Master of Science in Engineering: Computer Security
February 2019

Attacking the Manufacturing Execution System

Leveraging a Programmable Logic Controller on the Shop Floor

Fredrik Johansson

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfilment of the requirements for the degree of Master of Science in Engineering: Computer Security. The thesis is equivalent to 20 weeks of full time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Fredrik Johansson
E-mail: fredrikjohansson.thesis@gmail.com

University advisor:
Assistant Professor Martin Boldt
Department of Computer Science and Engineering

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

Abstract

Background. Automation in production has become a necessity for producing companies to keep up with the demand created by their customers. One way to automate a process is to use a piece of hardware called a programmable logic controller (PLC). A PLC is a small computer capable of being programmed to process a set of inputs, from e.g. sensors, and create outputs, to e.g. actuators, from that. This eliminates the risk of human errors while at the same time speeding up the production rate of the now near identical products. To improve the automation process on the shop floor and the production process in general a special software system is used. This system is known as the manufacturing execution system (MES), and it is connected to the PLCs and other devices on the shop floor. The MES have different functionalities and one of these is that it can manage instructions. Theses instructions can be aimed to both employees and devices such as the PLCs. Would the MES suffer from an error, e.g. in the instructions sent to the shop floor, the company could suffer from a negative impact both economical and in reputation. Since the PLC is a computer and it is connected to the MES it might be possible to attack the system using the PLC as leverage.
Objectives. Examine if it is possible to attack the MES using a PLC as the attack origin.
Methods. A literature study was performed to see what types of attacks and vulnerabilities that has been disclosed related to PLCs between 2010 and 2018. Secondly a practical experiment was done, trying to perform attacks targeting the MES.
Results. The results are that there are many different types of attacks and vulnerabilities that has been found related to PLCs and the attacks done in the practical experiment failed to induce negative effects in the MES used.
Conclusions. The conclusion of the thesis is that two identified PLC attack techniques seems likely to be used to attack the MES layer. The methodology that was used to attack the MES layer in the practical experiment failed to affect the MES in a negative way. However, it was possible to affect the log file of the MES in one of the test cases. So, it does not rule out that other MES types are not vulnerable or that the two PLC attacks identified will not work to affect the MES.

Keywords: Manufacturing Execution System, Programmable logic controller, attack types

Sammanfattning

Bakgrund. Automatisering inom produktion har blivit nödvändigt för att företag ska kunna tillgodose den efterfrågan som deras kunder skapar. Ett sätt att automatisera denna process är genom att använde en typ av hårdvara som på engelska kallas för programmable logic controller (PLC). En PLC är en liten dator som man kan programmera så att den bearbetar signaler in, från t.ex. sensorer, och skapar signaler ut, till t.ex. motorer, från det. Detta eliminerar då risken för mänskliga fel samtidigt som det snabbar upp produktionen av de nu nästan identiska produkterna. För att förbättra automatiseringsprocessen på golvet i fabrikerna och även tillverkningsprocessen generellt så används ett speciellt mjukvarusystem. Detta system kallas på engelska execution manufacturing system (MES), och detta system är kopplat till PLCerna och annan utrustning på produktionsgolvet. MESen har olika funktionaliteter och en utav dessa är hantering av instruktioner. Dessa instruktioner kan vara riktade både till anställda samt utrustning så som PLCer. Skulle det inträffa ett fel i MESen, t.ex. i instruktionerna som skickas till produktionsgolvet, så skulle företaget kunna få lida av negativa konsekvenser både ekonomiskt och för företagets rykte. I och med att en PLC är en dator och den är kopplad till MES så kan det finnas möjligheter att utföra attacker mot MESen genom att använda en PLC som utgångspunkt.
Syfte. Undersöka om det är möjligt att utföra en attack på en MES med utgångspunkt från en PLC.
Metod. En litteraturstudie genomfördes för att ta reda på vilka typer av attacker samt sårbarheter relaterade till PLCer som publicerats mellan 2010 och 2018. Ett praktiskt experiment utfördes också, där attackförsök gjordes på ett MES.
Resultat. Resultatet är att det finns många olika attacktyper samt sårbarheter som upptäckts relaterade till PLCer och att de attacker som utfördes i det praktiska experimentet inte lyckades skapa några negativa effekter i det MES som användes.
Slutsatser. Slutsatsen för examensarbetet är att två olika typer av de hittade PLC-attackerna verkar vara kapabla till att användas för att attackera MES-lagret. Metoden som användes i det praktiska försöket lyckades inte påverka MES-lagret negativt. Men det gick att påverka MESens logfil i ett av testfallen, så det går inte att fastslå att andra MES-typer inte är sårbara mot detta eller att de två identifierade PLC-attackerna inte kommer kunna påverka MES-lagret negativt.

Nyckelord: Manufacturing execution system, Programmable logic controller, attack typer

Acknowledgments

We would like to thank Assistant Professor Martin Boldt for the supervision and guidance throughout the thesis project. We would also like to thank the people at Volvo Group IT in Skövde for providing the opportunity to do the thesis together with them and for the support given when needed.

Contents

List of Figures

List of Tables

Chapter 1

Acronyms

CPU	Central processing unit
CVE	Common Vulnerability and Exposure
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DoS	Denial of service
ICS	Industrial control system
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet protocol
IT	Information technology
IoT	Internet of things
JSON	JavaScript Object Notation
MES	Manufacturing execution system
OPC-UA	Open platform communications - unified architecture
OS	Operating system
OSI	Open systems interconnection
PC	Personal computer
PLC	Programmable logic controller
RAM	Random access memory
RTU	Remote terminal unit
SNMP	Simple Network Management Protocol
TCP	Transmission control protocol
XML	Extensible markup language

Chapter 2

Introduction

Today most manufacturing processes requires a high throughput for their products to keep up with the demand created by the customers. For every customer to be pleased with the product they purchased they need to look, feel and behave as identical as possible compared to all the other products from the same process. To achieve this the most certain bet is to automate the process of manufacturing the products. This way there are no human errors in the process, or at least fewer of them, all of which contributes to the perfect copies of products. The automated process will do the same tasks that it was programmed or in other ways taught to do. One way of automating the process is by using a programmable logic controller (PLC).

A PLC is a computer purposely built for industrial applications [1] where the inputs from peripherals, such as sensors, are monitored in real-time. Using these inputs, the PLC then applies some predefined logic to it and the result is then sent to the output peripherals [1]. The introduction and motivation to why PLCs became wide spread is described shortly in [2]. The PLC was the replacement for the old and physical circuits based on relays. These physical circuits required rework as soon as a process was changed. The PLCs on the other hand, changed the game by providing much more flexible ways of altering the circuit for the current needs.

There are a few different system layers used in an automation process or environment. A general overview of these systems and where the PLC is placed can be seen in Figure 2.1. As seen in Figure 2.1 the PLCs are placed at the shop floor, in level 1, in the system hierarchy used by a producing company. Above the PLCs and the industrial control system (ICS), which is a term used to collectively describe the automation systems that is found outside cooperate network [3], there is another layer. At this layer of the hierarchy a component called manufacturing execution system (MES) can be found and it is connected to the shop floor and its devices [4] [5]. The MES is a software system built with several functions aimed towards the actual manufacturing process. The idea of MES was created for manufacturing enterprises for them to run and keep improving the processes they have. This include functionalities such as document control, labor management, data collection and more, as described in [5].

Since the MES is built with the task to keep the processes running and to improve the imperfections that exists, the information it contains and manages is critical to the manufacturing company. If the information in the MES would be incorrect it would have some consequences given the role of the system. These consequences could include the inability to keep deadlines, incorrect work instructions are provided to the shop floor, incorrect resources might be used, or the resources might be used

incorrectly. Depending on the inner workings of the MES and how it is connected to the PLCs on the shop floor, the PLCs could make the MES respond incorrectly if the PLCs sends incorrect values. Would the PLC on the other hand get exploited by a malicious actor, depending on how the MES interprets the incoming traffic from the PLC, the attacker might be able to affect the MES in other ways as well. This attack path, PLC-to-MES, could be easier for someone with shop floor access to exploit, rather than gaining direct access to the network or server room which is connected to the MES.

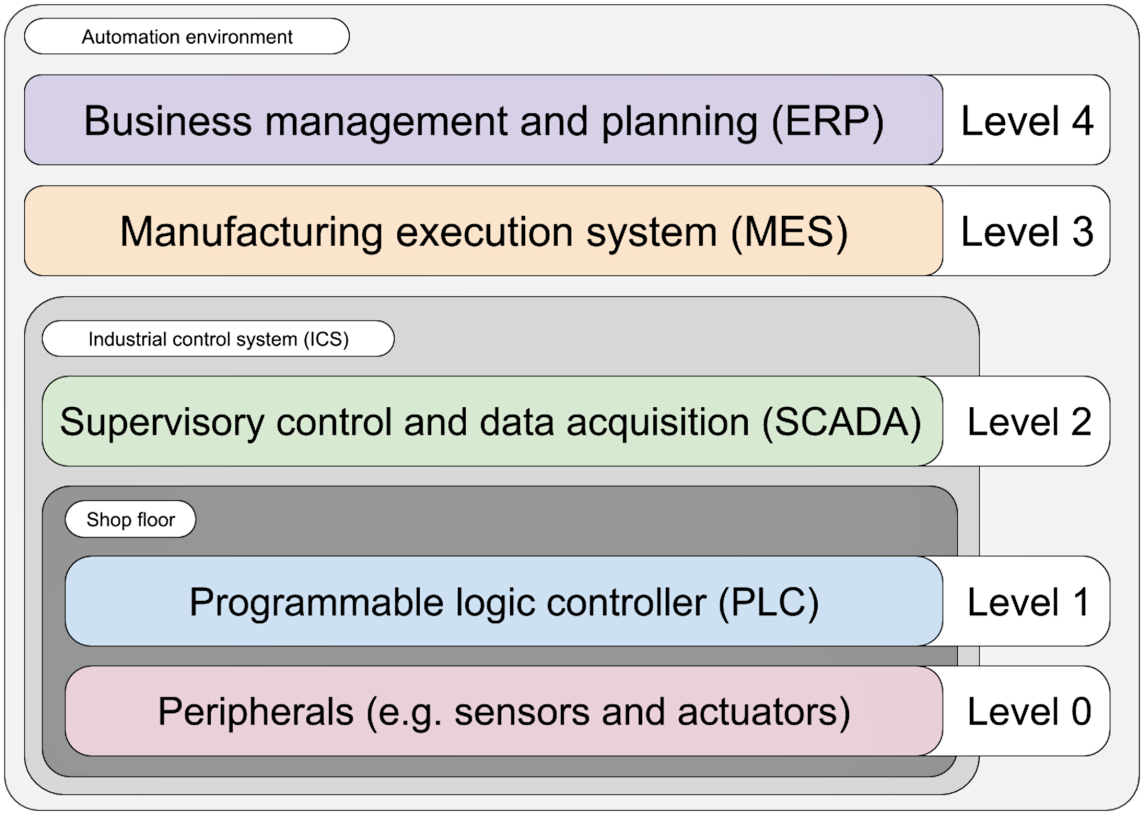

Figure 2.1: A general overview of the system layers in an automation process.

This possibility would definitely be unwanted by a manufacturing company since it could have severely negative consequences, e.g. an attacker could change shop floor instructions so that the products will not be built as desired, send incorrect readings of resource usage so that unnecessary orders for new resources can be placed, or to slow down the process line by reducing the rate of which the products are created. If none of those examples are possible there is always a possibility that an attacker could crash the MES all together or disclose confidential data from the system. All these examples would lead to economical loss for the manufacturing company in one way or another and possibly loss in their reputation among customers as well.

Thus, the overall goal of this master thesis is to investigate and examine if there are any ways for an adversary to leverage a PLC to attack the MES layer, i.e. affect a MES in a way that will introduce unwanted or undesired behaviour, using the PLC-to-MES connection.

2.1 Goal and research questions

The overall goal has been split into three separate sub-goals. The first goal is to identify the different PLC attack types that has been used, both in academic work and other, since the start of 2010. The general trends of the found attack types will also be investigated to see how they have changed during the years. The second goal is to try to attack the MES by leveraging a PLC connected to it. The third and final goal of the thesis is to try to find out what types of protection mechanisms that would be suitable to mitigate the successful attacks found for the previous goal. The research questions that will fulfil the goals, presented above, and what the thesis will try to answer are as follows.

RQ1 What different types of PLC attacks have been identified between 2010 and 2018 and what general trends can be observed in the different types?

RQ2 To what extent is it possible to negatively affect a MES security-wise by exploiting a connected PLC?

RQ3 Which methods are suitable for protecting the MES against those attacks?

2.2 Limitations

This thesis is limited to a couple of points and these limitations will be presented now. For RQ1 only work published between 2010-2018 will be considered. The reason being that too old attack methods will probably already be patched in the devices affected and newer devices will hopefully already have these issues fixed. Another reason for having this year as time constraint is that during the year 2010 the infamous Stuxnet malware [6] was discovered.

The experimentation done to answer RQ2 will be targeting hardware and software provided by Volvo Group IT in Skövde, Sweden. The provided PLC is a Siemens PLC and the targeted MES is a proprietary system responsible for sending instructions to the PLC used on the shop floor in normal operations. Any other brands of the PLCs will not be tested and no other type, version or form of MES will be considered during the experiment.

