counterparts, CHEs will start driving around looking for them. Removing container instances from the map will become a problem when one of these "invisible" containers needs to be moved as a part of another move, for example when the "invisible" container is on top of the container that needs to be moved.

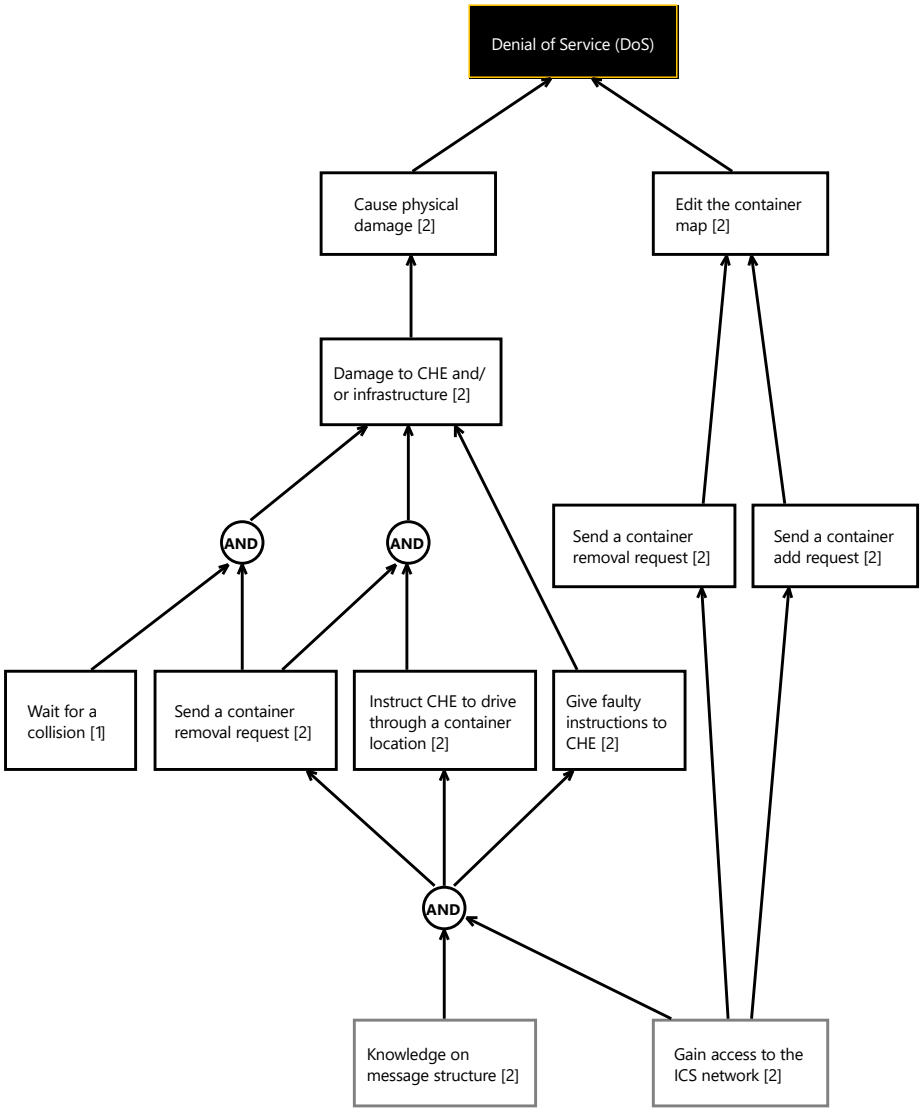

Figure A-5. *Attack tree: physical damage and editing container map.*

When considering EIS, nuking attacks (Figure A-6) are basically attacks that somehow prevent operation of EIS or the process server, which means that access to the server is required. Once an attacker has access to the process server, shutting down the Operating System (OS) or the virtualization environment is the matter of having high enough privileges to kill processes on the server. Typically in an industrial setting most or all users have administrative privileges for easier maintenance, so gaining access to the server is usually enough. The same goes for removing vital system data, such as configuration files or other files that are necessary for the system to keep running.