2.3 Structure of the thesis

The thesis is structured as follows. Chapter 3 will present information needed to understand the rest of the thesis. In chapter 4 a description of the related work is provided. In chapter 5 the methodology used to answer the research question is described. The chapters following, 6, 7 and 8, presents and analyses the results, discusses the validity of the chosen research methodologies and provides the conclusion for the thesis.

Chapter 3

Background

The world of industrial automation is unknown by many. The following sub-chapters presents background information about the systems used in this thesis as well as other information essential to the understanding of the thesis.

3.1 Overview of a PLC

In this sub-chapter a general concept overview of a PLC is presented. How a PLC is constructed and how each of the pieces operate, by themselves and as a whole, is essential for the understanding of the thesis.

3.1.1 Architecture

The architecture of a PLC consists of different layers, each with their own tasks, for the PLC to achieve its functionality. In [7] and [8] these layers are presented as three separate layers, the hardware layer, the firmware/OS layer and lastly the application layer. Both papers, [7] and [8], provides some more details about each of the layers. Below a summary of the layer descriptions is presented and in Figure 3.1 a graphical overview of these layers can be seen.

Hardware The hardware layer is the physical hardware that drives the PLC. A typical PLC consists of a CPU, RAM, non-volatile long-term storage and output and input modules, which connects to peripherals such as valves and sensors.

Firmware/OS The firmware used by the PLC is the piece of software that drives the information exchange between the hardware and the user written software running in the upper level. The tasks that the firmware or OS has is to handle the most basic level operations used by the PLC. This includes execution of user written programs, handle the inputs and outputs connected to the PLC and communications with other devices.

Application The application layer is where the operator written logic programs are being stored, and which the PLC then executes. It is these programs that are responsible for manipulating the output of the PLC. This is done by reading the input and apply some logic to it to calculate the correct outputs for the given inputs. The programs

can be written using different languages, both text and graphical based languages are used.

According to Milinković and Lazić in [9] most of the PLCs today runs a commercially available OSs. But Milinković and Lazić continues with that even though these OSs are commercial some of them are not as famous as the ones normally used in PCs, such as Windows and Linux, and therefore their vulnerabilities are less known [9].

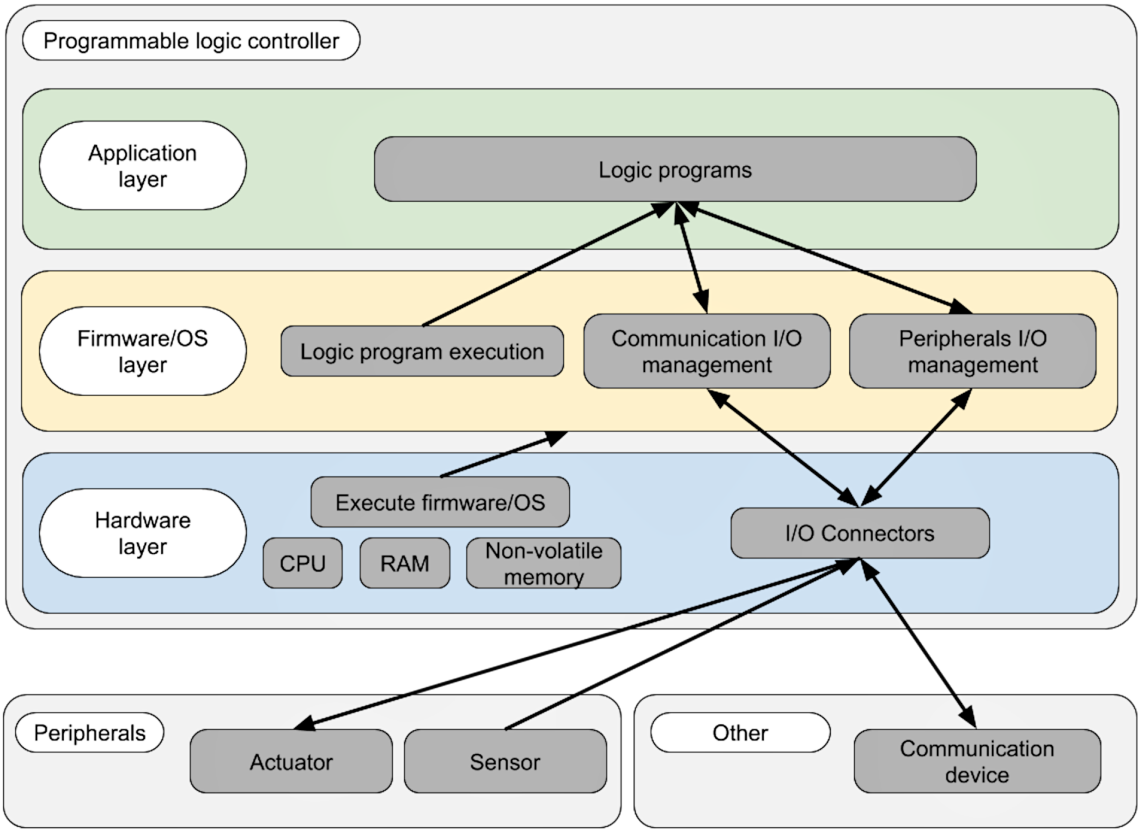

Figure 3.1: The general architecture of a PLC.

Above a more general description of the inner workings of a PLC was described. To better place these layers and their functions in a real-world application a short example will now be presented. The example follows Figure 3.1 closely so each step can be followed using the figure.

To make a simple example, let us say that the automation process only includes a PLC, a motion sensor and a suction cup. Once the motion sensor senses that an object has moved and now is in front of it, the suction cup right above the object is activated and lifts the object. To achieve this both the motion sensor and the suction cup must be connected to the PLC, in the hardware layer. The signals sent from the motion sensor is transported through the PLCs hardware connection. In the firmware/OS layer the peripheral I/O manager in the firmware reads the incoming sensor data and places it in memory available to the next layer. The next layer is the application layer, here the sensor data provided by the firmware is read. Using logic

code, the data is read and the PLC checks if the sensor data indicate that there is an object in front of the sensor. If the data indicate that there is an object present, the PLC sends a signal to the suction cup to activate it. The activation signal is caught by the peripheral I/O manager in the firmware and forwarded to the hardware layer and the connector to which the suction cup is connected to. The activation signal tells the suction cup to activate and it lifts the object in front of the motion sensor.

3.1.2 Communication

For a PLC to communicate with other devices, and now on later years communicate over the internet, the PLC needs to have a protocol and a transportation medium for it to use while communicating. The older PLCs, and still some of the new ones, used older serial connections such as RS-232C when communicating with each other and other devices [10]. However, many of the protocols used in the communication was proprietary and made it difficult for a manufacturing company to integrate a multi-vendor system since a common protocol might not have been available [10]. However, other interfaces and protocols do exist today and one very popular in the general IT world has been adopted into the industrial IT world as well. This is of course Ethernet [10]. Ethernet is what is used in most computers physically wired to the internet, and industrial equipment e.g. PLCs have started to adopt it as well. Since different protocols can use the same physical Ethernet interface and that the performance of Ethernet is better than the older serial connections, more modern protocols have emerged [10]. The industrial computer systems used today have more common protocols and [10] mentions some of them. Below follows a short description of some of the ones mentioned.

Modbus RTU Is a protocol used in a master/server configuration [11]. According to the protocol specification [11] the protocol is placed in the application layer of the OSI model and is the de facto standard for industrial serial communication. The Modbus RTU protocol allows for a request/reply protocol and its controlled via function codes sent in the communication [11].

EtherNet/IP An industrial protocol, based on the IEEE Ethernet standard and TCP/IP, built with the goal to be able to connect every type of device used in an industrial environment, including IoT devices [12]. Since it is based on the IEEE Ethernet standards the networks can be configured using any type of available connection option, such as fiber and wireless [12].

Modbus TCP/IP The same as the *Modbus RTU* above but it uses Ethernet for its transportation instead of serial connection [13], [11].

PROFINET PROFINET is an open standard for industrial communication, based upon Industrial Ethernet [14]. It has the capability to be switched just like IEEE Ethernet and allows for parallel operation with other Ethernet protocols. Similarly, to EtherNet/IP,

PROFINET is according to [14] built to be used with all kinds of different hardware.

However, there are other more modern protocols on the rise. One of these is a protocol called Open Platform Communications - Unified Architecture (OPC-UA) [15]. The developers of OPC-UA describe in [15] that it is a protocol with the aim of allowing all kinds of devices to talk to each other using a common communication protocol. The idea being that the device specific protocols is abstracted by an OPC system and the common OPC messages sent to a device are then converted back to the specific protocol used by the device. This allows a user, such as a manufacturing company, to mix and match equipment from different vendors for their production needs.

3.2 Overview of MES

In this sub-chapter an overview description of the MES is presented as a general concept. The general concept of how the MES works and how it is constructed is, similarly to the previous sub-chapter about PLCs, essential to understand the thesis.

3.2.1 Architecture

The architecture of an MES are made of modular pieces of subsystems [5] where each of the modules can be prioritized differently to fulfil the needs by the manufacturing company. Artiba, de Ugarte and Pellerin describes in [5] that most of these modules can be placed in one of the following layers.

Client/server This layer is described in [5] as the layer of which users can interact with the systems. Other systems can also be connected through this layer, enabling the different systems to communicate with each other.

Integration This is the main layer where all the information exchange between programs take place, often built using tools for standard distributed object [5].

Data management This layer is the service layer. It provides the services needed by the other two layers to work as intended according to Artiba, de Ugarte and Pellerin. The layer itself is built on a standard OS and databases for it to have stable long-term support. A service which the Data management layer is responsible for is e.g. network communications [5].

To place the MES in a real-world application an example continuing the example provided in the sub-chapter 3.1 will now be described. In the previous example a suction cup, connected to a PLC, was used to lift an object appearing in front of a motion sensor. The way that the MES comes into the picture is that it will be connected to the PLC in a way, more details about the connection is described in 3.3, and communicate with the PLC. The MES has many functionalities, as described

above, but one of these can be management of instructions intended for PLCs, which will be the case of this thesis. For the example, the MES can send an instruction to the PLC where it tells the PLC to lift the incoming object with a certain force. The PLC receives the instruction sent from the MES, and when an object has arrived and triggered the motion sensor it lifts the object with the specified force. After the action is completed the PLC responds to the MES with a confirmation if the operation executed as planed.

3.3 PLC to MES interconnection

The connection between programmable logic controllers and manufacturing execution systems can be connected through a few different ways. One way is to create custom in-house software that handles the connection between the PLC and MES and transforms the data to formats that both sides can understand. Another way to achieve this is to use an already existing framework, e.g. OPC-UA descried earlier. OPC-UA is platform independent and can be used to communicate between devices, e.g. PLCs, and software systems [15] [16], e.g. the MES.

3.4 Publication of vulnerabilities

Hardware and software vulnerabilities are found all the time, and often disclosed to the public in one way or another. One of these ways to publish a vulnerability is by requesting what is known as a Common Vulnerability and Exposure ID [17], or CVE ID for short. These CVE entries are published with information about each of the vulnerabilities, this often include e.g. a description of the vulnerability, an impact description and a severity score based on the impact description. This can then be used by researchers and others to keep track of what vulnerabilities has been found and learn from them. The CVE system has two different versions that is in use, version 2 and 3, and both scores the CVEs with an overall score based on the description of the vulnerability, as described above. This score is created using the Common vulnerability scoring system (CVSS).

3.5 Software testing

When testing a piece of software, for security, feature or any other reason, two different terms one can come across is black-box and white-box testing. These two terms describe how one would construct the test cases for a particular software. For black-box testing, the internals of the system tested is ignored [18]. So, for these kinds of tests one can not use e.g. the source code to target a specific path in the software. Instead of the internal information one can use e.g. the software specifications or requirements [19] [18] to construct the test cases.

The opposite to black-box testing is white-box testing. With white-box testing one uses the internal descriptions of the software to create the tests that one would like to perform [19] [18]. This internal description could e.g. be the source code of the software.

Chapter 4

Related Work

According to the best of our knowledge there does not exist any prior studies that addresses the possibility to alter or affect the MES by exploiting a PLC. There is however work done in the field of PLC exploitation and their vulnerabilities, which is something that is needed for this thesis. Following some of the work done in the field is presented, and in sub-chapter 6.1.1 even more related studies are described.

In the conference paper [9] Milinković and Lazić describe that the OSs used in PLCs has vulnerabilities just as "normal" computer OSs such as Linux, but some of them might be less known to the public. They continue saying that some vulnerabilities found in PLCs includes open ports, weak hashing implementations, brute-forcible authentication or being completely open to the public network. Milinković and Lazić also argues that the general security goal for IT is not the same as for ICSs, for general IT the top priority is confidentiality while in ICS it is availability. The solution to these problems is, according to Milinković and Lazić, to use standards which define communication zones for the ICSs as well as to separate the communication from the rest of the network using firewalls and virtual private networks. They finally conclude that ever since the infamous Stuxnet [6] malware was publicly announced, PLCs has gotten a much bigger attacker base and it is time to protect the industrial controls.

In the paper [8] Basnight et al. argues that the state of ICS security is not as sophisticated as the technology and tools used in general IT security. Basnight et al. continues by describing the work needed to identify how the PLC firmware is validated when it is installed. They use binary comparison, reverse engineering, black-box testing and hardware debugging when they try to identify the validation. A walkthrough of a simple firmware modification is then presented by Basnight et al. and they conclude that malware can be implemented into firmware with relative ease.

The conference paper [20] shows that attacking a PLC without anyone knowing about it is possible. Abbasi et al. presents a novel attack where one can modify the pin configuration on the PLC without any interrupts in the OS or similar that could have notified an operator. Given a certain number of requirements to the attack, Abbasi et al. envisions that the "Pin Control attack" is the most effective way to maximize the physical damage done to a process. In their own attack implementation Abbasi et al. managed to timely reconfigure output and input pins in such a way that the PLC thought everything was OK, but the "attackers" could make the PLC output any desired value while at the same time read arbitrary values. Abbasi et al. later discusses that their attack could be suitable to disrupt a water distribution

plant and while there are countermeasures to the attack all of them have issues.

The above described papers, and the papers described in sub-chapter 6.1.1, describe in different ways how one could attack or exploit a PLC. The goal for many of them is to disrupt the process that the PLC controls but no one of these examines if their attack creates the opportunity for further attacks upwards in the automation system hierarchy. In this thesis this identified research gap is examined, especially attacks targeting the MES, since it could provide important and relevant knowledge to the research community as well as to the companies using these systems.

Chapter 5

Method

In this chapter the methodology used to answer the research questions are described. Each of the research question has its own sub-chapter where the methodology for that specific question can be found. A detailed and descriptive methodology is essential for reproducing the test results if someone else is curious to validate the results presented in the thesis. It does also provide an insight in how the research has been performed.

5.1 Research question RQ1

RQ1: What different types of PLC attacks have been identified between 2010 and 2018 and what general trends can be observed in the different types?

To answer the first research question, RQ1, a literature study was performed to find the work done, both academic and other, regarding PLC attack types, PLC vulnerabilities and exploits written targeting PLCs. In the following sub-chapters, the methodology to find this information is described. The first sub-chapter describes the process to find the academic work and research done within the field of PLC attacks, exploits and vulnerabilities. The second sub-chapter describes the process to find the existing and published vulnerabilities that commercial PLCs suffer from. The third and final sub-chapter describes how known and already published PLC exploits have been collected.

5.1.1 Academic literature review

A literature review [21] [22] was performed to find the academic work done regarding attacks, exploits and vulnerabilities in PLCs. The information was collected using the Summon database search engine [23] provided by Blekinge Institute of Technology. This search engine allows one to search for academic papers in numerous databases such as Springer and IEEE Xplore.

The search string used to gather the necessary papers is the following,

("programmable logic controller") AND ((exploit) OR (attack) OR (vulnerability)).

These results were then filtered using the following criteria. The text needed to be *Full Text Online*, the content type had to be *Journal articles*, *Conference Proceedings* or *Publications*, its content had to belong to the disciplines *Computer Science* or *Engineering*. The publication date was limited to between 2010-01-01 and

2018-10-31 and the language used in the paper had to be English. The search string and the filters resulted in a match count of 614 documents. To get the number of papers to a more manageable number the following series of steps was taken, each of which discards irrelevant papers.

1. Read the title of each of every one of the papers and evaluating if it was describing anything that could be of interest, with the regards to PLCs and attacks. After this step 84 papers were left.

2. Read the abstract of the remaining papers. By reading the abstract a better understanding of the papers content could be achieved, and irrelevant papers that was not describing PLC specific attacks or vulnerabilities was discarded. After this step 37 papers were left.

3. The remaining papers was then skimmed to see more what the paper was about and if it still was considered as a relevant paper, in the sense that it described one/several attacks or vulnerabilities specifically targeting a PLC. Also, by word searching in some of the papers one can quickly find the context where, and if, the keywords such as PLC was being used. After this step 13 papers were left.

4. The papers remaining after being skimmed was then more thoroughly read and the last of the irrelevant papers was filtered out. As with the previous steps, a relevant paper would describe a PLC specific attack or vulnerability.

These steps resulted in that the 614 papers, that matched the original search string and filter criteria, was filtered down to a paper count of 9 relevant papers. These papers can be found in the References chapter, [24–32]. As an additional step to increase the number of relevant papers a bit, a couple of more steps was taken. The reference list of the papers that was skimmed through was read and by using the same steps, as described above, these references was filtered and the relevant once was added. This resulted in 10 additional papers, [6, 9, 33–40] in the Reference chapter.

5.1.2 Existing vulnerabilities

The CVE entries used in the thesis were all collected from the National Vulnerability Database (NVD) [41]. NVD provides the option to download the database as JSON or XML files, one file for each of the years. Using these database files, a python program, found in appendix A, was written and used to process and automatically filter out most of the irrelevant entries. All the JSON files, named CVE-2002 to CVE-2018, was downloaded on November 9th and was then processed by the program. The program first checks the publishing date to see that the CVE entries was published between 2010-01-01 and 2018-10-31. Following the date check the program uses regular expression, with two word lists, to examine the affected vendor name and the CVE description for each of the CVE entries. All the CVE entries that matched, at least one of the words in the lists, was then saved of to a separate file.

The word lists used consists of one list with PLC vendors and one with additional words. The vendor list was created using [42] which lists 16 of the top manufacturers of PLCs. These vendors were then added to the vendor list. Two other websites [43] and [44] was also found listing PLC vendors. Some of the vendors listed in [43] and [44] was already found in [42] but there were some new ones as well, and these was also added to the vendor list. As an addition to the vendor list an additional word list was also used. This list contained other words that could add some more matches in the case that the vendors did not match. The list with additional words mostly consisted of some PLC series names or model names. The final word lists used to filter out CVE entries is the following.

Vendors siemens, abb, schneider, schneider electric, rockwell, rockwell automation, mitsubishi, ge, general electric, ge-fanuc, omron, koyo, panasonic, idec, keyence, toshiba, fuji, beckhoff, bosch, rexroth, bosch rexroth, allen, allen bradley, allen-bradley, hitachi, delta, honeywell, yokogawa, b&r industrial automation, b&r, philips, philips components, festo, kim controls, kim, horner electric, horner, see automation & engineers, see

Extra plc, programmable logic controller, progammable-logic controller, simatic, ac500, modicon, micrologix, melsec, smartguard, directlogic, kostac, microsmart, smartrelay, opennet, smartaxis, kv, micrex, twincat, masterlogic, controledge

The output of the python program was a list containing 1197 unique CVE entries and their description. This list was then filtered by hand. First the description was skimmed through to see if any of the product names or similar was familiar and could quickly be placed in either the relevant list or in the irrelevant list. If there were no familiar words in the vulnerability description an internet search for the affected product(s) was done. By looking at the product pages by the product manufacturer or by reading the first few matches from the internet search one could often identify what type of product it was. However, some did require some more digging like reading a few other web pages to look for information.

The total number of published CVEs was also collected, this was done so one can compare the changes on the global scale to the PLC related CVEs. This information was collected using the same database files as mentioned above, but this was done using another small custom program. This program went through each database file and counted the number of CVEs published each of the years between 2010-01-01 and 2018-10-31.

5.1.3 Existing exploits

When someone doing research on vulnerabilities in software or hardware it is not uncommon to disclose proof-of-concept code or exploits to show it and so others can try it out. One of these sites that one can submit an exploit to is Exploit Database [45] created by Offensive Security, which is the creators of the famous penetration/security testing OS Kali-Linux.

To find the already published exploits the site [45] was used. The website allows a user to download the entire Exploit database with an accompanying search utility called SearchSploit. By using SearchSploit (downloaded November 15th) in a bash script, found in appendix B, together with the same words as when searching for CVE entries, described above, all exploits whose title contained any of the words in the word list was extracted to JSON files. These JSON files was then processed by a python script, also found in appendix B, which further filtered the exploits and printed them. By manually going through the printed exploit titles the irrelevant entries were filtered out. To determine which one was relevant the title of the exploit was read and if that alone was not enough an internet search was used. Firstly [45] was used but in the case of when it was not enough a general internet search was used. Finally, the exploits remaining was checked so that the published date was between 2010-01-01 and 2018-10-31, as with the CVE entries and the academic work, and the ones that could be placed in the time range was the final and relevant exploits needed for the thesis.

The total number of published exploits was also collected, so that one could compare the curve of the PLC related exploits and the total number of exploits published each year. This data was collected using the statistics page for Exploit Database [45], then specifying the time range to 2010-01-01 and 2018-10-31. The number of published exploits for each month was then summed up to get the total number of exploits published each year.

5.2 Research question RQ2

RQ2: To what extent is it possible to negatively affect a MES security-wise by exploiting a connected PLC?

In this sub-chapter the experimentation phase needed to answer the second research question is described. Firstly, the experimentation setup is described followed by the steps taken to attack the MES.

5.2.1 Experimentation setup

The experimentation conducted took place in a lab environment. The lab environment, as seen in Figure 5.1, was constructed with a few different components. The components used are the targeted MES, an equipment broker and a PLC. To these components a computer was connected to be used for different tasks. The computer could read the logs from the MES and equipment broker systems as well as write and read the content in the memory of the PLC.

The targeted MES was a system that is used to send instructions to an already programmed PLC. A simple example of this system could be e.g. that the MES tells the PLC to run instruction "A", and the PLC then runs the pre-programmed instruction named A. The equipment broker, placed in between the MES and the PLC, is responsible for converting the traffic going to and from the PLC to a protocol readable to the MES, and while the data is converted it evaluates it to see if it breaks already defined constraints in place.

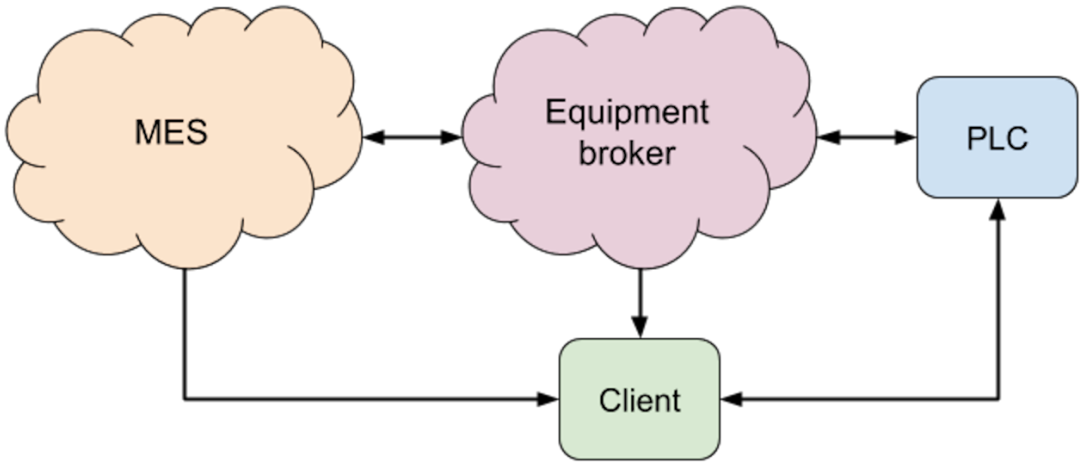

Figure 5.1: The setup used for the practical experiment.

The attacks targeting the MES was conducted by writing to the memory, existing in the PLC, where the instructions sent from the MES is placed as well as where the PLC writes the result for the given instruction. This memory field was, in this setup, of a size of eight bytes. These bytes were split into three different sections based on how it is used by the equipment broker and the PLC. The sections used in the memory field is a one-byte field for the status of the instruction, a two-byte field to hold the length of the written data and lastly a four-byte field for the data itself. The last byte of the eight in the memory field is placed in between the status field and the length field and acts as a padding to align the data.

Given the attack surface of the experiment all the attacks had to be based on manipulation of binary data in one way or another. A few different types of attacks could be achieved from this. This first attack could be a buffer overflow attack, this would occur if the length written in the length field is bigger or smaller than expected and if the systems does not check memory boundaries correctly. The second kind of attack could be a fuzzing or invalid data attack, where the data written into any of the fields could break or crash the system if not handled properly. A third attack type that could be plausible would be an arbitrary code execution. The arbitrary code execution does however depend on a successful buffer overflow attack and that it is possible to write more data to the PLC than is specified in the memory field.

The attacks performed in the lab environment targeting the MES was a black-box attack approach [19]. The reason being that the protocols used between the PLC - equipment broker and equipment broker - MES was known as well as how the overall system was meant to operate in normal use. The internal workings, e.g. source code, of the equipment broker and MES was not known however thus not making it a white box approach to the attacks.

5.2.2 Experimentation steps

To test if the MES is susceptible to the attack types listed above a structured testing approach was used. Starting with two correctly stated responses for the given instruction, the different fields of the correct responses was modified to see if the modifications would crash or in any other way affect the MES in a negative fashion. The reason for having two different responses is that the two systems expects either an "OK" or "Error" response, so both paths need to be examined. The tests were done in sequence, each of which handled a different part of the memory field described, by performing the following steps.

Status field The first field in the memory area is as mentioned the status field. There might be other values, other than the "OK" and "Error" values, that could work so a few different values was tested to see if they were approved or not. An incorrect value that is accepted could potentially break something in the receiving systems.