Freezing OSGi bundles is a more complicated task and requires the ability to install new bundles on the server. Creating a malicious bundle requires knowledge of how OSGi works, but as was discussed in Sub-section 4.3.1, there are vulnerabilities in OSGi that allow bundles to be frozen.

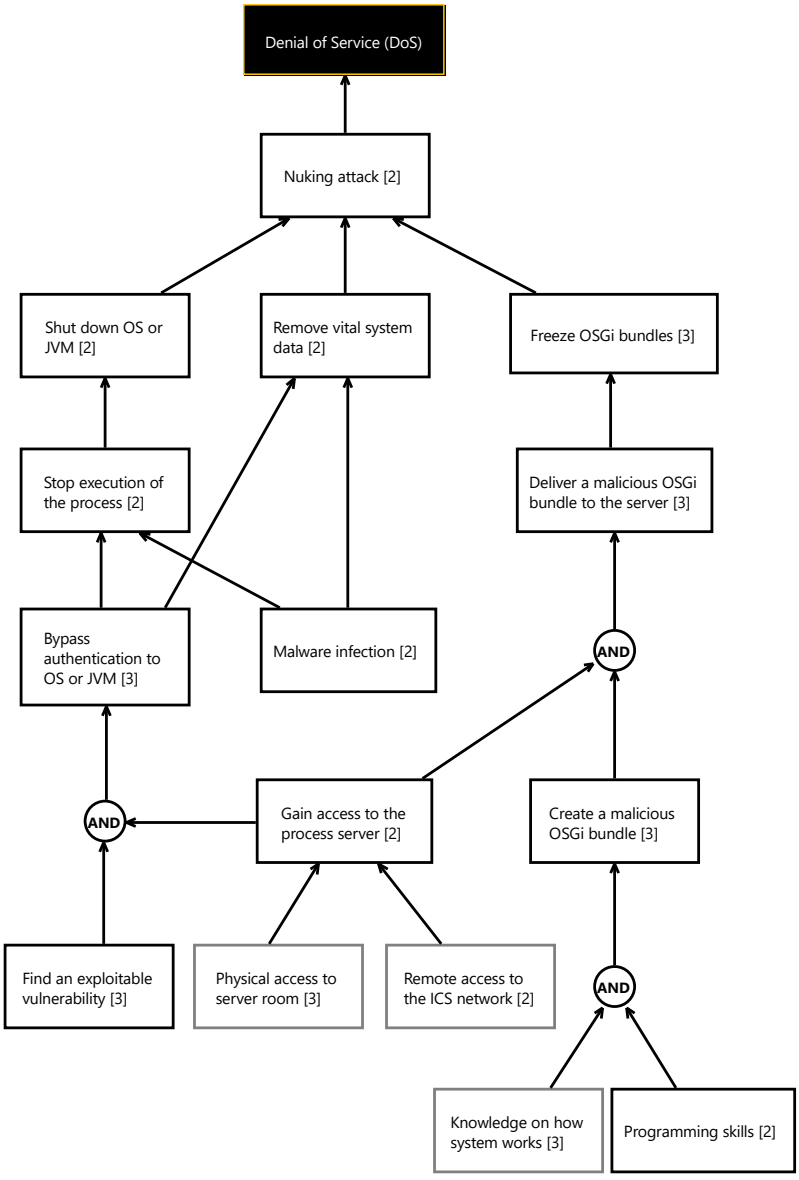

Figure A-6. *Attack tree: nuking attacks.*

The second main goal identified in this thesis is moving a CHE. Being able to move a CHE means that it is also possible to move containers. When exploiting EIS to move containers, the most convenient way is to craft messages that appear to be coming from the Terminal Operating System (TOS) and inject these messages to the ICS network (Figure A-7). An attacker can either create new messages or modify existing ones.