Length field Once the accepted status values were decided the next step was to test the length field. Using the approved status values found in the previous test, the length value for each of the status values would be modified to see the affect it would have. Modifying the length value could potentially have broken the MES and/or the equipment broker if the memory boundaries is not enforced correctly.

Padding byte The padding byte was targeted after the length values was decided. Using both the approved status and length values the padding byte was modified to see the affect on the MES and equipment broker.

Data field Lastly after all the other steps was completed the data field was targeted. Here any data could be sent to the MES and equipment broker to see if it would break anything in the MES.

5.3 Research Question RQ3

RQ3: Which methods are suitable for protecting the MES against those attacks?

The goal of the third and final research question, RQ3, was to describe already existing techniques or discuss techniques that one could apply to protect the MES against the attacks that were successful in the second research question, RQ2. However, as presented in sub-chapter 6.2, none of the attacks tested was successful in the sense that none of them managed to introduce negative behaviour or consequences in the MES. So instead of providing descriptions on specific mitigation techniques a short and more general discussion and reasoning, from our point of view, on attack mitigation and protection of the MES will be presented.

Chapter 6

Result and Analysis

In this chapter the results for each of the research questions will be presented and analysed in their own sub-chapter.

6.1 Research question RQ1

RQ1: What different types of PLC attacks have been identified between 2010 and 2018 and what general trends can be observed in the different types?

In this sub-chapter the results of the first research question are presented. The presented results show the work done in the scientific world regarding PLC attacks, the types of PLC vulnerabilities that has been disclosed and how the number of published exploits targeting PLC vulnerabilities has changed. All theses topics has been limited to the years 2010-2018.

6.1.1 PLC attacks in research

The work that has been done in the academic world since 2010 regarding PLC attacks has used several different ways to accomplish their goal. The attacks can however roughly be placed in one of a few common categories. The following categories has been identified in the work done in the academic world.

- An attack that uses the network, that the PLC uses, to achieve its goal.
- An attack that uses the logic code of the PLC to achieve its goal.
- An attack that uses the underlying firmware/OS of the PLC to achieve its goal.

The attacks found in each of the categories uses a separate layer in the PLC as the target, as can be seen in Figure 3.1. The network attack targets the firmware/OS layer where the network communication is managed, the logic code attacks targets the application layer in PLC and finally the firmware/OS attack targets the hardware layer since the hardware is responsible for the execution of the firmware used by the PLC.

Network based attacks

The first category is the one category that has gotten the most attention from the research community. The network attacks differ in the vulnerability that they are exploiting. In the papers [24], [29], [28] the idea of different types of DoS attacks are described. In [24] a DoS attack, that most people would think of when hearing the word DoS, is described. This attack simply floods a port on the PLC with network packets. The other two papers, [29] and [28], has another approach to the DoS attack. The attack still floods the PLC, but the attack is timed in such a way that the data from peripherals does not reach the PLC. That forces the PLC to work with the last received data, and a well-timed DoS attack could then drive the process to an undesired state by reusing old data.

Another type of network-based attack is false data injection attacks. These attacks build on, as the name suggests, injecting false information into the network traffic of the process. Both papers [27] and [25] describes these attacks and in [27] the false data injection is used to disrupt the process by informing sub-PLCs that the process is in another stage than it actually is. A similar goal but executed differently is presented in [25], in this paper the goal of the false data injection is to hide the malicious work done in another part of the process. By injecting data in the network that shows that the process is staying within the limits of what is expected the actual attack is performed in the dark changing the state of the process. A similar but very different attack to the false data injection attack is the false logic attack. In a false logic attack the goal of the attack is to issue incorrect commands to the targeted PLC to make the PLC drive the process into an undesired state. This is presented in both [27] and [38].

In the paper [34] more networked based attacks are described. Here the attack type called replay attack is used and described. A replay attack is an attack which uses pre-recorded legitimate network traffic and sends it to the targeted device for an action to take place. In [34] this replay attack is described to be capable of several different attacks, e.g. change the PLC password and lock out the operators, fingerprinting the PLC and get information about the PLC system and running program, or read and write to the PLC memory.

In the paper [9] there are three short descriptions of different attacks that uses special packets to attack a PLC over the network. The first one of them forces the CPU of the PLC to stop, the second dumps the boot code of an Ethernet card used by the PLC and the last of the attacks resets the entire Ethernet card.

Many of the PLCs in use uses the same kind of library that executes the logical program written by an operator [40]. This library has a network flaw that allows an adversary to remotely read and write logic code from the PLC, and the flaw can also be utilized to write or read any file existing on the PLC [40].

Logic code based attacks

Logic code attacks uses the logic code, that is written by an operator to drive the process that the PLC is in control of, to achieve the goal of the attack. Stuxnet, which is probably the most infamous and known attack on automation systems, belongs to this category of PLC attacks. Stuxnet used the logic code in the infected PLC

to monitor and control an industrial process [6]. But Stuxnet is however not alone with the use of logic code to achieve an attack. The papers [26], [36], [33] and [32] all uses the same idea. In [26] the proposed idea is that logic code can be used by an adversary to perform data logging or creating a timed or remotely triggered DoS attack inside of a PLC. The paper [36] demonstrates that the logic code is capable enough to be able to implement a SNMP scanner and a SOCKS proxy. The last two of the mentioned papers takes the approach of malicious logic code a bit differently. In [33] and [32] the idea is to make a program automatically write the logic code payload needed to disrupt the process. This program achieves this by using an adversary written process specification and by analysing the internals of the PLC that the attack is targeting.

Firmware based attacks

The next category of the ones listed above goes even deeper than the previous category of attacks, that was just mentioned. These attacks utilize the underlying firmware of the PLC to perform the wanted attack. The papers [30], [31], [39] and [35] all describes different ways that the firmware can be used for malicious activities. The general approach to reverse engineer and identify the firmware verification algorithm is provided in [31]. This is then later used in [30] to embed three different types of DoS attacks into the firmware of a PLC without affecting the original PLC operation, except when the DoS attack is active. In [39] the implementation of a PLC I/O rootkit is described. The result being a rootkit which can intercept and change any data used in the logic program that is sent to and from a PLC. The last paper [35] uses the same idea as the previous mentioned works, but instead of targeting the PLC itself the work targets the Ethernet card that the PLC uses. The use cases for this type of an attack is described using several examples, e.g. crashing the Ethernet card or use it to spread a worm.

An attack like the I/O rootkit mentioned in [39] can be found in [37]. The attack described in [37] describes two different attacks that can be used to manipulate the I/O configuration of a PLC. By changing the configuration of the I/O pints the attack in [37] can block outgoing data and change incoming data from peripherals. This is achieved by either injecting a custom hardware intercept handler or by injecting a program that can manipulate the internal configuration of the I/O pins of the PLC.

6.1.2 Disclosed PLC vulnerabilities

The work done regarding the topic of PLC attacks and vulnerabilities are however not limited to the work of academic institutions. There are other people, communities and organizations that also looks for these. The vulnerabilities found are often disclosed in some vulnerability database where other researchers or curious people can read about them. There are 160 different CVE entries published since 2010 regarding PLCs[1]. As one can see in Figure 6.1 the number of found and disclosed

[1]Of these 160 CVE entries there are six entries that at the time of writing had not been evaluated, these six entries are among the most recent ones found. Since they have not yet been evaluated they are missing both CVSS score, complexity and vulnerability type. As a result, these has not been included in any of the figures other than Figure 6.1 and Figure 6.5

PLC related CVE entries has increased every year since 2010, apart from a minor drop in 2015-2016. If one would compare the curve displaying the total number of CVEs published and the curve of PLC related CVEs, one can see that the PLC curve somewhat follows the changes on the global scale. When the number of published CVEs goes up so does the number of PLC related and the opposite when it goes down, however during 2010-2011 and 2017-2018 these two curves does not resemble each other. At the time of writing this, 2018 has not ended yet and with the increase that can already be seen in Figure 6.1 the number of found vulnerabilities will probably increase even more.

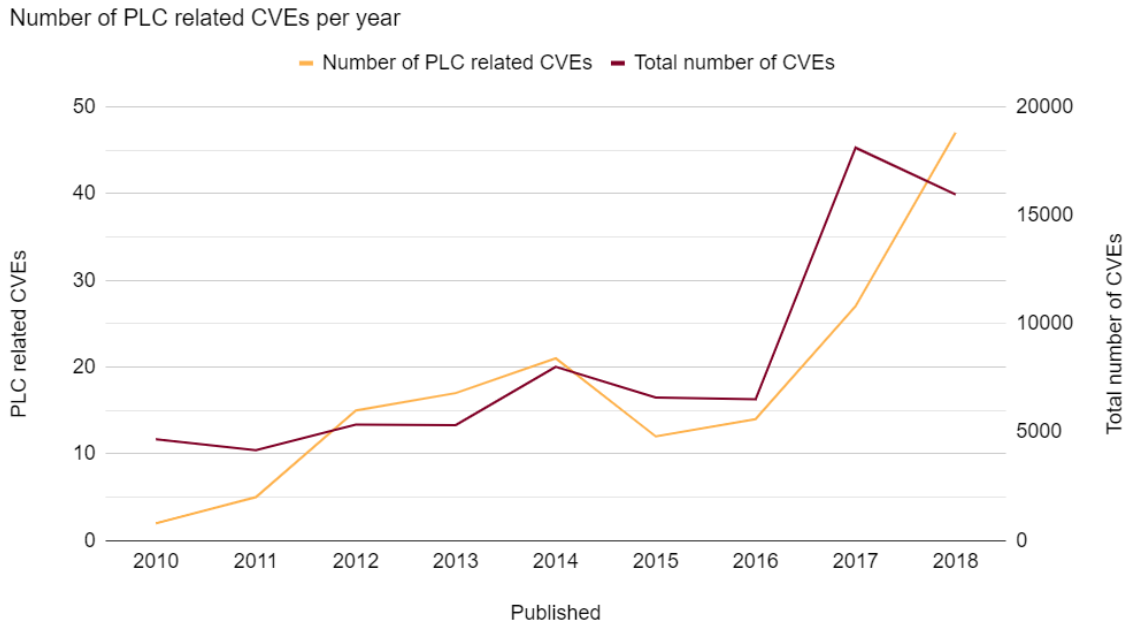

Figure 6.1: The number of PLC related CVEs and the total number of CVEs published each year.

In Figure 6.2 one can see that there are a number of different vulnerability types that have been found over the years. The one vulnerability that has been identified the most times over the years, and almost every year as well, is buffer errors. These vulnerabilities occur when the program can write and/or read outside the intended memory area. This could further be used by an adversary to perform a buffer overflow attack and execute arbitrary code. Other vulnerabilities that also have been found most of the years are resource management errors, authentication issues and access control. All the mentioned vulnerabilities affect some of the foundations needed for secure systems. To see that these kinds of errors re-appears is not a good sign. There are e.g. still in 2018 PLC systems that fails one of the most basic tasks like authentication, which is the first line of defence against attackers in a system.

By looking at the types of vulnerabilities that has been disclosed, Figure 6.2, one can see that the focus has shifted or broadened over the years. During 2010 there was only two different types found, the year after four types and after that nine different types. This increase in the number of types found is not present in all the years. But what is present in the figure is that there is always an overlap between the

research of the current year and the years before. There is not a single year where only completely new vulnerabilities were disclosed. This could be because a few of the researchers decides to postpone the disclosure of vulnerabilities to e.g. the year after the discovery. But another, and probably more likely, reason is that researchers learn from the work done by others and by using the work already created it is easier to test the idea on other devices. Easier tests also lead to more time left to explore other ideas or fields. The overlap could also indicate that the PLC vendors does not learn how, or prioritize, to develop devices that are secure from the beginning, without the need for patching the devices.

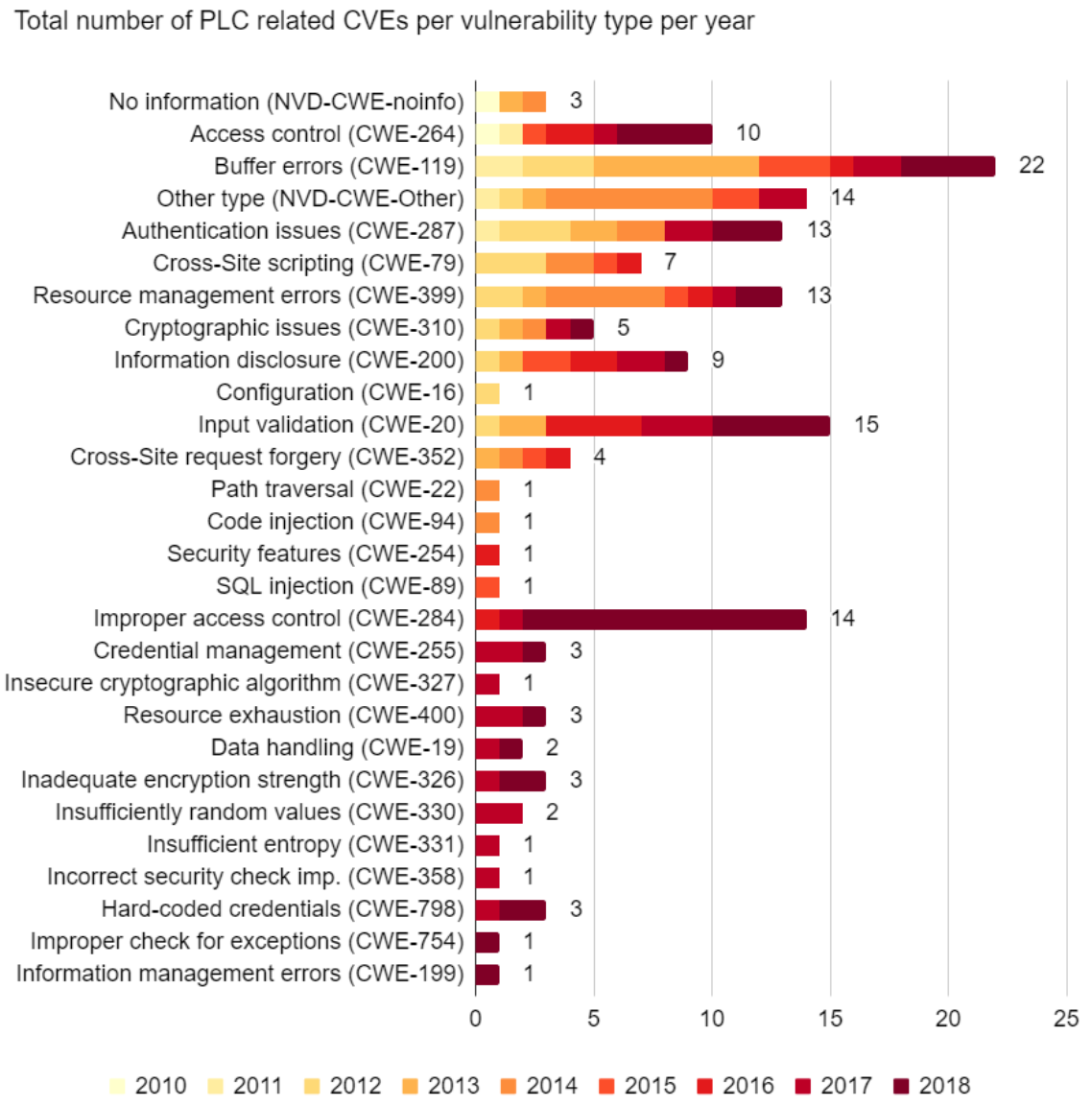

Figure 6.2: The total number of PLC related CVE entries for each of the vulnerability types found.

Additionally, one can see in Figure 6.3 that the mean complexity of the vulnerabilities found are quite low each of the years. Even though there are some variation in the complexity, it is not that big. Most of the vulnerabilities has a low complexity

which means that it is fairly easy to exploit the vulnerabilities found. The complexity in Figure 6.3 is represented by numbers instead of the string value that was originally assigned. A low complexity has been converted to the value 0.0, medium complexity (CVSS V2 only) to the value of 0.5 and lastly high to the value of 1.0.

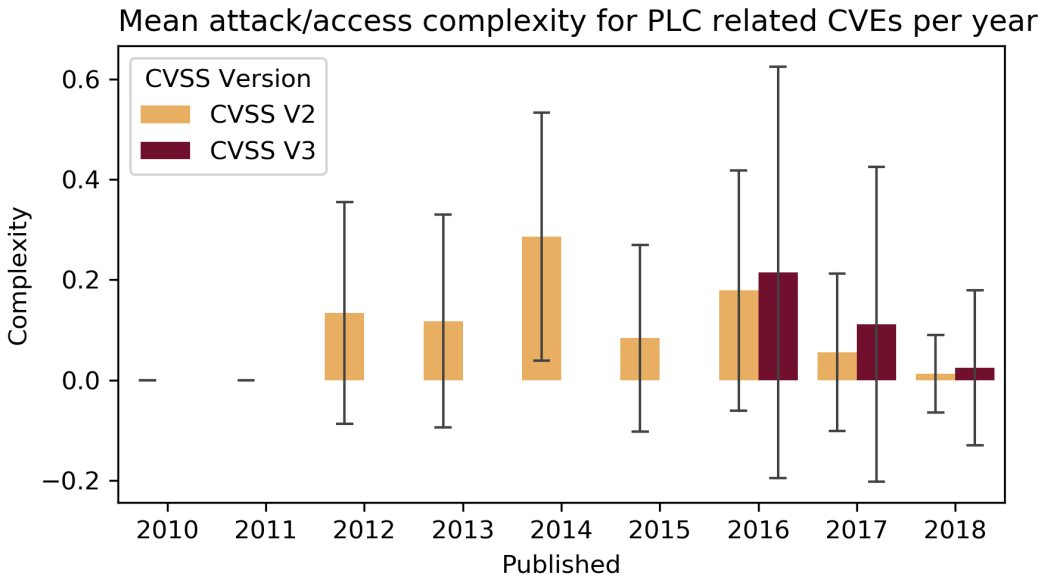

Figure 6.3: The mean access/attack complexity for PLC related CVEs per year. This is depicted as a bar plot with the mean complexity as well as the standard deviation.

Looking at the yearly mean CVSS score in Figure 6.4 one can see that the found vulnerabilities not only have a quite low complexity to perform but the amount of damage they can inflict is quite high as well, since the mean CVSS score is quite high most of the years. Even though there are some variation in the score each year, most of the vulnerabilities still has a score which is on the upper half of the scale, meaning that the vulnerabilities are quite severe. The first two years just a couple of entries were found, and most of these had a version 2 score of ten. But in 2012 the score dropped a little, most of them is placed a bit lower that eight and this trend continued until 2015. In 2015 the score is more spread out over the range and this continues, but with less and less spread, until 2018. One interesting thing that one can see is that the version 3 score for the entries during 2016 to 2018 is quite a lot higher than the version 2 score for the same year. So, if the version 3 score were used for all the years it is possible that the overall score could have become higher as well, since it seems like the version 3 system gives higher score than the previous version.

When looking at which of the PLC vendors that is affected by the vulnerabilities found, one can see in Figure 6.5 that there are a few vendors that has gotten the most attention. It is however worth noting that the vulnerabilities found could very well affect other vendors and products as well, depending on the source of the vulnerability, and not just the ones that is reported in the CVE entries and what is shown in the figure. But given the results shown in Figure 6.5 one can see that majority of the published CVEs related to PLCs are affecting products from Rockwell

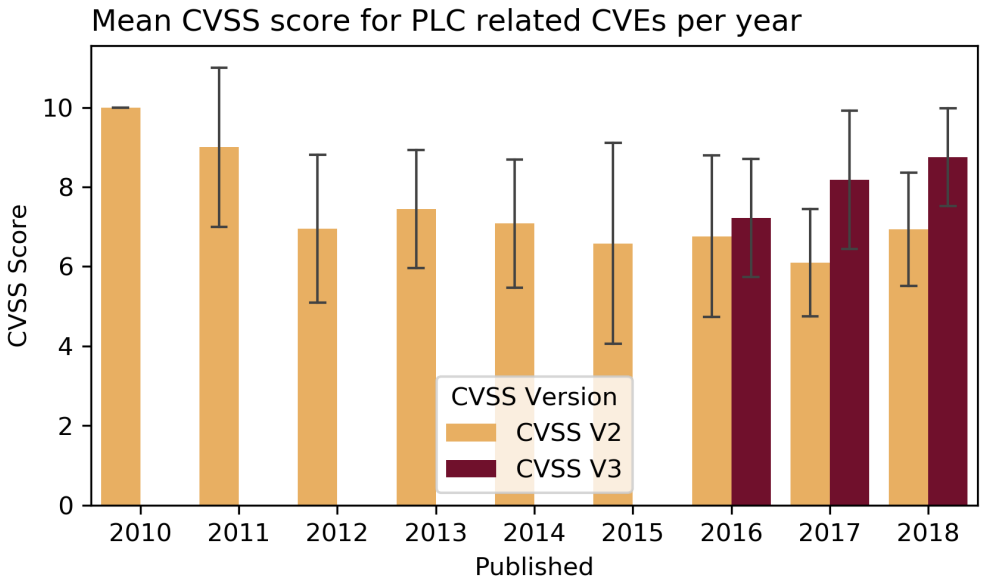

Figure 6.4: The mean CVSS score for PLC related CVEs per year. This is depicted as a bar plot with the mean complexity as well as the standard deviation.

Automation, Siemens and Schneider Electric. According to [44] these companies are all really big PLC vendors, or "Tier 1" manufacturers. This could be one of the reasons for why the three companies is affected by most of the CVEs, that it is more interesting to investigate products from the really big companies since a security flaw could affect many other using them. Another possibility is that the less known or popular PLC vendors are more secure than the larger and more popular vendors.

6.1.3 Published PLC exploits

To test the vulnerability that the CVE entries describes, researchers sometimes write and publish exploit code that others can read and test. The number of published PLC exploits is not huge, but there are 14 PLC exploits written and published since the start of 2010. As seen in Figure 6.6 the release of these exploits is not coherent but sporadic. The number of exploits increases little by little until 2012, then the yearly increase trend is gone. During 2014 when there were about 20 vulnerabilities disclosed there was no exploits published at all. Comparing the curve for the PLC related exploits and the total number of exploits one can see that the two curves does not strongly resemble each other. Both the total number and PLC related curves does decrease after 2012, however the sudden decrease in PLC related exploits after 2015 does not exist in the curve for the total number of PLCs. This probably means that the decrease in PLC related exploits is not because the attention to exploits decreased, rather that the interest to PLC exploits decreased. However, it seems like the number of exploits is on the rise during 2018 and as of now there are six exploits published.

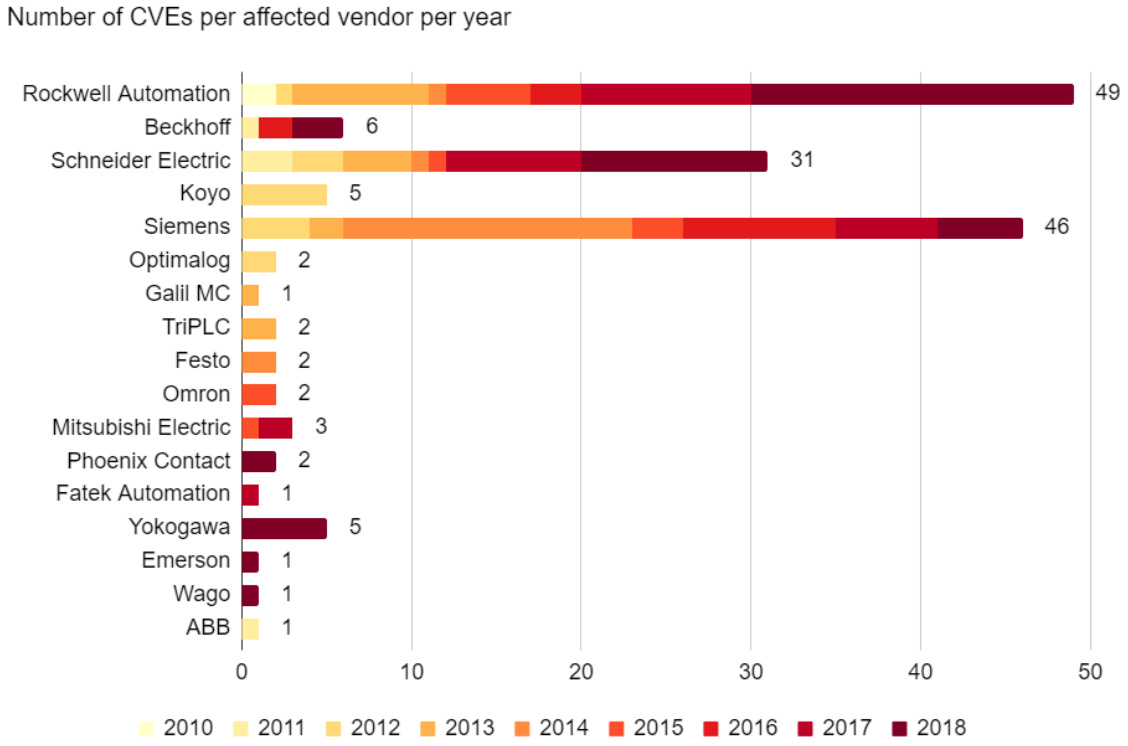

Figure 6.5: The number of CVEs per affected PLC vendor.