When modifying existing messages, one doesn't need to worry about the syntax of the entire message as only small adjustments are needed, but one has to be able to intercept and modify messages. When creating new messages, the attacker must understand the exact structure of the messages, but interception of messages is not necessary. It is also possible to give multiple move requests at once using a job list. Sending multiple move requests at once shortens the time that the attacker needs to be present or have access to the ICS network, which lessens the likelihood of getting caught.

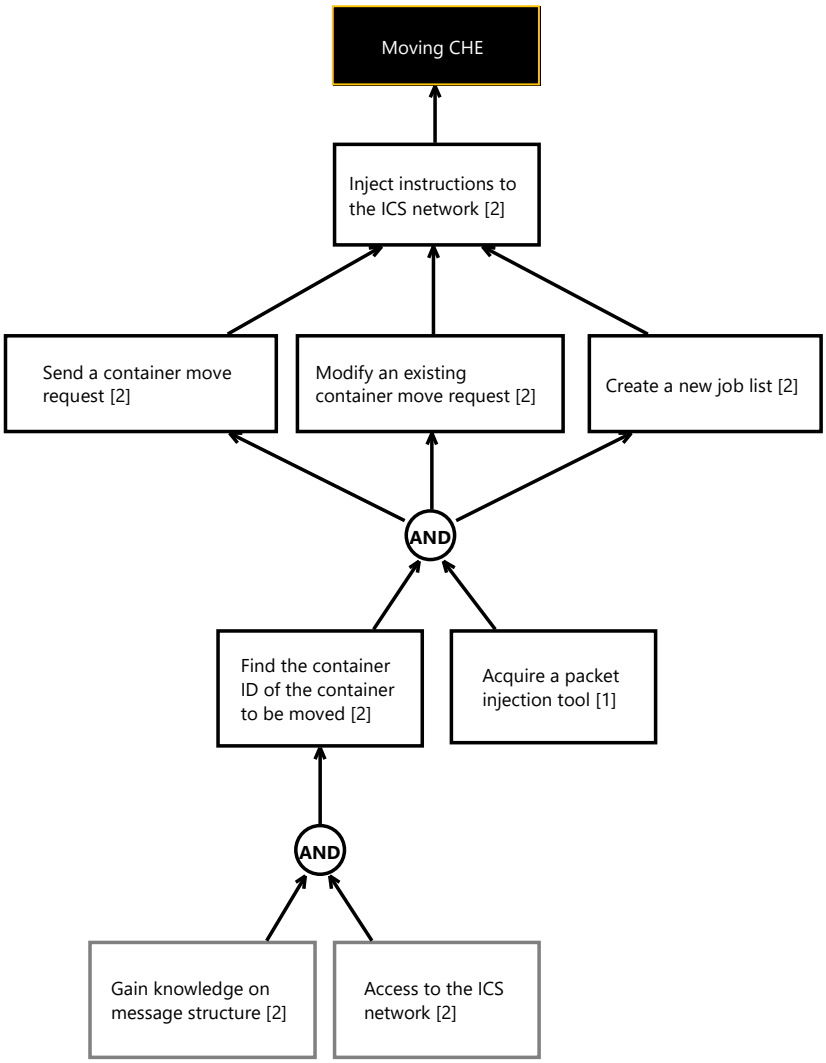

Figure A-7. *Attack tree: move CHE by injecting instructions to the ICS network.*

There is also another way to move CHE: exploiting OSGi (Figure A-8). Exploiting OSGi to modify data is similar to freezing OSGi bundles, but in order to be able to modify data, an attacker would have to understand how other bundles work. This information can't really be obtained from the system itself, so this attack vector requires either a knowledgeable insider or stealing information from other sources. Of course the attacker must also possess the required knowledge to create a malicious bundle that exploits a vulnerability in OSGi and makes controlled changes to the data.