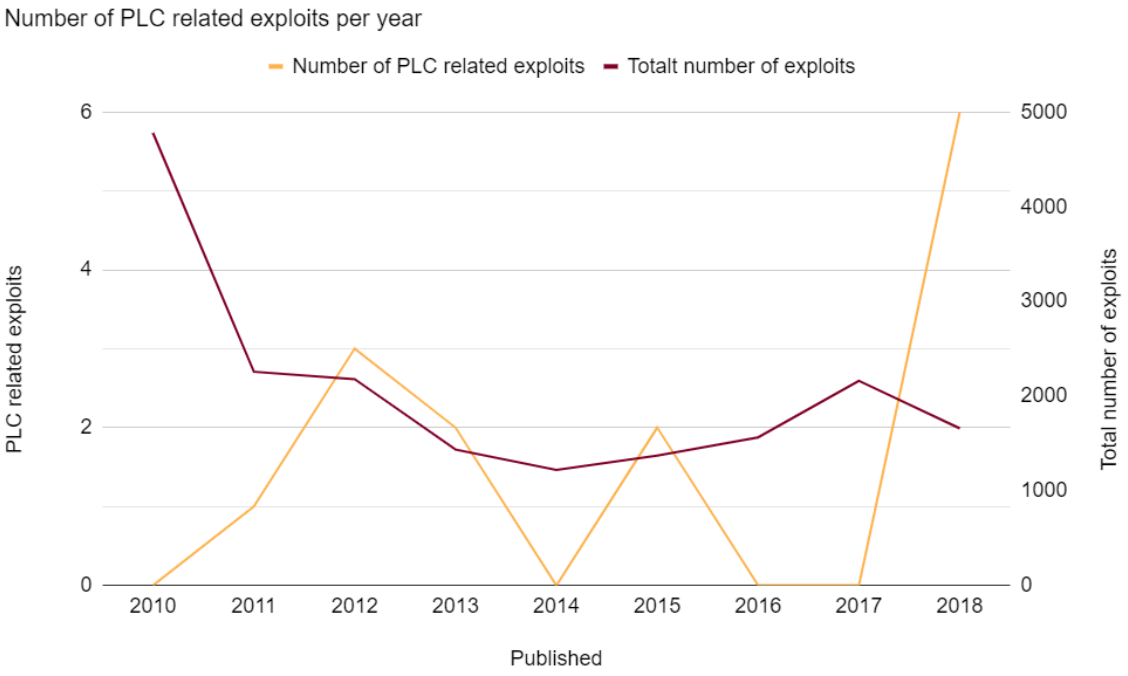

Figure 6.6: The number of PLC exploits and the total number of exploits published each year.

6.2 Research question RQ2

RQ2: To what extent is it possible to negatively affect a MES security-wise by exploiting a connected PLC?

In this sub-chapter the answer for the second research question will be presented. Each of the sub-chapters following describes the results for one of the experimentation steps.

6.2.1 Status field

As mentioned earlier in sub-chapter 5.2.2, the MES and equipment broker expects an "OK" or "Error" response from the PLC when a given instruction has been performed. To test and see if the two systems were susceptible to incorrect statuses a few other status values was written to the status field. However, none of these values affected the MES and equipment broker in any way, the systems simply ignored the PLC until one of the two correct status values was written.

These two status values, "OK" and "Error", was used to conduct the subsequent tests and is presented as in Table C.1.

6.2.2 Length field

As seen above the two status values that was accepted was "OK" and "Error", these was further used to test the length field. The tests, using both the "OK" and "Error" status values, was done in two parts. Firstly, smaller than expected lengths was tested, from the expected four down to zero, and after that larger length values were tested. Since most of the values in a byte is greater than four just three different values were chosen, the reason being that any of these values could cause a buffer overflow in the MES. The values chosen were the maximum value for two bytes, the maximum value for one byte and five. The length of five is the first value that could cause a buffer overflow, since the expected data length is four, and the other two values would be able to cause even greater buffer overflows than with the value five. The results, as can be seen in Table C.2, is that only the length of four and zero is seemingly accepted in the case of using the "OK" status value. For the "Error" status value only the length value of four is accepted by the equipment broker and MES.

Regarding the rows in Table C.2 where the results is *Seems to be accepted, no affect*, see sub-chapter 7.2 for more information.

6.2.3 Padding byte

Using the status values and the corresponding accepted length values the padding field was targeted. Since the field is a padding-byte the expected result is that it will not affect the operation what so ever. So, for this test only the padding values of zero and the maximum value for a byte, 255, was tested. The results presented in Table C.3 shows that padding does not affect the results.

Regarding the rows in Table C.3 where the results is *Seems to be accepted, no affect*, see sub-chapter 7.2 for more information.

6.2.4 Data field

As presented above, the padding does not affect the result of the response. Thus, the padding is left out from the test targeting the data field and since the objective was to send data to the MES and equipment broker the maximum data length allowed must be used. In this case it is the data length of four for both status values.

Before a structured testing approach was applied, a few tests were done with random data values using both the "OK" and "Error" statuses. This was done just to see what would happen when this field was modified. To begin with the status value was set to "OK" but after just a few tests it became clear that only results that was pre-configured was accepted by the equipment broker. Thus, any other values sent in an "OK" response was ignored. The "Error" responses on the other hand was sent through to the MES even though the data had been modified. These results lead to that there was no need to do more tests with the status of "OK" so instead the focus switched to responses with "Error" status. The subsequent tests with the "Error" status was done in a more structured approach, and firstly all the control characters existing in the ASCII table was tested. As can be seen in the Table C.4 where the data goes from the values of 0-7F, the only one that manages to affect the MES is 0A which is the hexadecimal value for a "newline". Using this data value four additional newlines was injected into the MES log, however no negative affects where noticed.

After all the control characters was tested a different approach was tested. Instead of trying to break the MES by passing characters that are not graphical characters the subsequent tests instead tried to break the MES in how the data is used. In the MES log the accepted messages is presented as XML trees. So, the goal was now to break the parsing of the XML tree itself. One of these tests was to inject a new XML tag into the log, with the data 3C 41 42 3E (<ab>), which could break the XML parse if the data is not handled properly. This did not work however since the characters < and > was escaped so that the parser could not read them as < and > characters.

The last tests that was performed was found in [46] where several different XML injection test cases are presented. But since the data field available to be manipulated is only four bytes the number of tests that could be performed was highly restricted. But in the tests that was done the following characters was tested: ‘, “, &, <!–,]]> and <\>. Neither of these tests was successful in the sense that it affected the MES in a negative way, as summarized on the last line in Table C.4, because the characters were escaped here as well.

Regarding the rows in Table C.4 where the results is *Seems to be accepted, no affect*, see sub-chapter 7.2 for more information.

6.3 Research question RQ3

RQ3: Which methods are suitable for protecting the MES against those attacks?

As described in sub-chapter 5.3, the prerequisites to answer RQ3 is not the result presented in sub-chapter 6.2. So, mitigation and protection mechanisms for specific attacks can not be presented. Instead a reasoning, from our point of view, on more general mitigation techniques will be presented.

To mitigate and protect the MES from attacks originating from the PLC there are two places where this protection could be deployed. This could either be added to the MES itself and/or to the PLCs connected to it. Starting with the hardening of the PLCs, the foundations must be in place. This includes tasks like proper authentication to verify that the person accessing the PLC has indeed permission to do so and access control so that the people accessing the PLC can not do anything that they are not supposed to do. These two points is mentioned in sub-chapter 6.1.2 among with other vulnerabilities that PLCs has. Buffer overflows are also mentioned and must be addressed to secure the PLCs. Here one must verify that the PLC is not vulnerable to overflows from both the inside of the system but especially from the outside, a buffer vulnerability could allow unauthenticated people to access the PLC even if the authentication is in place.

Logic code and firmware based attacks are presented in sub-chapter 6.1.1. These two types of attacks could prove to be capable to be further used in attacks against the MES. To mitigate and protect against these kinds of attacks the most important thing is that the authentication and access control is in place. It must be configured so that only the personal that must be able to update the firmware and/or logic code can do so. Also, validation of the firmware and logic code running would also increase the protection if one can see if improper software is being used.

As for protecting the MES the same as for the PLC applies. Authentication and access control must be in place. However, here the access control must also be placed on the internal subsystems of the MES. Reason being that the subsystems must not be able to access other files or similar if it would be compromised. Another very important part in protecting the MES against attacks is input validation and sanitization on everything received by the MES. As described in sub-chapter 6.2, the data sent to the MES was validated and the characters of the received data that could break the XML structure was rewritten so it would not happen. Without input validation the system could easily break if unexpected values were received. Lastly the communication between the PLC and MES must be secured. This is so that no one can intercept the network traffic, and if it would be intercepted no one should be able to read the information transmitted. This could be done using an encrypted communication channel and certificates or strong secret pre-shared encryption keys.

Chapter 7

Discussion

In this chapter the different chapters of the thesis will be discussed. In the first sub-chapter a discussion related to the methodology applied to answer the research questions will be described as well as how these methodologies affect the internal and external validity for the thesis. Secondly a discussion regarding the results will be presented.

7.1 Method

Here a discussion regarding the methodology used in the thesis will be presented. The methodologies used to answer all the research questions is discussed and presented in their own sub-chapters.

7.1.1 Research question RQ1

To the first research question a literature review was used. This was the most suitable approach to find the information needed regarding past attack attempts on PLCs in the research and security community. The approach to the literature study was structured from the beginning and a few different search strings was tested before the one used was decided. The steps that was used during the filtration of irrelevant papers was decided before the actual process and thus the number of relevant papers sorted out will hopefully be minimized. There is a possibility that there are relevant papers that are not included in the results, since the search string uses the full "programmable logic controller" and not the abbreviation papers only using the abbreviation has not been included. The abbreviation PLC is used for different things and in different research fields, adding this to the search string would have increased the numbers of irrelevant papers significantly as well as the time needed to filter these. Additionally, there were a few papers during the filtration steps that could not be read, due to the termination of the agreement between Elsevier and the Swedish research libraries.

As for the information collected regarding PLC CVE entries and PLC exploits, this was done mostly manually. Due to the lack of a general search feature in the databases used a few small programs was written to speed up the process. These programs sped up the process significantly, but the process has its down falls. Due to the structure of the databases used there are, as said, no general terms that one can use when searching for information and because of that one must search for a specific vendor or brand name to get the information. The word list used for the

thesis was constructed from the most popular PLC vendors according the sources used, but this might still have left out some results for vendors or brand names that were not included. Another step in the process that could have affected the result is that all the entries for CVEs and exploits that matched a word in the word lists had to be read and filtered by hand. The description of these CVEs is only a small compact piece of text with, mostly, only brand names, version numbers and the impact of the vulnerability. Reading and filtering these by hand could very well have led to that some relevant entries that matched were filtered out as irrelevant. Surely another program that could automatically filter out the irrelevant entries could have been possible, but the time needed to find all the relevant keywords would have been much greater than doing it by hand.

The internal validity of the first research question is impacted by the points made above. But we would argue that if additional attacks or CVEs were found it would not decrease the credibility of the results already presented. Additional information would just prove that the there are even more attack surface and that the situation has been even worse than originally presented. As for the external validity for the result, since the result is a summary of the attack surface on PLCs between 2010 and 2018, this information could be used everywhere where such information is needed. The programs written could also be modified and improved to search through the CVE and exploit databases to search for any other information and not just PLC related.

7.1.2 Research question RQ2

For the second research question a practical black-box experiment was chosen. A practical experiment would be the fastest way to see if attacks, trivial or complex, would be possible on the MES. Since there are no papers describing this attack path a simple proof-of-concept would be desirable to see how trivial these attacks could be. The attacks tested was a mix of buffer overflow and random data/fuzzing attacks, the reason being that these are some of the simpler methods one could break a system with. Would these attacks have been able to break the targeted MES a quite serious problem would have arisen. The result of the experiment could however indicate that these kinds of attacks is plausible since additional newlines were successfully injected into the log file. Even though we would argue that additional newlines are not a negative affect in the MES, it shows that this field of study needs to be further explored.

As for the testing methodology used during the experiment not all byte values and combinations in the memory field was tested. The reason being that the person who wrote the code for, e.g. buffer lengths, would most probably done some range check, i.e. checking if a number is greater and/or smaller than a limit, and not checking each number individually between the smallest and largest value. So, testing the edge cases, using as small or big values as possible, would be enough to see if it can crash. During the tests targeting the data field on the other hand the edge cases were not as interesting as previously. Here the data was interpreted as text in an XML structure, thus the goal was to break the parsing of the text and/or structure.

The experiments were also only tested with the same PLC-MES combination.

However, changing the PLC in the experiment setup would not have changed the results since the instruction format is enforced by the MES, thus any PLC that needs to communicate with it needs to comply with the specifications. Changing the MES on the other hand could have changed the results, but it could also have resulted in the same results. Since I have not tested another MES and no previous work has explored this topic it is impossible to say how other MES systems would have responded to the attacks.

The experiment was done in a structured way in a closed lab environment, and because of that there would not be that much that could impact the internal validity. All combinations could have been tested, but the result would most probably not change. As for the external validity on the other hand, since the systems used for the thesis are proprietary in-house systems the results will not be applicable on all other MES systems available, commercial or otherwise. However, would the experiments been successful in the sense that an attack induced a negative behaviour in the MES, it could have shown that the attack path is possible. That result could then inform the MES creators and foremost the companies using these systems that there exist attack paths between the PLCs and their MES.

The plan for RQ2 was originally to test the PLC attacks found in RQ1 in practice and try to leverage these attacks to attack the MES. However due to the time constraint that exist for the thesis and resource limitations that existed it was not plausible to try out these attacks in practice. The next best thing to test was the experiment presented here in the thesis. Using the approach presented in sub-chapter 5.2 most of the attack types originally planned could be tested anyway, i.e. buffer overflow, invalid input/data and if possible arbitrary code execution. The attacks found in RQ1, e.g. firmware modification and logic code attacks, would however quite probably been a lot more powerful given the capabilities described. So other more advanced attacks could have been tested given that enough time to implement them was available.

7.1.3 Research question RQ3

The third and final research question, RQ3, was not answered as planned. No successful attacks were found so a more general reasoning about protection from our point of view has been presented instead. The result in of itself does not contribute to any new information regarding securing systems. But it does point interested readers to some mitigation and protection techniques that could be applied. The validity of the result, both the internal and external, is impacted by the fact that the written research question is not answered. But the reasoning provided regarding protection techniques is applicable to more devices and systems other than the PLC and MES. So, the external validity of the result would be improved a small bit.

7.2 Contributions

As can be seen in the figures and the results presented for the first research question, RQ1, there are a lot of different types of attacks and vulnerabilities targeting PLCs. Once can also see that most of the found vulnerabilities affects a few of the most

popular PLC vendors, which could indicate that most vulnerability research has been done targeting these bigger vendors. We would argue that the most critical vulnerabilities of the ones found are the lack of weak authentication and access protection control. These provide, as already stated, a part of the foundation to building a secure system. These vulnerabilities combined with the power presented with firmware and logic code-based attacks creates a great security concern if an adversary would gain access to the network connected to the PLC, or physical access for that matter. With a firmware modification attack, one could do what ever one would like to, given the knowledge and the time needed. One could imagine the injection of a full and complete proxy or a command and control server, controlling botnets elsewhere on the internet. Proper authentication and access control could probably mitigate the risk of uploading unauthorized firmware/logic code and installing it on a PLC.

In the results presented for the second research question RQ2 there are many rows in the tables that has the result *Seems to be accepted, no affect*. For these tests it seems like the data written to the PLC has been forwarded by the equipment broker to the MES, but no logs related to receiving these could be found during the time of testing. However apart from no logs was found at the time no negative impacts could be observed either, thus making the attack unsuccessful.

Chapter 8

Conclusions and Future Work

Here the conclusion for the thesis is provided, as well as what the contributions of the thesis are and the future work that can be explored in the same field of study. Given the results there are two PLC attack types, firmware based, and logic based that are most likely to be able to leverage the PLC into attacking the MES layer above it. The firmware-based attacks seem like the most plausible but there are many PLC vulnerabilities of different types that has been disclosed and it is possible that some of these could be used to attack the PLC in another way as well. The practical experiment done did not result in a proof-of-concept attack that could affect the MES in a negative way. However, additional newlines were indeed injected into the log file of the MES. So, it does not rule out that these attacks could work using different approaches to leverage the PLC, other MES versions, or other attack techniques applied targeting the MES.

The contribution of this thesis is two-fold. Firstly, a summary of the PLC related vulnerabilities that has been disclosed between 2010 and 2018 has been provided. The summary shows both how the growth and interest in PLC vulnerabilities has changed during the years since the disclosure of Stuxnet, see Figure 6.1 and 6.6. The summary continues by showing how the distribution of these PLC related vulnerabilities varies, both in terms of what type of vulnerability that has been disclosed (see Figure 6.2), their complexity and impact according to the CVSS (see Figure 6.3 and 6.4) and finally which PLC vendors that are affected by the published vulnerabilities (see Figure 6.5. The summary also describes the type of PLC related attacks that has been explored in the research community during the same period. The most dangerous attack type according to us, based on the flexibility and plausible impact, but also the most complex attack is the firmware modification attack, which would make the adversary the owner of the PLC. Secondly, the thesis also describes a practical experiment where the targeted MES was attacked. The attacks tested were not successful, but the results could indicate that attacking the MES using the normal data path between a PLC and the MES layer might be possible.

A work that would enhance theses studies that handles the vulnerabilities disclosed for a type of device, would be a reworked vulnerability database. A database that would have classified each of the vulnerabilities to more general "tags" and allow the search for these tags would improve the studies that need this information. The ability to search for e.g. "programmable logic controller" and only get all the vulnerabilities related to it instead of having to search for vendors and vendor specific brand names, would decrease the time and effort needed to get the results sought after.

As for further studies that examines the same field as what has been explored in this thesis, the following points are work that could be of interest.

- Examine if the two identified PLC attack types, firmware modification and logic code based, can be used to exploit a PLC in such a way that one can reach the MES.
  - If it is possible to reach the MES using these two attacks, would it possible to negatively affect the MES by implementing attacks that leverages those PLC attacks?
  - Would it be possible to reach other network resources from the PLC leveraging the same PLC attacks?
  - Would it be possible to access other open ports on the MES? If so these ports might be susceptible to similar attacks tested in this thesis.
- Examine if other MESs are susceptible to the attacks tested in this thesis.
- Examine if the physical connectors on a PLC could be exploited to reach the MES, e.g. USB ports or Ethernet ports.
- Examine how one could design the PLCs for them to be less vulnerable to the vulnerabilities presented in this thesis.
  - Could it be possible to create a proof-of-concept version of a more secure PLC system? Open source maybe?
- Examine how the vulnerability research is done.
  - Is the research done targeting the more popular PLC vendors? Or is the PLCs from the less popular vendors more secure?

As seen in Figure 6.1 there is a drop in the number of published PLC related CVEs during 2015-2016. One can also see that the curve showing the total number of CVEs also drops a bit during these two years. But an examination of why the number of PLC related CVEs dropped would be interesting, is the drop caused by the fact that the devices was more secure these two years, or was there just fewer tests done examining PLCs?

References

[1] Advanced Micro Controls Inc., "What is a PLC?." Online. Available at: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/, accessed: 2018-10-04.

[2] J. L. Fauci, "PLC or DCS: selection and trends," *ISA Transactions*, vol. 36, no. 1, pp. 21 – 28, 1997.

[3] Trend Micro, "Industrial Control Systems." Online. Available at: https://www.trendmicro.com/vinfo/us/security/definition/industrial-control-system, accessed: 2019-02-02.

[4] Critical Manufacturing, "What is MES?." Online. Available at: http://www.criticalmanufacturing.com/en/critical-manufacturing-mes/what-is-manufacturing-execution-system, accessed: 2018-10-04.

[5] A. Artiba, R. Pellerin, and B. S. de Ugarte, "Manufacturing execution system – a literature review," *Production Planning & Control*, vol. 20, no. 6, pp. 525–539, 2009.

[6] E. Chien, N. Falliere, and L. O. Murchu, *W32.Stuxnet Dossier*. Symantec, www.symantec.com, Feb 2011. Version 1.4.

[7] I. Ahmed, S. Obermeier, V. Roussev, and S. Sudhakaran, "Programmable Logic Controller Forensics," *IEEE Security Privacy*, vol. 15, pp. 18–24, November 2017.

[8] Z. Basnight, J. Butts, T. Dube, and J. Lopez, "A Firmware Verification Tool for Programmable Logic Controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76 – 84, 2013.

[9] S. A. Milinković and L. R. Lazić, "Industrial PLC security issues," in *2012 20th Telecommunications Forum (TELFOR)*, pp. 1536–1539, Nov 2012.

[10] AutomationDirect, "PLC Communications | Coming of Age." Online. Available at: https://library.automationdirect.com/plc-communications-coming-of-age/, accessed: 2018-10-09.

[11] Modbus Organization, *Modbus application protocol specification*, Apr 2012. V1.1b3.

[12] ODVA, "EtherNet/IP." Online. Available at: https://www.odva.org/Technology-Standards/EtherNet-IP/Overview, accessed: 2018-10-10.

[13] Real Time Automation, "Modbus RTU." Online. Available at: https://www.rtaautomation.com/technologies-2/modbus-rtu/, accessed: 2018-10-10.

[14] PROFIBUS & PROFINET International, *PI White paper:PROFINET - The solution platform for process automation*, Jun 2015. English version 2015.

[15] OPC Foundation, "What is OPC?." Online. Available at: https://opcfoundation.org/about/what-is-opc/, accessed: 2018-10-17.

[16] OPC Foundation, "Unified Architecture." Online. Available at: https://opcfoundation.org/about/opc-technologies/opc-ua/, accessed: 2018-12-06.

[17] Mitre.org, "CVE - Common vulnerability and exposure." Online. Available at: https://cve.mitre.org/, accessed: 2018-12-06.

[18] T. M. Thomas, T. Badgett, C. Sandler, and G. J. Myers, *The Art of Software Testing*. John Wiley & Sons, 2 ed., June 2004.

[19] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2008.

[20] A. Abbasi, M. Hashemi, E. Zambon, and S. Etalle, "Stealth Low-Level Manipulation of Programmable Logic Controllers I/O by Pin Control Exploitation," in *Critical Information Infrastructures Security* (G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, eds.), (Cham), pp. 1–12, Springer International Publishing, 2017.

[21] G. Paré and S. Kitsiou, *Handbook of eHealth Evaluation: An Evidence-based Approach [Internet]*, ch. 9.2. Victoria (BC): University of Victoria, Feb 2017. Available at: https://www.ncbi.nlm.nih.gov/books/NBK481590/, accessed: 2019-01-15.

[22] University of Bedfordshire, *Writing the method section of a systematic literature review in a dissertation*. Available at: https://lrweb.beds.ac.uk/__data/assets/pdf_file/0005/502169/Writing-the-method-section-of-a-systematic-lit-review-in-a-dissertation.pdf, accessed: 2019-01-15.

[23] "Summon@BTH." Online. Available at: bth.summon.serialssolutions.com, accessed: 2018-12-03.

[24] T. Alves, R. Das, and T. Morris, "Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers," *IEEE Embedded Systems Letters*, vol. 10, pp. 99–102, Sept 2018.

[25] W. Li, L. Xie, and Z. Wang, "Two-Loop Covert Attacks against Constant-Value Control of Industrial Control Systems," *IEEE Transactions on Industrial Informatics*, 2018.

[26] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On Ladder Logic Bombs in Industrial Control Systems," in *Computer Security* (S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J. Mylopoulos, A. Antón, and S. Gritzalis, eds.), pp. 110–126, Springer International Publishing, 2018.

[27] P. Xun, P.-D. Zhu, Y.-F. Hu, P.-S. Cui, and Y. Zhang, "Command Disaggregation Attack and Mitigations in Industrial Internet of Things," *Sensors*, vol. 17, no. 10, 2017.

[28] M. Krotofil, A. A. Cárdenas, B. Manning, and J. Larsen, "CPS: Driving cyber-physical systems to unsafe operating conditions by timing DoS attacks on sensor signals," in *Proceedings of the 30th Annual Computer Security Application Conference*, ACSAC '14, pp. 146–155, ACM.

[29] M. Krotofil, A. Cárdenas, J. Larsen, and D. Gollmann, "Vulnerabilities of cyber-physical systems to stale data - Determining the optimal time to launch attacks," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 4, pp. 213–232, 2014.

[30] C. Schuett, J. Butts, and S. Dunlap, "An Evaluation of Modification Attacks on Programmable Logic Controllers," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 1, pp. 61–68, 2014.

[31] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware Modification Attacks on Programmable Logic Controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76–84, 2013.

[32] S. McLaughlin and P. McDaniel, "SABOT: Specification-based Payload Generation for Programmable Logic Controllers," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pp. 439–449, ACM, 2012.

[33] S. McLaughlin, "On Dynamic Malware Payloads Aimed at Programmable Logic Controllers," 6th USENIX Workshop on Hot Topics in Security, Aug 2009.

[34] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," in *Proceedings of Black Hat USA*, Aug 2011.

[35] D. Peck and D. Peterson, "Leveraging Ethernet Card Vulnerabilities in Field Devices," Digital Bond, Jan 2009.

[36] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-Facing PLCs - A New Back Orifice," in *Proceedings of Black Hat USA*, Aug 2015.

[37] A. Abbasi and M. Hashemi, "Ghost in the PLC: Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack," in *Proceedings of Black Hat EU*, Nov 2016.

[38] W. Li, L. Xie, D. Liu, and Z. Wang, "False Logic Attacks on SCADA Control System," in *2014 Asia-Pacific Services Computing Conference*, pp. 136–140, Dec 2014.

[39] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, and O. Mohammed, "Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit," Network and Distributed System Security Symposium, Feb 2017.

[40] Éireann Leverett and R. Wightman, "Vulnerability Inheritance in Programmable Logic Controllers," in *Proceedings of GreHack*, Nov 2013.

[41] National Institute of Standards and Technology, "National Vulnerability Database." Online. Available at: `https://nvd.nist.gov`, accessed: 2018-12-03.

[42] E. R. Alphonsus and M. O. Abdullah, "A review on the applications of programmable logic controllers (PLCs)," *Renewable and Sustainable Energy Reviews*, vol. 60, pp. 1185–1205, 2016.

[43] elprocus.com, "What is a PLC System - Different types of PLCs with applications." Online. Available at: `https://www.elprocus.com/programmable-logic-controllers-and-types-of-plcs/`, accessed: 2018-12-06.

[44] Businesswire.com, "Technavio Announces Top Five Vendors in the Global Micro PLC Market from 2016 to 2020." Online. Available at: `https://www.businesswire.com/news/home/20160502005493/en/Technavio-Announces-Top-Vendors-Global-Micro-PLC`, accessed: 2018-12-06.

[45] Offensive Security, "Exploit Database." Online. Available at: `https://www.exploit-db.com`, accessed: 2018-12-03.

[46] OWASP, "Testing for XML Injection (OTG-INPVAL-008)." Online. Available at: `https://www.owasp.org/index.php/Testing_for_XML_Injection_(OTG-INPVAL-008)`, accessed: 2018-12-29.