Even though interfering with network packets directly may seem more convenient than exploiting OSGi, this method has its advantages. For example if the connection between TOS and EIS was protected with a VPN tunnel, the messages wouldn't be accessible using simple MitM-attacks, but the messages would again be unencrypted once they reach EIS. Therefore VPN wouldn't protect the communication from OSGi exploits. Also an OSGi exploit would presumably be more difficult to detect than a MitM-attack, which would help the attacker to stay undetected longer.

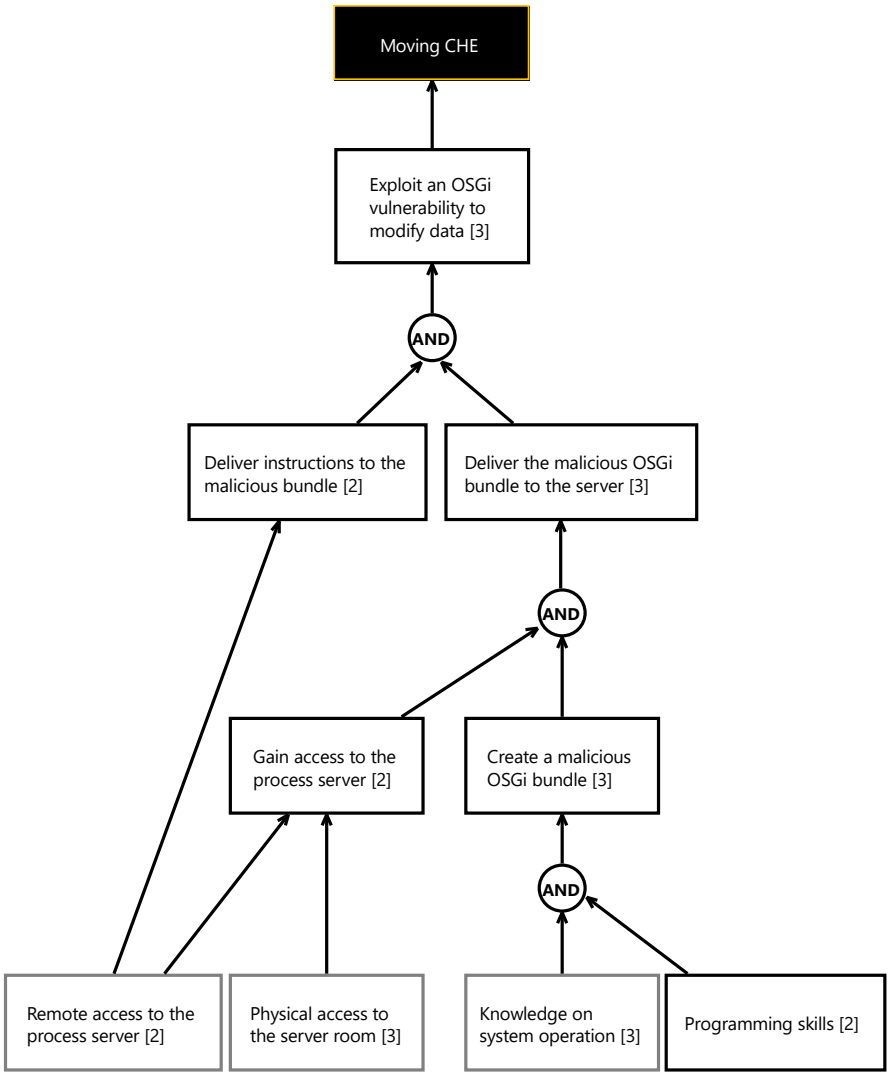

Figure A-8. *Attack tree: move CHE through OSGi exploit.*

The third and final main goal of a cyber attack is data leakage. In the scope of this thesis this goal is divided in two sub-goals: leakage of container information and leakage of process data. Process data would most likely be stolen by capturing data between EIS and Fault Management and Diagnostics System (FMDS) or Marine Telematics System (MTS), both of which produce data about the state and performance of the system (Figure A-9). As the data is not encrypted, capturing the data isn't that difficult in itself, but access to the ICS network must be obtained first.

Container information is an essential part of attack vectors where containers are moved. There are two ways to obtain container information from the system: directly from inside EIS (Figure A-9), or by capturing messages between EIS and other system components (Figure A-10). If the attacker has access to the process server, it is again possible to use an OSGi vulnerability. The attack vector is basically the same as when moving CHE and the same exploit could be used for both purposes. As container information is required when moving containers, these attacks are likely to occur simultaneously.

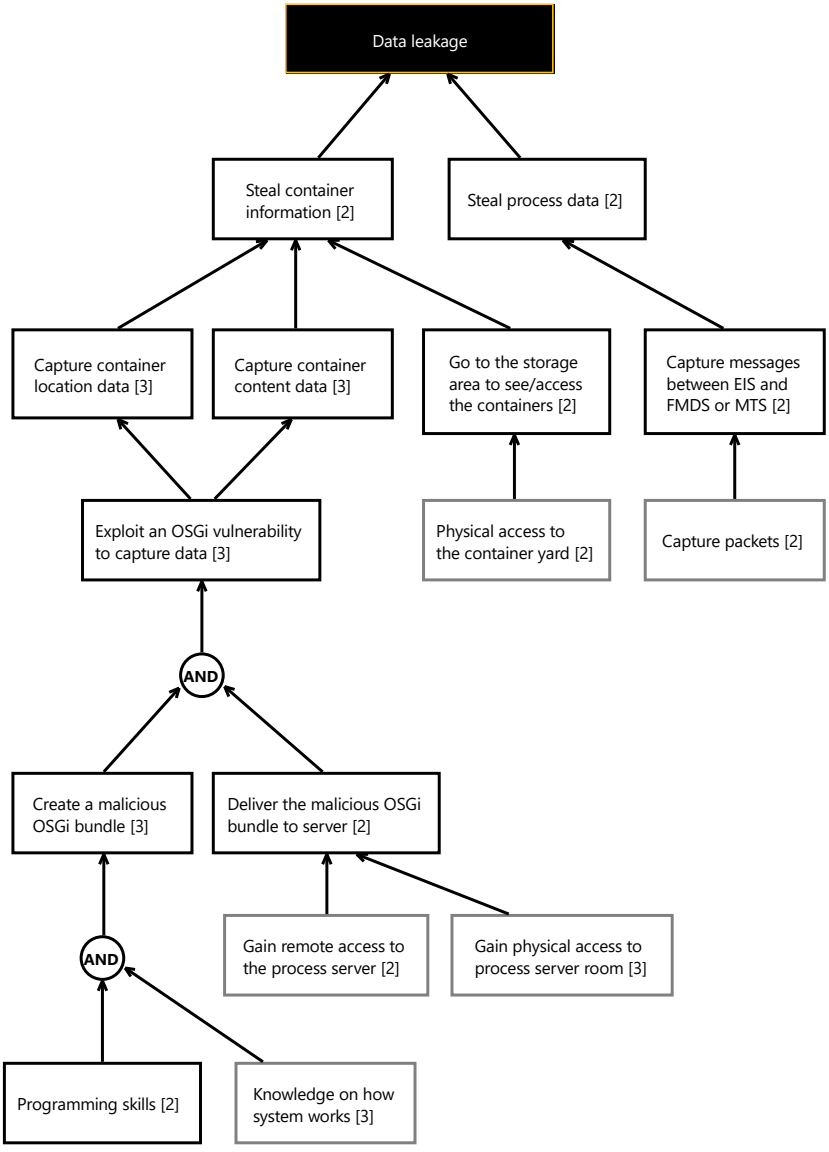

Figure A-9. *Attack tree: stealing container information and process data.*

Container information can also be obtained from communication between EIS and other components of the system. Data can either be captured until required information has been found, or certain messages can be injected to the network in order to invoke desired replies. For example sending a map synchronization request to EIS invokes a reply that includes ID of each container on the yard. Using the IDs obtained, an attacker can then request the location of a container.

Container information can also be obtained by physically accessing the container yard, as was described in Figure A-3. By accessing the yard, attackers can determine which containers they want to steal and then use a cyber attack to move the container for example on a trailer.

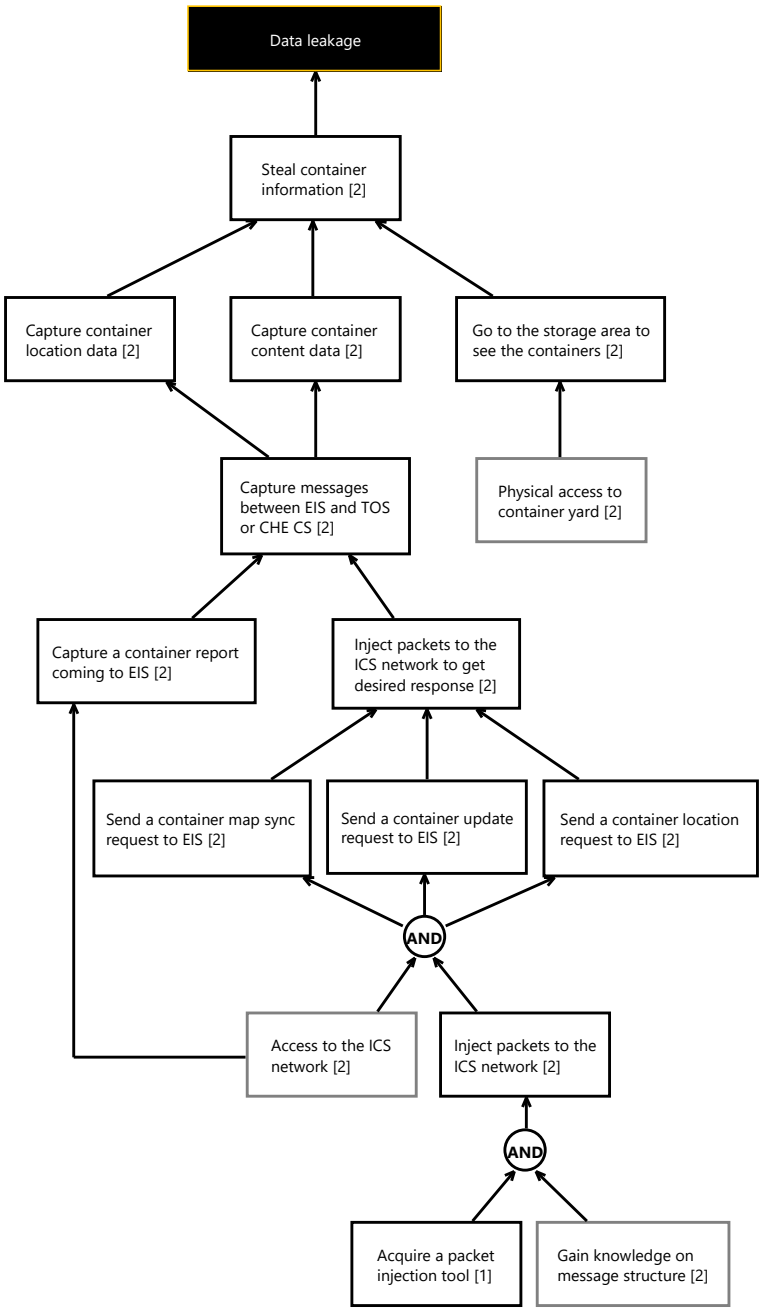

Figure A-10. *Attack tree: more ways to steal container information.*