Appendix A

CVE extractor

```
import json
import os
import re
import textwrap

jsonCVE = "D:/Users/Fredrik/Desktop/RQ1/CVE_JSON_FILES"
outputFileInclude = "D:/Users/Fredrik/Desktop/RQ1/
    ↪ included_CVE.txt"

try:
    os.remove(outputFileInclude)
except OSError:
    pass

def writeCVEInfoFile(cve, file):
    file.write("{0}\n".format(cve["cve"]["CVE_data_meta"]["ID
        ↪ "]))
    for vendor in cve["cve"]["affects"]["vendor"]["
        ↪ vendor_data"]:
        file.write("Vendor: {0}\n".format(vendor["vendor_name
            ↪ "]))

    if("baseMetricV3" in cve["impact"]):
        cvssV3 = cve["impact"]["baseMetricV3"]["cvssV3"]
        file.write("CVSS V3: {0} base score, {1}\n".format(
            ↪ cvssV3["baseScore"], cvssV3["vectorString"]))
    if("baseMetricV2" in cve["impact"]):
        cvssV2 = cve["impact"]["baseMetricV2"]["cvssV2"]
        file.write("CVSS V2: {0} base score, {1}\n".format(
            ↪ cvssV2["baseScore"], cvssV2["vectorString"]))
    file.write("\n")
    file.write("{0}\n".format(textwrap.fill(cve["cve"]["
        ↪ description"]["description_data"][0]["value"])))
    file.write("\n\n\n")
```

```
startDate = "2010-01-01T00:00Z"
stopDate = "2018-10-31T23:59Z"

cveFiles = os.listdir(jsonCVE)

cveCounter = 0
includedIndex = []

includedVendors = ["siemens", "abb", "schneider", "schneider␣
    ↪ electric", "rockwell", "rockwell␣automation", "
    ↪ mitsubishi", "ge", "general␣electric",
                   "ge-fanuc", "omron", "koyo", "panasonic",
                     ↪  "idec", "keyence", "toshiba", "
                     ↪ fuji", "beckhoff", "bosch", "
                     ↪ rexroth", "bosch␣rexroth", "allen"
                  "allen␣bradley", "allen-bradley", "hitachi
                     ↪ ", "delta", "honeywell", "yokogawa",
                     ↪  "b&r␣industrial␣automation", "b&r",
                     ↪  "philips", "philips␣components",
                  "festo", "kim␣controls", "kim", "horner␣
                     ↪ electric", "horner", "see␣automation
                     ↪ ␣&␣engineers", "see"]

includedDescWords = ["plc", "programmable␣logic␣controller",
    ↪ "progammable-logic␣controller",
                 "simatic", "ac500", "modicon", "micrologix"
                     ↪ , "melsec", "smartguard", "
                     ↪ directlogic", "kostac", "microsmart",
                 "smartrelay", "opennet", "smartaxis", "kv",
                     ↪  "micrex", "twincat", "masterlogic",
                     ↪ "controledge"]

includedVendorsRegX = [re.compile(r"^{0}".format(
    ↪ includedVendor), flags=re.IGNORECASE) for
    ↪ includedVendor in includedVendors]
includedDescWordRegX = [re.compile(r"\b{0}\b".format(
    ↪ includedDescWord), flags=re.IGNORECASE) for
    ↪ includedDescWord in includedDescWords]
includedDescWordRegX.extend([re.compile(r"\b{0}\b".format(
    ↪ includedVendor), flags=re.IGNORECASE) for
    ↪ includedVendor in includedVendors])

for cveFile in cveFiles:
    with open(os.path.join(jsonCVE, cveFile), encoding="utf8"
        ↪ ) as file:
        data = file.read()
        parsedCVE = json.loads(data)
```

```
    index = 0
    for cveEntry in parsedCVE["CVE_Items"]:
        publishedDate = cveEntry["publishedDate"]
        if(publishedDate >= startDate and publishedDate <=
            ↪ stopDate):

            for vendor in cveEntry["cve"]["affects"]["vendor"
                ↪ ]["vendor_data"]:
                if(any(regex.search(vendor["vendor_name"]) is
                    ↪  not None for regex in
                    ↪ includedVendorsRegX)):
                    includedIndex.append(index)
                    break

            if(index not in includedIndex):
                cveDescription = cveEntry["cve"]["description
                    ↪ "]["description_data"][0]["value"]
                if(any(regex.search(cveDescription) is not
                    ↪ None for regex in includedDescWordRegX)
                    ↪ ):
                    includedIndex.append(index)

        index += 1

    with open(outputFileInclude, mode="a", encoding="utf8")
        ↪ as file:
        for cveIndex in includedIndex:
            currentCVE = parsedCVE["CVE_Items"][cveIndex]
            writeCVEInfoFile(cve=currentCVE, file=file)
            print("Included: {0}".format(currentCVE["cve"]["
                ↪ CVE_data_meta"]["ID"]))

    cveCounter += len(includedIndex)
    includedIndex.clear()

with open(outputFileInclude, mode="a", encoding="utf8") as
    ↪ file:
    file.write("Number of CVEs in this file: {0}".format(
        ↪ cveCounter))
```

Appendix B

Exploit extractor

B.1 Exploit-DB search

```
#!/bin/bash

declare -a arr=("siemens" "abb" "schneider" "scheider␣
    ↪ electric" "rockwell" "rockwell␣automation" \
"mitsubishi" "ge" "general␣electric" "ge-fanuc" "omron" "koyo
    ↪ " "panasonic" "idec" "keyence" "toshiba" "fuji" "
    ↪ beckhoff" "bosch" "rexroth" "bosch␣rexroth" \
"allen" "allen␣bradley" "allen-bradley" "hitachi" "delta" "
    ↪ honywell" "yokogawa" \
"b&r␣industrial␣automation" "b&r" "philips" "philips␣
    ↪ components" "festo" "kim␣controls" "kim" "horner␣
    ↪ electric" "horner" "see␣automation␣&␣engineers" "see" "
    ↪ plc" "programmable␣logic␣controller" "programmable-
    ↪ logic␣controller" "simatic" "ac500" "modicon" "
    ↪ micrologix" "melsec" "smartguard" "directlogic" "kostac
    ↪ " "microsmart" "smartrelay" \
"opennet" "smartaxis" "kv" "micrex" "twincat" "masterlogic" "
    ↪ controledge")

for i in "${arr[@]}"
do
        searchsploit -tj "$i" > $"./Json/$RANDOM"
done
```

B.2 Filtering and sorting

```
import json
import os
import re

jsonFiles = "/home/dev/Json"

files = os.listdir(jsonFiles)
```

```
wordList = ["siemens", "abb", "schneider", "scheider electric
    ↪ ", "rockwell" ,"rockwell automation",
"mitsubishi", "ge", "general electric", "ge-fanuc", "omron",
    ↪ "koyo", "panasonic", "idec", "keyence", "toshiba", "
    ↪ fuji", "beckhoff", "bosch", "rexroth", "bosch rexroth",
"allen", "allen bradley", "allen-bradley", "hitachi", "delta"
    ↪ , "honywell","yokogawa",
"b&r industrial automation","b&r","philips","philips
    ↪ components","festo","kim controls","kim","horner
    ↪ electric","horner","see automation & engineers","see","
    ↪ plc","programmable logic controller", "programmable-
    ↪ logic controller" ,"simatic","ac500","modicon","
    ↪ micrologix","melsec","smartguard","directlogic","kostac
    ↪ ","microsmart","smartrelay",
"opennet","smartaxis","kv","micrex","twincat","masterlogic","
    ↪ controledge"]

regexList = [re.compile(r"\b{0}\b".format(word), flags=re.
    ↪ IGNORECASE) for word in wordList]
matchingExploits = {}

def asInt(x):
                return int(x)

for file in files:
        with open(os.path.join(jsonFiles, file), encoding="
            ↪ utf8") as f:
                data = f.read()
                parsed = json.loads(data, strict=False)

        for entry in parsed["RESULTS_EXPLOIT"]:
                if(any(regex.search(entry["Title"]) is not
                    ↪ None for regex in regexList)):
                        if(entry["EDB-ID"] not in
                            ↪ matchingExploits):
                                matchingExploits[entry["EDB-
                                    ↪ ID"]] = entry["Title"]

for edbid in sorted(matchingExploits, key=asInt):
        print("{0}\n{1}\n\n".format(matchingExploits[edbid],
            ↪ edbid))
```

Appendix C

Experimentation test cases and result

Table C.1: The memory fields representing a valid "OK" and "Error" response.

Response	Memory field (Hex)
OK	3|0|4|30 30 30 31
Error	5|0|4|30 30 30 31

Table C.2: The results after the completion of the tests targeting the length value field in the PLC.

Memory field (Hex)	Result
3|0|4|30 30 30 31	Accepted (Expected result)
3|0|3|30 30 30 31	Denied
3|0|2|30 30 30 31	Denied
3|0|1|30 30 30 31	Denied
3|0|0|30 30 30 31	Seems to be accepted, no affect
3|0|FFFF|30 30 30 31	Denied
3|0|FF|30 30 30 31	Denied
3|0|5|30 30 30 31	Denied
5|0|4|30 30 30 31	Accepted (Expected result)
5|0|3|30 30 30 31	Denied
5|0|2|30 30 30 31	Denied
5|0|1|30 30 30 31	Denied
5|0|0|30 30 30 31	Denied
5|0|FFFF|30 30 30 31	Denied
5|0|FF|30 30 30 31	Denied
5|0|5|30 30 30 31	Denied

Table C.3: The results after the completion of the tests targeting the padding field in the PLC.

Memory field (Hex)	Result
3|0|4|30 30 30 31	Accepted (Expected result)
3|FF|4|30 30 30 31	Accepted, no negative affect
3|0|0|30 30 30 31	Seems to be accepted, no affect
3|FF|0|30 30 30 31	Seems to be accepted, no affect
5|0|4|30 30 30 31	Accepted (Expected result)
5|FF|4|30 30 30 31	Accepted, no negative affect

Table C.4: The results after completion of the tests targeting the data field in the PLC.

Memory field (Hex)	Result
3|0|4|30 30 30 31	Accepted (Expected result)
3|0|4|30 30 30 32	Accepted (Expected result)
3|0|4|00 30 30 31	Denied, not configured
3|0|4|30 30 30 33	Denied, not configured
3|0|4|30 30 30 34	Denied, not configured
5|0|4|30 30 30 31	Accepted (Expected result)
5|0|4|41 42 43 44	Accepted, no negative affect
5|0|4|41 42 43 0	Seems to be accepted, no affect
5|0|4|01 31 34 41	Seems to be accepted, no affect
5|0|4|0 0 0 0	Seems to be accepted, no affect
5|0|4|01 01 01 01	Seems to be accepted, no affect
5|0|4|02 02 02 02	Seems to be accepted, no affect
5|0|4|03 03 03 03	Seems to be accepted, no affect
5|0|4|04 04 04 04	Seems to be accepted, no affect
5|0|4|05 05 05 05	Seems to be accepted, no affect
5|0|4|06 06 06 06	Seems to be accepted, no affect
5|0|4|07 07 07 07	Seems to be accepted, no affect
5|0|4|08 08 08 08	Seems to be accepted, no affect
5|0|4|09 09 09 09	Seems to be accepted, no affect
5|0|4|0A 0A 0A 0A	Accepted, adds four new lines in MES log file
5|0|4|0B 0B 0B 0B	Seems to be accepted, no affect
5|0|4|0C 0C 0C 0C	Seems to be accepted, no affect
5|0|4|0D 0D 0D 0D	Seems to be accepted, no affect
5|0|4|0E 0E 0E 0E	Seems to be accepted, no affect
5|0|4|0F 0F 0F 0F	Seems to be accepted, no affect
5|0|4|10 10 10 10	Seems to be accepted, no affect
5|0|4|11 11 11 11	Seems to be accepted, no affect
5|0|4|12 12 12 12	Seems to be accepted, no affect
5|0|4|13 13 13 13	Seems to be accepted, no affect
5|0|4|14 14 14 14	Seems to be accepted, no affect
5|0|4|15 15 15 15	Seems to be accepted, no affect
5|0|4|16 16 16 16	Seems to be accepted, no affect
5|0|4|17 17 17 17	Seems to be accepted, no affect
5|0|4|18 18 18 18	Seems to be accepted, no affect
5|0|4|19 19 19 19	Seems to be accepted, no affect
5|0|4|1A 1A 1A 1A	Seems to be accepted, no affect
5|0|4|1B 1B 1B 1B	Seems to be accepted, no affect
5|0|4|1C 1C 1C 1C	Seems to be accepted, no affect
5|0|4|1D 1D 1D 1D	Seems to be accepted, no affect
5|0|4|1E 1E 1E 1E	Seems to be accepted, no affect
5|0|4|1F 1F 1F 1F	Seems to be accepted, no affect
5|0|4|7F 7F 7F 7F	Accepted, no negative affect
5|0|4|01 F6 3A 20	Seems to be accepted, no affect
5|0|4|3C 41 42 3E	Accepted, no negative affect (characters escaped)
5|0|4|"OWASP tests"	Accepted, no negative affect (characters escaped)

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden