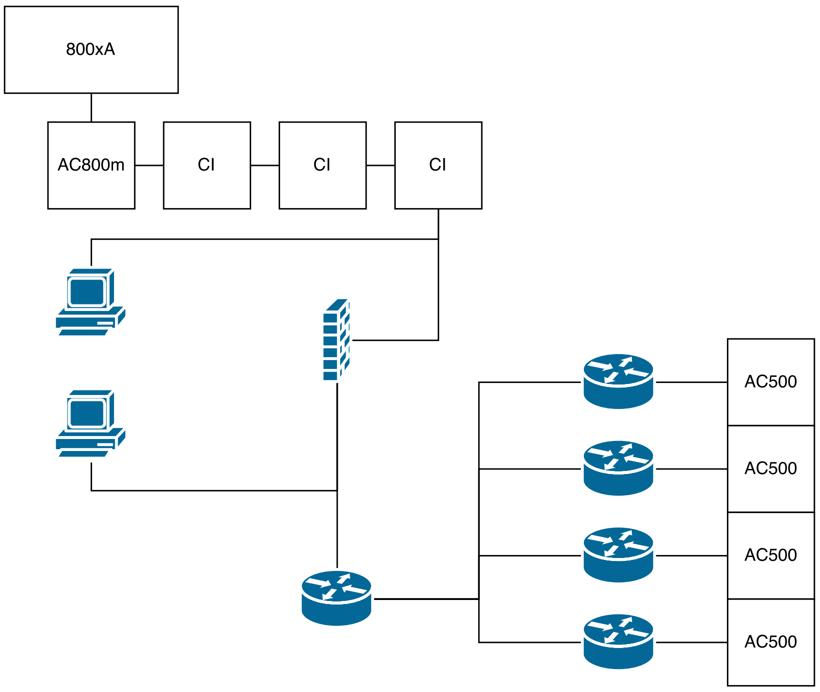
CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG



Network-based Intrusion Detection Systems for Industrial Control Systems

Detecting anomalies and semantic tampering in Industrial Control Systems

Master's thesis in Computer Systems and Networks (MPCSN)

JOHAN ANGSÉUS & RIKARD EKBOM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2017

Network-based Intrusion Detection Systems for Industrial Control Systems

Detecting anomalies and semantic tampering in Industrial Control Systems

JOHAN ANGSÉUS
RIKARD EKBOM

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2017

Network-based Intrusion Detect Systems for Industrial Control Systems
Detecting anomalies and semantic tampering in Industrial Control Systems
JOHAN ANGSÉUS, RIKARD EKBOM

Supervisor: Magnus Almgren, Networks and Systems Division, Department of Computer Science and Engineering, Chalmers University of Technology
Examiner: Erland Jonsson, Networks and Systems Division, Department of Computer Science and Engineering, Chalmers University of Technology

Master's Thesis 2017
Department of Computer Science and Engineering
Chalmers University of Technology University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover:
Network model that has been used to capture the network data analyzed throughout the thesis.

Gothenburg, Sweden 2017

Network-based Intrusion Detection Systems for Industrial Control Systems
Detecting anomalies and semantic tampering in Industrial Control Systems
JOHAN ANGSÉUS, RIKARD EKBOM
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

As Industrial Control Systems (ICSs) become more and more connected it follows that they need to become more secure. Traditional Intrusion Detection Systems (IDSs) do not work well due to the fact that they mostly work on a signature basis and there are not many known signatures to detect attacks on ICSs. Since the network traffic from an ICS is claimed to be static and signatures are scarce, searching for anomalies in the network to detect threats is more effective. This can be achieved using machine learning and other statistical models, teaching the system to tell regular traffic from irregularities. In this thesis we survey papers from related work and evaluate their results, conduct a risk analysis of ICSs from published sources and a workshop with an industry expert. Based on the survey and the risk analysis we analyze over 100 days of network traffic from a water distribution system in order to get further understanding of how ICSs act, both considering network traffic and process semantics. From this work we propose and evaluate three methods to be used when creating a more data driven IDS, capable of detecting process semantic tampering within an ICS.

Our results from conducted experiments exhibit a static nature of the data originating from the ICS and the result from evaluating two of the three proposed methods using proof of concept systems, we deem that these anomaly-based detection methods work well for both semantic tampering as well as on a network basis. Having an IDS using a fusion of all three proposed methods, would benefit the security of an ICS since both semantics and network behavior are taken into account.

Keywords: intrusion detection system; industrial control system; network-based; anomaly detection; data-driven

Acknowledgements

We would like to thank persons who have provided valuable guidance and help throughout this thesis. We would like to specially thank **Rikard Bodforss** for providing in-depth knowledge about security in industrial control systems. We would also like to thank our supervisor, **Magnus Almgren**, for providing helpful feedback, and for guiding us through this thesis.

Terminology

CI	Communication interface. Interface used for communication to an arbitrary medium.
CIA	Confidentiality, Integrity and Availability.
DCS	Distributed Control Systems. A decentralized system with each node having control over its own domain.
HMI	Human Machine Interface. User interface for the control over an industrial system.
ICS	Industrial Control System. A system that controls an industrial process.
IDS	Intrusion Detection System. A system that detects and alarms on intrusions.
IDPS	Intrusion Detection and Prevention System. A system that detects and prevents intrusions.
IP	Internet Protocol. A protocol used for addressing on the Internet.
NIDS	Network Intrusion Detection System. An IDS that operates on network traffic, often packets.
NIST	The National Institute of Standards.
PLC	Programmable Logic Controller. A controller often used for automation in industrial systems.
RMA	Reliability, Maintainability and Availability.
RTU	Remote Terminal Unit. Micro-processor that interfaces with e.g., sensors in a DCS.

RTT
: Response trip time. Measurement of how long time it takes to send a packet to a node and receive an answer.

SCADA
: Supervisory Control And Data Acquisition. A system for regulating and controlling processes.

SOM
: Self-Organizing-Maps.

SVDD
: Support Vector Data Description.

SVM
: Support Vector Machines.

TCP
: Transmission Control Protocol. A transport protocol commonly used on the Internet.

UDP
: User Datagram Protocol. A transport protocol commonly used on the Internet.

VLAN
: Virtual Local Area Network.

VPN
: Virtual Private Network.

WAN
: Wide-Area-Network.

Contents

List of Figures

List of Tables

List of code and logs

1 Introduction

Computer security has moved beyond normal office systems, and attackers are now also targeting Industrial Control Systems (ICSs) [6]. One of the most noteworthy attacks against an ICS in recent history is the Stuxnet attack, which is called the first cyber-warfare weapon. Unlike a common espionage attack, the main objective of Stuxnet was likely to physically sabotage the Iranian nuclear program [7]. However, this is not the only attack that has occurred. In 2007 the U.S. government showed how hackers could potentially destroy a generator in a power plant with just 21 lines of code [8]. The approach of the attack was to continuously connect and disconnect a generator to the power grid, out of phase, which caused the generator to fail [9]. Also, in late 2015 and early 2016 two power distribution companies in Ukraine stated that their systems were hacked and hijacked in order to shut down electricity to more than 80.000 people in the country. The hackers also tampered with the engineer workstations, thus making it harder for the companies to restore power to their customers. Ukrainian officials have blamed Russia for the attack, but Russia has not acknowledged anything [8]. A similar event occurred in Germany 2015 where a steel mill was hacked in order to tamper with a blast furnace, that eventually resulted in massive damage on the plant [10].

A recent security study of U.S. based energy companies reported that 76% of the 291 involved utility and energy companies suffered from one or more security incidents in 2010 [6]. Due to these events and due to the fact that ICSs often control infrastructural systems e.g. water plants and power grids, which are essential for society, the attention to security in these systems has increased as of late.

Based on the need to protect ICSs, the focus of this thesis is to better understand risks, towards such systems, and how security monitoring mechanisms should be constructed and built by investigating previous methods in literature. One of the problems in the domain, though, is to get data from real systems in order to perform validation of the results. Many of the research prototypes are only theoretical, and tested with a limited simulation model, or in the best case, having an experimental validation in a real system but only for a limited time. Such a short validation may not allow a proper verification of the underlying assumptions used for the security monitoring system, nor demonstrate its effectiveness. For that reason, we will also use a larger dataset collected from a real ICS (water distribution system) to validate previous claims in literature, as well as trying to further understand how threats can be seen in the dataset.

Our methodology is data-driven and is focused on a large collected dataset from a water distribution system, which is further described in Chapter 5 (Experiments and system model).

Our thesis is organized as follows. To better understand the context of ICSs and what kind of contributions that have been done previously, we study previous methods in Chapter 3, that then form the basis for our more experimental work. More precisely, we survey the scientific literature to better understand what has been done previously, which methods were used, the validation methodology, which can be validated in simulated or real traffic. In this survey we explicitly focus on assumptions or claims in the papers, such as, the traffic in ICSs is very regular *and* thus suited for anomaly detection, how is regularity defined? How many events seem to be time-triggered, event-triggered versus user-triggered in a real system? Such implicit assumptions may play a role in the choice of methods for security monitoring and should be carefully analyzed.

After the survey has been performed, we perform a risk analysis in Chapter 4 in order to get a deeper insight in what kind of risks that exist in an ICS. The risk analysis consists of an extensive analysis of standards published by the National Institute of Standards (NIST), as well as a workshop with local expert with insight in ICS security. This is important in order to understand what to look for in the datasets, as well as a motivation for developing more secure ICSs.

Based on the survey results, we choose a few assumptions and methods and investigate them from an experimental perspective in Chapter 5 and Chapter 6. Will the methods also work on our data? Are the results comparable? Can we draw any conclusions from their and our result? These are some of the questions that we answer.

The basis for Chapter 7 is a large dataset of captured network traffic from a water distribution system, which we use in order to evaluate three proposed methods, which are based on results from the survey, risk analysis and experiments. The methods consist of one method that looks for anomalies in process semantics, one method that perform analysis on traffic flows by means of network parameters and one that use neural networks to watch for anomalies by means of network traffic. The methods are evaluated in order to indicate whether or not they are suitable in an Intrusion Detection System (IDS) for ICSs.

1.1 Purpose

The goal of this master thesis is to analyze traffic from a real world ICS, and to better understand what security mechanisms may work for ICSs, and then propose methods which should be more data driven than an IDS with only predefined rules to match against. The new proposed methods could be deployed in systems currently used, in order to protect the infrastructure from malicious traffic in order to protect ICSs further.

1.2 Scope

We perform literature studies from articles related to our thesis in order to gain knowledge about how IDSs and ICSs work together. We also analyze real life network traffic from an ICS to gain insight of how such a network acts and communicates. From the knowledge gained during the analysis and literature study about IDSs we propose three methods that can be used when implementing an IDS for ICSs. Some of these methods will be from the field of machine learning since our approach will be data driven. However, we do not propose any new machine learning methods that can be used in an IDS. We only use already proposed methods, such as different regression methods and anomaly detection methods. Proposing a new machine learning algorithm would take a lot of effort and time, and is not considered to be within the scope of this thesis due to the limited time frame.

1.3 Road map

Chapter 1 - Introduction Contains an introduction to this thesis, the purpose of the thesis, the scope of the thesis and this road map that is supposed to help the reader understanding how the report is structured.
Chapter 2 - Background Describes the various areas that needs to be understood in order to comprehend the thesis in its entirety.
Chapter 3 - Related work and survey Describes what other authors have done in the related area as well as a comparison of our work and theirs. This chapter also includes a survey that goes through relevant articles that have been selected depending on relevance, results, conferences and citations. This is done in order to get a clear picture of how the field of ICSs looks like at the moment.
Chapter 4 - Risk analysis An analysis of the different risks that may threaten an ICS, as well as different threat actors. This chapter consists of a general description of various factors to consider when building an ICS, various threats and threat actors that exist. The chapter also includes a workshop with an industry expert and what they think is important when maintaining an ICS.

Chapter 5 - Experiments and system model This chapter consists of the experiments that we perform on our data, based on the previous findings in the survey from Chapter 3, and with regard to the findings from the Risk analysis in Chapter 4. In this chapter we state how we are going to probe and analyze our dataset, which is explained in this chapter as well, in order to check if the findings from the related papers are relevant to us, and if we can detect any of the threats that are explained in the Risk analysis.
Chapter 6 - Results from experiments The Results chapter contain the results yielded from the experiments explained in Chapter 5.
Chapter 7 - Proposed methods Contains our three proposed methods and evaluation when designing an IDS for ICSs.
Chapter 8 - Discussion A discussion about our findings from the thesis work.
Chapter 9 - Ethics and sustainable development Presents a discussion about the ethical aspect of the work and its sustainability.
Chapter 10 - Future work Ideas and thoughts about things that can be done further in the area.
Chapter 11 - Conclusion Concludes the result and the discussion of the thesis.

2 Background

This chapter introduces Supervisory Control and Data Acquisition (SCADA), Programmable Logic Controllers (PLCs), the Modbus protocol and different systems, such as ICSs and IDSs to the reader. The purpose of this is to gain a general knowledge about terms and systems that will be used throughout the report, so the reader can comprehend the rest of the report. Two established IDS programs, Snort and Bro, are also presented since they are used in order to analyze the captured data, as well as three popular network analysis tools, Wireshark, tshark and tcptrace.

2.1 Industrial Control Systems

ICSs is a branch of systems that often are present in critical infrastructure and industrial environments. The term ICS covers general systems that utilize PLCs, which is later explained in Section 2.1.2, SCADA systems and Distributed Control Systems (DCS). This thesis mainly focuses on ICSs in consideration to SCADA and PLC systems, not DCS. However, the systems, SCADA and DCS, are very similar and many of the techniques and much of the technology is common for both of them. A commonly used communication protocol for these systems is the Modbus protocol, later explained in Section 2.1.3.

2.1.1 Supervisory Control and Data Acquisition

SCADA is a system that is built in order to handle numerous inputs and outputs [1]. The idea is to collect information to a central place where it can be processed and many times presented to a human via a Human-Machine-Interface (HMI). The data collection is done in real time and makes it easier to control systems that exists on multiple locations. Thus, it is only needed to have one central point of operation. It also allows for automatic runtime if programmed, but also provides support for control commands by an operator, which is often an engineer.

The structure of a SCADA system is very hierarchical. The most important part in the system is the control server, which collects information from other nodes in the system, provides support for automatic runtime and provides an interface for operators. Between the control server and other parts of the system there is communication equipment of various forms, e.g. fiber, radio-links or telephone lines, which is sometimes also called Wide-Area-Network (WAN). The other nodes in the network are called field sites, which often are sites that are located geographically apart. The field sites often consist of a remote terminal unit (RTU) and one or more PLCs. Figure 2.1 shows how the SCADA structure looks like, and this is typically how a SCADA system is modelled in means of communication between different nodes.

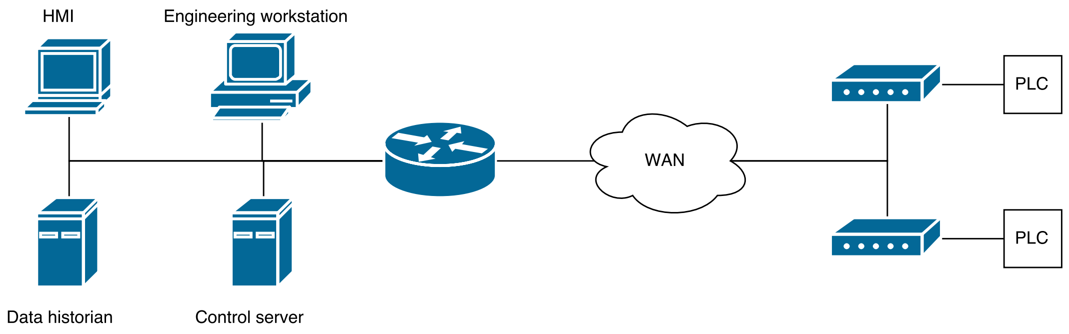


Figure 2.1: A typical structure of a SCADA system [1].

Figure 2.1 only shows one way of building a SCADA system. The network can also be built in series where you connect multiple field-sites to each other, thus removing the need for each field-site to have its own connection to the control server. You can also arrange the system as a star, where one field-site connects multiple sub-stations. Another approach is the multidrop hierarchy, where each modem in the field-sites are daisy chained together.

When building ICS networks it is recommended to separate corporate networks from control networks. There are several reasons for this. Traffic on corporate networks is generally different from traffic on control networks. Corporate networks often consist of many services, such as email, FTP, HTTP and other protocols. Control networks on the other hand, should only consist of one protocol that is chosen for communication between nodes in the network. Firewalls can be used in order to separate traffic between different networks, only allowing valid traffic to pass through between the networks. Figure 2.2 shows how a typical separation between the corporate network and the control network could look like. This is just one of the various approaches that exist, but is one of the simpler approaches.

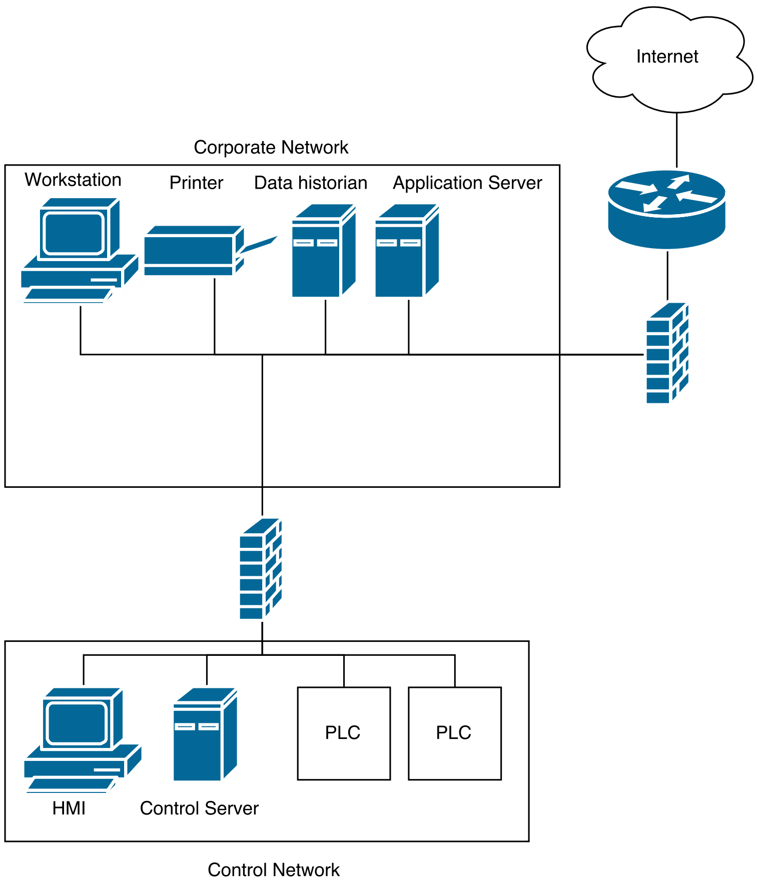


Figure 2.2: Typical separation between corporate networks and control networks [1].

Segmentation can also be applied between different control networks, based on the authority of the personnel using the nodes in the control network, different policies, level of trust, functional criticality but also the amount of traffic. The usage of segments on the network creates different security domains, which is considered more secure than having all nodes in the same domain. There are multiple ways of separating ICS networks. One way of separating the networks is by using Virtual Local Area Networks (VLAN), which logically separates networks on Layer 2 of the OSI-model, but they can still be on the same physical network. Virtual Private Networks (VPN) are also popular, which creates virtual networks in Layer 3 of the OSI-model. If security is considered to be very important, physical separation is also possible, where you physically separate the networks, which means that you have separate network peripherals and separate network media. You can also use firewalls here to only allow traffic between certain nodes in the network.

2.1.2 Programmable Logic Controller

PLCs are widely used in the area of automation, like in industrial processes, such as ICSs. The PLC itself consists of a processor with programmable memory, various input and output channels, and communication interfaces, which is shown in Figure 2.3. PLCs often receive various forms of input, e.g. sensor data, control commands and other perceived data. From the input data they perform some kind of action, which is the output from the PLC, e.g. turning on or off a relay, sending some data or using actuators. The communication interfaces allow the PLC to be connected to networks, which sometimes can be seen as a security risk [11].

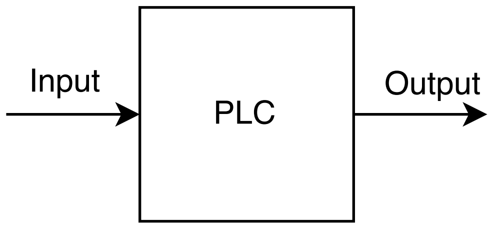


Figure 2.3: Logical chart of a simple PLC showing Input/Output (I/O).

The operation of a PLC can be described in the following way [12]:

Step 1: Input scan. Scan and receive values from various input devices that are connected to the PLC.

Step 2: Program scan. Run specific programs that are provided by the user of the PLC.

Step 3: Output scan. Either energize or de-energize the outputs of the PLC depending on the outcome of Step 2.

Step 4: Housekeeping. Perform various operations, such as communication with terminals and perform internal diagnostics.

2.1.3 Modbus protocol

Modbus is a messaging protocol that is placed in the seventh layer of the OSI model [13], the Application Layer. It has been the de facto industrial standard since 1979 for client and server communication between automation devices and controllers found in ICSs, such as the PLC. Modbus utilizes a client-server communication model. This is also sometimes called a master-slave technique in which the master is a device that controls other devices and queries them for either information or issues commands to them. The slaves listen to incoming queries from the master, and perform actions accordingly to the query. Figure 2.4 shows how a typical message session between a master device and a slave device looks like [2].

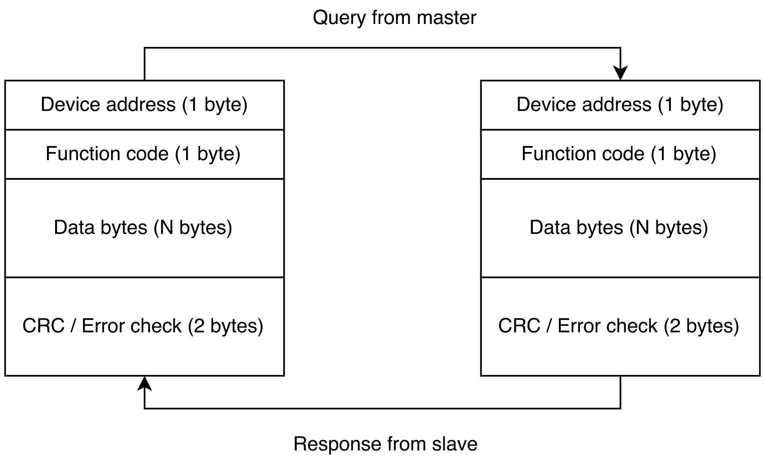


Figure 2.4: Modbus protocol frame [2].

2.1.4 Water treatment plants and distribution

A water distribution network acts as a connection between consumers of water and sources of water. The network consists of different hydraulic pieces, such as pipes, reservoirs, valves and pumps. One of the major tasks is to optimize water flow to the consumer in order to make sure that all consumers receive the amount of water they demand [14].

Water treatment plants and water distribution systems often include the use of ICSs. Many critical parts of the operation is performed by either SCADA systems or DCSs and both of them mostly utilize PLCs as the controlling components for the systems [1].

2.2 Intrusion Detection Systems

An IDS is a program that monitors computer systems for signs of unauthorized access, attempts to access restricted computer systems or other undesirable and illegal behavior [3, 15, 16]. If the IDS detects any strange behavior it will alert the system administrator to bring further attention. IDSs can be divided into several categories depending on the technologies that they employ. Scarfone and Mell [3] divide them into four categories in their publication from the National Institute of Standards, Guide to Intrusion Detection and Prevention Systems (IDPS); network-based, wireless, network behavior analysis and host-based. The most popular ones are network-based IDSs and host-based IDSs. A host-based system listens and analyzes the internal of the computer system while a network-based listens to network traffic and performs analysis on the captured packets. In order to detect possible intrusions these systems often base their detection on certain criteria and methodologies. Examples of those types are; anomaly-based that searches for behaviur that deviates from a predefined set of rules, and signature-based that listens to traffic and alerts if packets matches any of its predefined malicious signatures or patterns. Figure 2.5 shows a possible setup for a network using a Network Intrusion Detection System (NIDS).

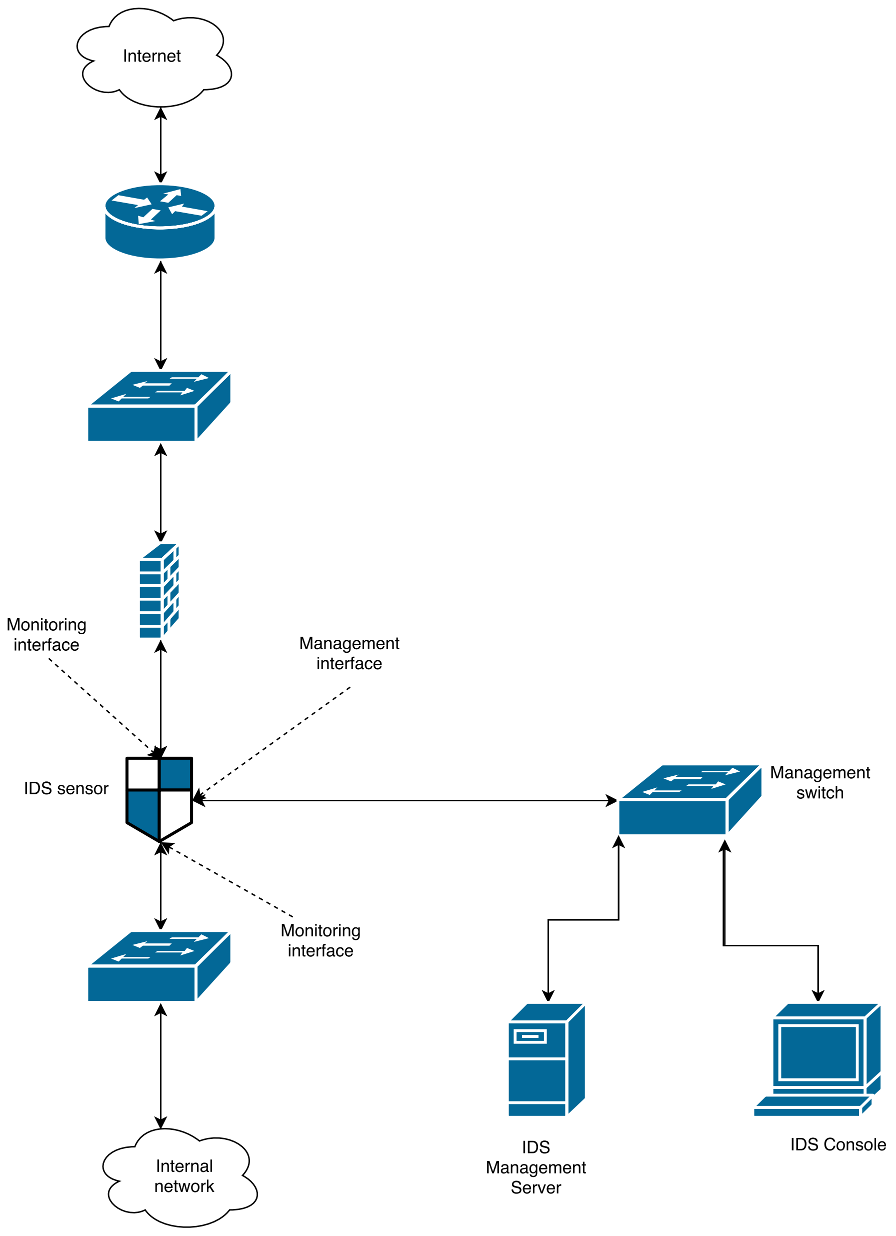


Figure 2.5: Example of a NIDS architecture [3].

2.2.1 Bro

Bro [17], developed by Vern Paxson, has been used for over 15 years. It is rooted deeply in research, and is used in many scientific environments to protect them from being attacked. Bro is a versatile network analysis framework that provides network intrusion detection functionality. It is efficient, stateful and provides in depth analysis for many known protocols which sets a good ground for an IDS. It also provides ways to reconfigure its functionality by providing open interfaces to other applications, as well as a domain-specific scripting language that allows for specifying policies. It also features support to detect and analyze traffic generated from the Modbus protocol based on the work done by Hadžiosmanović el al. [4].

2.2.2 Snort

Martin Roesch developed Snort [18] to be a lightweight network IDS which would monitor networks in order to detect suspicious network traffic and attacks. Snort detects suspicious network traffic and attacks by employing pattern matching rules, and when a match is found the IDS triggers an alert. By doing this, it would supply administrators with enough data to act accordingly to prevent any further suspicious activity on the network. Snort is the most widely deployed IDS in the world [19]. In the update to version 2.9.2 of Snort, a module to decode the Modbus protocol was added which would allow users to write rules for Modbus packets. Figure 2.1 shows an example of Snort output of an alert entry.

Listing 2.1: Snort alert output example.

```
[**] [1:382:7] ICMP PING Windows [**]
[Classification: Misc activity] [Priority: 3]
12/09-12:42:13.427034 10.34.211.71 -> 192.168.211.82
ICMP TTL:128 TOS:0x0 ID:9184 IpLen:20 DgmLen:60
Type:8  Code:0  ID:512   Seq:2560  ECHO
[Xref => http://www.whitehats.com/info/IDS169]
```

2.2.3 Employing an IDS in ICSs

Snort and Bro are examples of two open-source NIDSs that are used to analyze network traffic. However, employing ordinary security measures, such as Snort and Bro in ICSs have been investigated by several authors. Hadžiosmanović et al. [6] investigate the difficulties that arise when trying to secure ICSs, and the fact that traditional IDSs cannot easily be used to monitor ICSs. A common way to secure systems is to use a detection system that analyzes data and then alert on given signatures. This approach is not very effective in ICSs due to the small amount of known signatures.

Hadžiosmanović et al. [4] also explain the difficulties in using normal IDSs for ICSs. This is because they do not take the actual process semantics into account, such as behavior of different nodes in the ICS and how they relate to each other. They propose a method that watch variables and notice when they go out of bounds and exhibit unexpected behavior, both considering possible minimal and maximal values of the variable in question and by using an autoregressive method to see if it experiences odd behavior.

As stated earlier in this section, an IDS can be anomaly-based, having it search for irregularities apart from the ordinary behavior. Machine learning can be used in order to implement such an anomaly-based IDS i.e., learning what traffic exhibits good behavior and what traffic is suspicious. Even though the techniques seem promising, Sommer et al. [20] discuss potential problems with implementing such systems, and possible guidelines that could be followed in order to ease development, which is further explained in Section 3. We will consider these guidelines in our experiment.

2.3 Network analysis tools

There are several network analysis tools that can be used in order to perform analysis on captured data from a network. This section further describes three tools, namely Wireshark, tshark and tcptrace.

2.3.1 Wireshark

Wireshark is a network protocol analyzer tool [21]. It provides a graphical interface for both live packet capture and analysis, as well as for offline packet capture analysis. It can parse and interpret many protocols from Layer 2 up to Application Layers. Wireshark is therefore very usable and versatile for many kinds of packet capturing and network analysis tasks.

2.3.2 tshark

tshark is a network protocol analyzer tool [22]. It supports capture of packets from a live network for analysis, as well as for analysis of offline packet capture files. It is very similar to Wireshark, but tshark is a command line based version of Wireshark. tshark goes through packet capture files and analyses their content according to certain filters. It can either filter traffic and output the actual packets that match the filter presets, or summarize packets according to certain display presets, such as conversations endpoints, e.g. Internet Protocol (IP), Transport Control Protocol (TCP) or User Datagram Protocol (UDP).

tshark can be used as a network analyzer, in order to gather information about the network in network metrics, such as endpoints that communicates with each other, data flows, type of traffic and various other parameters. These parameters can be used in order to better understand the network, as well as providing useful parameters that can be used in certain machine learning algorithms.

2.3.3 tcptrace

tcptrace is a network protocol analyzer tool that analyzes tcpdump files [23]. It can display extensive information about each connection seen, such as number of packets, bytes transferred and window sizes.

tcptrace is used in this project as a simple network analyzer in order to gain some further understanding of how the connections look like. One of the perks of tcptrace is that it can generate a combined output for multiple packet capture files, which is helpful in cases when you have several capture files that stretch over many days.

2.4 Data driven methods

Data driven methods are methods that use data from previous states of a system in order to predict a future state. This can be done in a statistical manner, or by using various machine learning approaches. The concept of data driven methods can be used in anomaly-detection, since the system can learn to differentiate between normal and anomalous traffic, from previous states of the system.

According to García-Teodoro et al. [24], anomaly-based approaches can use various techniques, for example Statistical based, Knowledge based or Machine learning based. They divide anomaly detection using data driven methods into three stages, namely parameterization, training stage and detection stage. Three specific methods and algorithms in the field of data driven methods are further explained in Chapter 7.

3

Related work and survey

This chapter presents related work performed by other authors which have studied similar fields as the topic of this thesis. It also contains the performed survey from a number of papers. These papers are chosen because they address properties such as IDSs, ICSs or data driven methods. The method and experiment from each paper, such as what kind of properties they are searching for and algorithms used, is presented in Table 3.3. The result of this survey serves as a source of inspiration for the proposed methods in Chapter 7.

3.1 Related work

The usage of IDSS in ICSs has been evaluated in previous work. Hadžiosmanović et al. [6] speak about the challenges that exist when trying to secure ICSs overall, but especially with the usage of ordinary IDSs. They explain that it is due to most IDSs are based on signatures of known attacks and vulnerabilities, but there are only a handful of these for ICSs. Hadžiosmanović et al. continue to speak about the difficulties that come with securing an ICS in [4]. Here they focus more on solving the issue by watching network packets and make sure that the data sent is following the semantics of the process that is being observed. They propose some data driven methods that watch the values of different registers in the system, and make sure that they do not go out of bounds.

Using data driven methods such as statistical approaches, and forms of machine learning, is a popular design choice when developing an IDS for ICSs. Sommer and Paxson [20] discuss the topic of using machine learning for IDSs and give guidelines that can be followed in order to ease development of such tools and frameworks, which have been taken in consideration throughout this thesis. Also, several authors, e.g. Valdes et al. [25] and Hui Lin et al. [26] discuss how process semantics for the actual ICS can be taken into consideration for electrical circuits and power grids. Valdes et al. talk about taking advantage of anomaly detection via unsupervised machine learning. Hui Lin et al. discuss similar topics and propose a method to identify sequences of control commands as either acceptable or faulty. Despite the fact that these studies have been performed on electrical circuits and power grids, some of the results are usable because of the similar nature of power grids and water treatment and distribution plants.

There are also authors that focus on pure network parameters, for example Valdes and Cheung [27]. They discuss how one can employ pattern-based anomaly detection and flow-based anomaly detection in ICSs and present various methods that can be used in order to detect anomalies.

By contrast to other authors, we do not only focus on one aspect. Instead, we choose to perform analysis on both process semantics and pure network statistics and behavior. Regarding process semantics, we utilize some of the thoughts brought up by Hadžiosmanović et al. [6, 4], but in contrast to them we have a far larger dataset to run experiments on, which in hand yield different results compared to theirs. Regarding network statistics and behavior, we utilize some of the methods proposed by Vales and Cheung [27], but perform our analysis on a large dataset from a real world ICS. Valdes and Cheung perform some tests on their approach, for example by running an nmap scan, which it indeed detects. However, we deem that this could provide inaccurate results compared to a real world incident or attack.

In an article published by the National Institute of Standards, Stouffer et al. [1] propose guidelines of how one should handle a SCADA system and ICSs overall. Their guidelines bring various issues to light, e.g. how one can configure their firewalls in order to prevent intrusion, when one should be worried about that an intrusion occurred and how SCADA systems and DCSs work in general. We conduct an extensive risk analysis by collecting important information from this article, and also hold a workshop with an industry expert. This is done in order to get further understanding of actual threats and risks, and how to look for them.

Our work focus on proposing methods that can be used when implementing an anomaly based IDS for securing ICSs, by taking inspiration from other authors, for example Hadžiosmanović et al., and Valdes and Cheung. Our proposed methods are based on and motivated by the results from a survey, a risk analysis and experiments that we run on our data.

3.2 Survey of papers

This section includes a survey of selected articles and papers related to the field of IDSs and ICSs. We start by motivating how and why we have selected the articles for our survey. What follows is a section where each article in the survey is summarized with their objective, findings and result. Following that we made a list of criteria in which we could more easily categorize the different articles based on properties such as if they had any assumptions beforehand, their data or if they used any particular algorithm. Finally, we summarize the survey results and compare the articles to each other and draw our own conclusions about patterns we have observed from the surveyed articles.

3.2.1 Methodology

As stated in Chapter 1, we present a survey of previous papers that have been published in the area of ICSs and IDSs. We chose a number of papers based on two factors; the quality of them by means of conferences, citations, as well as relevance to our work.

The survey started by looking at previous work performed by Hadžiosmanović and Sommer, which are performing work in similar areas. From this, a search on Google Scholar, ACM Digital Library and IEEE Xplore Digital Library is performed in order to find additional articles. During the search, articles with conferences and citations are prioritized, but articles without many citations are not necessarily biased because the area of security in ICSs using IDSs is fairly new. Generally, articles with citations are picked, but this is also influenced by the articles that cite them. If an adequate article is found, its related articles are also investigated. The relevance of the articles were measured by comparing the articles to our subject. Therefore, articles that contained data-driven approaches are prioritized, as well as articles that were using network traffic and process semantics. Based on the findings of the search, a first selection of 27 articles is made.

After this, we read the abstract and the conclusion of these articles independently, and ranked the articles based on their fit to our work. If an article got two scores of being well suited, we included it for the full analysis using all the context from that article. After the second selection we chose 8 papers to include in the full analysis.

In the full analysis we include two different kinds of surveys, namely one quantitative survey and one qualitative survey. In the quantitative survey a summary of all the articles is performed independently in order to provide a first glance of their performed work. A short summary of all the articles, seen in Table 3.4, is also performed in order to give an easier overview of the selected articles. In the qualitative survey we look for some specific parameters, such as data environment, method, algorithm, level of data, capture position, test validation, data extraction and various other assumptions, as seen in Table 3.3. The full description of the criteria used for the survey is seen in Table 3.2. A summary of the qualitative survey is also included in Section 3.2.4.

3.2.2 Quantitative survey

The following sections offer a summarized text as to what each paper chosen for the full analysis contained and highlight their objectives, approaches and results. The purpose of this section is to give further understanding of what is happening in the field of ICSs and IDSs, before continuing to the more qualitative survey in Section 3.2.3 The findings from the quantitative survey is later summarized in Section 3.2.4 in Table 3.4.

3.2.2.1 Communication Pattern Anomaly Detection in Process Control Systems

Valdes and Cheung [27] propose a learning-based approach that does not need attack-specific knowledge in order to work. Their approach learns from network communication patterns in ICSs, and especially focuses on basic network flow information. They specify that they store various network information in a database in their IDS, which keeps track of connection endpoints, flow of network traffic and a set of hosts which communicate with each other. When observing patterns, the system can either match a previously learned pattern or not, which could mean that the pattern is a new flow. The authors say that their approach can detect network probing attacks, denial of service attacks or flooding attacks, propagation of malicious software and the introduction of new devices in the network, such as master or slave devices. They motivate that this approach is relevant because ICSs often experience fairly regular communication patterns, especially between a master device and a slave device, and that the address space is often static and fairly limited.

As far as we can see, the authors provide all the relevant parameters and mathematical and statistical functions that they have been using. They also provide information about their test environment. However, they do not or describe provide the actual data that they have been using when performing the tests.

3.2.2.2 A Module for Anomaly Detection in ICS Networks

Mantete et al. [5] propose an approach for using Self-Organizing-Maps (SOM) in restricted IP networks, which exhibit some form of determinism, which ICS networks often and mostly do. They state that their approach is a complement to the already implemented security features, such as firewalls and other measures, in order to further strengthen the security of an ICS network. Their implementation is built on top of Bro's scripting engine (see Section 2.2.1 for additional information about Bro). Their SOM implementation takes 9 different parameters in consideration, as seen in Table 3.1.

The number of live TCP connections at the moment of connection termination
The number of live UDP connections at the moment of connection termination
The number of live ICMP connections at the moment of connection termination
Duration of connection that terminated
Overall network fragments pending reassembly by Bro
The amount of data (bytes) sent by connection responder
The amount of data (bytes) sent by connection originator
Number of packets sent by the connection responder
Number of packets sent by the connection originator

Table 3.1: Parameters utilized in the SOM extension [5]

The authors include information of where their testing data comes from, but do not provide any specifics about the actual data that their implementation runs on. They also provide figures and information about their system model and how the capturing and analysis are done. The provided result does not provide many metrics, but provide some reasoning about the outcome. The authors also provide information about how long it took to run the learning mode, which can be relevant as evaluation criteria.

3.2.2.3 Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes

Hadžiosmanović et al. [4] present a detector that continuously watch the network traffic in ICSs and extract the variable values for different registers in the process. With these variables they create prediction-based models that can predict what the value is supposed to be. The IDSs can therefore read variables from the network traffic and compare the predicted value against the actual value and see if there are any irregularities. Using this approach they can see if there are any direct or indirect attacks that aim to manipulate the process control.

The data used for their verification is gathered from two water treatment plants that serve about 1 million users in two urban areas. However, during their test phase, they have used a test-bed which is also described. In order to extract the data from the network traces they have used Bro with a Modbus analyzer, which is included nowadays. The tool that they implemented in order to calculate the predictions is available on Github [28].

Hadžiosmanović et al. [4] do not explain their results much, but provide some motivation in their conclusion. They claim that their approach and implementation works well for static variables as well as non-constant variables. However, they state that there are many corner cases in real world environments that might be hard to cover. Therefore, they recommend to watch a few, high value variables that are critical for the process and provide a complementary to the already existing safety monitoring systems.

3.2.2.4 Challenges and opportunities in securing industrial control systems

Hadžiosmanović et al. [6] discuss how one can secure ICSs in general, rather than how different algorithms can be used in order to do it. In their article they present material about ICSs in general, as well as risks and different aspects that should be covered when securing an ICS. Their paper conclude several experiences from research efforts for developing tools that further can increase security in ICSs. They perform their analysis in two steps. First they analyze host-based and network-based data resources. In the second step they try to identify relevant criteria that can be used during an evaluation. After that they highlight challenges that relate for each criteria and for each different aspect they have covered.

Hadžiosmanović et al. [6] provide some information about where their data comes from, as well as various conclusions from the results that they have gathered.

3.2.2.5 Anomaly-based network intrusion detection: Techniques, systems and challenges

Teodoro et al. [24] review many anomaly-based intrusion detection techniques as well as different existing platforms and frameworks that can be used in the area. Their article brings fourth explanations of statistical-based techniques, such as multivariate models and time series. They also explain knowledge-based techniques such as finite state machines, description languages and expert systems. Finally, they explain machine learning-based techniques, such as Markov models, neural networks and fuzzy logic. They also provide a list of several anomaly-based network IDSs research projects and IDS platforms.

Their result is a compilation of the previously described techniques that can be used in order to evaluate different anomaly-based techniques and systems.

3.2.2.6 Analysis of Intrusion Detection in Control System Communication Based on Outlier Detection with One-Class Classifiers

Onoda and Kiuchi [29] introduce Support Vector Machines (SVM) and Support Vector Data Description (SVDD) when performing outlier detection. They have practiced the two methods in an experimental control system network and the outcome of their work is a comparison of the two methods and what they detected respectively.

They provide extensive information about how their network model looks like, what packets they have captured and what kind of data they have used when training their outlier detection algorithms. Also, they provide results for the two different methods in matter of how long time it took for the algorithms to detect outliers.

3.2.2.7 Network Traffic Features for Anomaly Detection in Specific Industrial Control System Network

Mantere et al. [30] discuss how the machine learning approach can be used when constructing IDSs for ICSs, in order to decrease the amount of manual configuration that needs to be done before the IDS is fully operational.

In their report they show how data has been collected and other useful statistics about the data. They compare their own captured data against different statements in order to analyze what kind of parameters that can be useful for watching when using machine learning. Their conclusions and results are covered for several parameters.

3.2.2.8 Neural Network Based Intrusion Detection System for Critical Infrastructures

Linja et al. [31] use a combination of two neural network algorithms in order to detect deviations from normal behavior, mostly by looking at network traffic. They use a testbed where they have a hub connecting a PLC and a control PC. In order to simulate attacks they use Nmap, Metasploit and Nessus, which are network and forensics tools.

Their analysis is based on windows of network traffic, from where they extract certain information. The most important attributes that they extract from each packet are the following: number of IP-addresses, number of packets with window size of 0 bytes, average interval between packets, number of packets with data length of 0 bytes, number of protocols, average window size, number of flag codes and average data length. This information is later used in their neural networks algorithm to identify potential outliers. This article focuses more on the machine learning and neural networks part, rather than actually securing an ICS.

3.2.3 Qualitative survey

This section presents the results from a qualitative survey of the 8 chosen papers for full analysis. In Table 3.2 the parameters for the survey can be found. These criteria are selected based on common factors observed from reading the articles chosen during the full analysis in Section 3.2.2. After this, in Table 3.3, the results of the survey can be found. The purpose of this section is to give a quick, objective and structured view of how chosen papers perform according to parameters chosen for the survey. A brief summary of these results can also be seen in Section 3.2.4.

Table 3.2: List of criteria assessed for the various papers surveyed. The second column, Symbol, correlates a certain criterion to a symbol. The third column contains a short description of the criterion.

CRITERION	SYM	DESCRIPTION
Data environment	**A.1**	
Real system	●	The data that is used is originating from a real system.
Simulated system	❍	The data that is used is originating from a simulated system or a testbed.
Both	■	Both real system and simulated system data is used.
Method	**A.2**	
Network traffic	●	Analysis is performed on raw network data.
Process semantics	❍	Analysis is performed on process semantics, such as application data.
Both	■	Analysis is performed on both network traffic and process semantics.
Algorithm	**A.3**	
Algorithm used	-	Only description of algorithm is available.
Code is provided	●	Code is provided.
Mathematical formula	❍	Mathematical formulas are provided.
Both	■	Both above criteria are provided.
Level of data	**A.4**	
TCP/IP	●	Data is from TCP/IP layers.
ICS Protocol	❍	Data is from an ICS Protocol.
System data	■	Data is extracted from a database.
Capture position	**A.5**	
Field network	●	Data is captured from field network.
Control network	❍	Data is captured from control network.
Corporate network	■	Data is captured from corporate network.
Test validation	**A.6**	
Network testing	●	Tested using network tools, such as Nmap.
Semantic testing	❍	Tested using semantics, such as direct and indirect control attacks.
Both	■	Tested using both methods.
Data extraction	**A.7**	
Number of days captured	#days	Number of days captured from the network.
Assumptions	**A.8**	
Various assumptions		Written as short comments.

Table 3.3: List of papers with respect to criteria from Table 3.2. If a paper does not state anything about a certain criterion, ✻ is used to state this.

PAPER	SYM	COMMENT
Communication Pattern Anomaly Detection in Process Control Systems [27]		
Data environment	❍	Test environment is based on a DCS from Invensys Process Systems.
Method	■	Checks network flow, endpoints (src, dst, IP-addr, port).
Algorithm	-	Pattern based & Flow based anomaly detection.
Level of data	●	The data captured is taken on the flow of traffic and includes the TCP/IP-layers.
Capture position	●	Collects TCP traffic on the field network.
Test validation	●	Nmap scans. Decreasing/Increasing flow rate.
Data extraction	< 1	The data is generated for 30 minutes to learn then the anomaly testing starts.
Assumptions	✻	
A Module for Anomaly Detection in ICS Networks [5]		
Data environment	●	Using a real network to test traffic.
Method	●	Focuses on network traffic.
Algorithm	-	Self-Organizing Maps that are used to detect anomalies.
Level of data	●	They mainly look at TCP/IP traffic, but also UDP and ICMP.
Capture position	❍	According to models they capture from control network.
Test validation	●	Using Nmap to test.
Data extraction	✻	
Assumptions	✻	
Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes [4]		
Data environment	■	Data captured originates from two water treatment plants. Also set up a controlled environment as a testbed.
Method	❍	Extracts application data from the network traffic.
Algorithm	■	Autoregression modelling & control limits. Predictions of next value for process values.
Level of data	●	Captured data contains a mix of ICS protocols and non-ICS protocols.
Capture position	❍	Trace capture is from a switch between the PLCs and the ICS servers.
Test validation	❍	Crafts attacks in testbed environment to check for deviations and raising alerts.
Data extraction	14	The data captured was recorded over a period of 2 weeks and 3 days from both plants.
Assumptions	✻	
Challenges and opportunities in securing industrial control systems [6]		
Data environment	●	Only using real systems to validate.
Method	■	Both network traffic and process semantics are used.
Algorithm	✻	No algorithm used.
Level of data	●	TCP/IP flows are used, but also some log data, which is considered System data.

Capture position	✻	Not mentioned, but assumable in the control network.
Test validation	✻	
Data extraction	2	Ranging from 20 minutes to 50 hours.
Assumptions	✻	
Anomaly-based network intrusion detection: Techniques, systems and challenges [24]		
Data environment	✻	
Method	✻	
Algorithm	-	Anomaly based algorithms. Statistical, knowledge-based and machine-learning.
Level of data	✻	
Capture position	✻	
Test validation	✻	
Data extraction	✻	
Assumptions	✻	
Analysis of Intrusion Detection in Control System Communication Based on Outlier Detection with One-Class Classifiers [29]		
Data environment	❍	Simulated in a laboratory.
Method	■	Both network parameters and application specific parameters are used.
Algorithm	❍	Support Vector Machine and Support Vector Data Description is used in order to find outliers.
Level of data	●	They capture packets in order to test, so we assume TCP/IP.
Capture position	❍	IDS is placed inside "Control Center" according to the paper.
Test validation	✻	
Data extraction	✻	
Assumptions		They use human-based interaction in order to whitelist outliers.
Network Traffic Features for Anomaly Detection in Specific Industrial Control System Network [30]		
Data environment	■	Factory site and laboratory equipment.
Method	●	Limited the investigation to network traffic.
Algorithm	-	Self-Organizing-Maps as an extension to Bro scripting language.
Level of data	●	Analysis is based on the TCP/IP layer.
Capture position	❍	Two capture positions in a ring topology.
Test validation	✻	Never makes it to a testing phase in this paper.
Data extraction	< 1	Captured data is about an hour long.
Assumptions	✻	

Neural Network Based Intrusion Detection System for Critical Infrastructures [31]		
Data environment	●	Data was recorded from network traffic from an existing infrastructure.
Method	●	Analysis is applied only to the IP-packets.
Algorithm	❍	Combination of Error Back-Propagation and Levenberg-Marquardt.
Level of data	●	Data is captured from full network traces.
Capture position	❍	Data is captured between a PLC attached to a control PC station with a hub in the middle.
Test validation	●	Intrusion attempts using Nmap, Nessus and MetaSploit.
Data extraction	✻	
Assumptions	✻	

3.2.4 Summary

This section contains a summary of both the quantitative survey and the qualitative survey. We also draw conclusions of patterns that we have observed based on the survey parameters and criteria shown in Table 3.2.

First, a summary of the quantitative survey is shown, in Table 3.4 which contains a small recap of each of the 8 chosen articles, explaining their purpose, method and achieved results. After this, a summary of the qualitative survey is shown, in Table 3.5. This table shows a brief summary of the observed criteria and parameters found in the chosen articles. Lastly, we present our own conclusions of patterns that we have observed throughout the survey.

Table 3.4: Summary of research literature in the field of IDS technology in ICS environments.

Title	Purpose & Method	Results
Communication Pattern Anomaly Detection in Process Control Systems [27]	Find malicious traffic in network traffic flows by analyzing different network communication patterns, such as communication endpoints and traffic flows. Mostly using statistical methods to see the probability that a flow is new or not.	They have performed various tests, e.g. nmap scanning and flow rate changes. Most of the tests were successful and they have announced their own test scores (probability, anomaly score).
A Module for Anomaly Detection in ICS Networks [5]	Find unusual traffic in network traffic by using SOM models. They utilize different parameters such as flow rate and communication endpoints in order to detect changes.	There are not many results provided by the article, but some reasoning about the outcome of the experiments.

Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes [4]	Detect direct or indirect process control attacks by watching variables in the network traffic in an ICS. Compare captured values to predicted values that have been computed from a learning period. Uses autoregressive models.	Provides some result but not many metrics. They claim that most of the attacks were detected, especially attacks against static variables.
Challenges and opportunities in securing industrial control systems [6]	Identify various challenges in the field of securing ICSs. They discuss various issues and techniques that are important for the field, rather than proposing algorithms that can be used.	Their results are mainly about different techniques that are relevant to securing ICSs.
Anomaly-based network intrusion detection: Techniques, systems and challenges [24]	The authors goal is to review anomaly-based techniques and frameworks that are available as well as to list different challenges that exist in the field of anomaly-based detection in IDSs.	A compilation of techniques and frameworks that can be used for anomaly-based IDSs, and different challenges that exist in the field.
Analysis of Intrusion Detection in Control System Communication Based on Outlier Detection with One-Class Classifiers [29]	Detect outliers in control system networks using SVM & SVDD.	A comparison of the two methods and their results respectively.
Network Traffic Features for Anomaly Detection in Specific Industrial Control System Network [30]	Compares different network traffic features for anomaly detection in ICS networks.	Comparison of the authors own data and various statements about network traffic in order to see what parameters that are feasible to use when using machine learning for ICSs.
Neural Network Based Intrusion Detection System for Critical Infrastructures [31]	Detect deviations from normal behavior in the network traffic. The authors use two neural networks algorithms over different windows of traffic in order to detect the unusual behavior.	The authors claim that they have almost perfect hit ratio and no false-positives using their proposed method.

The quantitative survey provides various results. Most of the articles did not provide any detailed information about where their data came from, what kind of assumptions that they made on the network, details on the system model itself or their results. This made the survey quite hard to summarize since most parameters that we wanted to know were left out. However, we gained much knowledge about what kind of approaches that can be applied on different kinds of data, as well on what data to apply them on. Some articles bring forth algorithms that can be used on network statistics such as network flow, amount of packets and various other parameters, whilst some of them bring forth algorithms that can be used on process semantics such as Modbus data.

Table 3.5: A summary of papers with respect to criteria from Table 3.2. If a paper does not state anything about a certain criterion, ✻ is used to state this. A.8 is omitted in the summary.

PAPER	A.1	A.2	A.3	A.4	A.5	A.6	A.7
[27]	❍	●	-	●	●	●	✻
[5]	●	●	-	●	❍	●	✻
[4]	■	❍	■	●	❍	❍	14
[6]	●	■	✻	●	✻	✻	2
[24]	✻	✻	-	✻	✻	✻	✻
[29]	❍	■	❍	●	❍	✻	✻
[30]	■	●	-	●	❍	✻	<1
[31]	●	●	❍	●	❍	●	✻

From Table 3.5 it is easy to spot common properties used throughout the chosen articles. *Note: Article [24] contains various information about using anomaly based network intrusion detection, and do not include any experiments or results. Thus, we do not have any useful information about this article in the table.*

For the criterion about the data environment (A.1), we can see that 5 out of 8 authors have access to data originating from real systems. However, out of these 5 only 3 of them choose to provide information about the amount of data that they have used. For the next criterion, method (A.2), there is a mix of what level of analysis is performed. Only one article focus solely on the process semantics, three only analyze network data and another three analyze both of them.

When it comes to the algorithm (A.3) used in the articles, only two authors provides mathematical formulas and only one provides both sample code and mathematical formulas. Thus, it becomes complicated when attempting to repeat their approach. According to level of data (A.4), one striking pattern is that the authors all use data from the full TCP/IP stack.

When it comes to capture position (A.5), five of the authors uses captured data originating from a control network, one captures from a field network and one fails to mention the position from where the data is captured but shares how many days of data they have access to. A field network generally only contains information received from and sent to a specific or many field stations. A control network generally consists of all control system information, such as sensor readings and commands sent to field stations, as well as information from engineering stations. A corporate network generally contains various other traffic, such as HTTP, FTP or SMTP traffic, thus considered noisy.

When it comes to testing their proposed method (A.6), some of the authors who validate their approach consort to simple network attacks, such as nmap scans. These types of attacks create a large amount of traffic over a small period of time in the network and is not a fair validation test against more sophisticated attacks. However, one group of authors utilize direct control attacks in order to test their approach, but do not provide any detailed information about it. As we just recently mentioned out of the 5 authors who had access to real system data only 3 of them provided information as to the amount of data (A.7). One with less than 1 day, one with 2 days worth of data and the largest data set consists of 14 days, compared to the 106 days of data that we have been able to utilize.

Based on the findings that are presented in the survey and related work, we are going to proceed with certain experiments. We need to validate what other authors write about the traffic and process semantics, and see if their approaches and arguments are applicable to our data as well. In Chapter 5, we perform various experiments in order to answer questions like: How much of the data is regular and automatically generated? How much of the data is generated by human interaction? Can we categorize between different types of events, e.g. time-triggered, user-triggered, event-triggered? Can we characterize device interaction patterns based on a data-driven approach?

4
Risk analysis

Risk analysis is an important subject for ICSs. This is due to the fact that ICSs often control critical infrastructures such as water distribution, power plants and other applications. These applications rely on safe and stable systems in order to avoid putting human life at risk. This chapter is about risk analysis and covers two different parts, a general risk analysis based on papers from the field, and a workshop with a domain expert. We present the risk analysis in order to better understand the system at risk, such as different parts of the processes, what can go wrong and what the consequences could be.

We will use the findings from this chapter to design and conduct experiments, in order to further understand the network data we have been handed, from the ICS. These experiments is further explained in Chapter 5.

4.1 Methodology

The risk analysis is performed as follows: A published source in the topic of security and risk management in ICSs is searched, namely the Guide to Industrial Control Systems (ICS) Security by the National Institute of Standards (NIST). This article is chosen because it provides a solid foundation to the topic at hand, and is also recognized by means of citations. This guide provides a general list of considerations to use when securing an ICS. Further, a workshop together with a domain expert from the industry is also conducted in order to gain a deeper knowledge what their opinions are about the attack risks for ICS. The result from both the published sources and the workshop is presented below in Section 4.2.

4.2 Result

The results from published sources and from the workshop are presented below. The published sources work more as a general list of considerations to use when securing an ICS while the result from the workshop provides a more detailed glimpse about potential attack risks.

4.2.1 General

As stated in Section 4.1, NIST has developed a document that acts as a guide for securing Supervisory Control and Data Acquisition systems, as well as general purpose ICSs [1]. This section is based on the findings in that article.

4.2.1.1 IT systems and ICSs

ICSs have been developed in a different way than ordinary IT systems. For example, most ICSs were built a long time ago, before public and private networks were used, before Internet was used for most operations within a company and before everyone used workstations in order to get their work done.

ICSs were built in order to meet three different important design criteria, Reliability, Maintainability and Availability (RMA). Cyber-security was often not a critical need, since most of the systems were operating within a closed and restricted environment. However, now that IT systems are becoming more low-cost, (for example with cheap IP solutions) many of the old proprietary systems and solutions are being replaced with new cheaper ones. This has created a sudden need for more secure solutions for ICSs, especially since the new IT systems often expose the ICSs to the Internet which make them less isolated.

One way to describe it is IT systems often require Confidentiality, Integrity and Availability (CIA), where confidentiality is quite important. However, in ICSs, Availability is often more important than Confidentiality, which is needed because of the very nature of the networks that they were developed for [32].

NIST has published a list of some special considerations that should be kept in mind when considering security for ICSs compared to IT systems:
Performance Requirements: There are some specific performance requirements for ICSs. They do not rely on high throughput of network traffic, but they do rely on fast delivery and minimal delay.
Availability Requirements: There are rather strict availability requirements on ICSs due to the fact that most ICSs are continuous and often hard to start, stop or restart without consequences for the industrial process.
Risk Management Requirements: Data confidentiality and integrity is not of the utmost concerns when dealing with ICSs. In ICSs, fault tolerance and human safety are more important factors. This is also due to that ICSs often control critical processes that may endanger health and equipment.
Architecture Security Focus: Focus on architecture differs between IT systems

and ICSs. In ICSs, edge nodes of the network, e.g. PLCs, are responsible for the final operations of the industrial process. However, the main controller such as the main node in a SCADA system is still important as well, since it have the ability to control edge nodes as well.
Unintended Consequences: ICSs control industrial processes, which often include interactions with physical processes. It is important that security measures do not interfere with the regular behavior of the industrial process.
Time-Critical Responses: Some operations in ICSs need to provide quick system responses, e.g. user authentication should not slow down emergency actions that are time critical. It is also important that flow of information is not interrupted or compromised in any way.
System Operation: System operation is generally more difficult in ICSs than IT systems. ICSs are often controlled by control engineers, while IT systems are controlled by IT personnel. Also, many desirable security features might not be implemented in the proprietary software and hardware that often is used in ICSs.
Resource Constraints: Devices in ICSs are generally resource constrained in some way, and often lack ordinary IT security features. In some cases there are also issues with ICS vendor license agreements which makes it hard to employ third party security measures.
Communications: Communication protocols, especially those used for intra-processor communication and protocols used for field device control, usually differ from the type of protocols that are used in IT systems. Also, they are sometimes proprietary.
Change Management: Change management is of utmost importance in order to maintain security in both IT systems and in ICSs. Systems that are not maintained and updated can be vulnerable. It is often easy to update IT systems in a timely manner, but it is often more inconvenient to update ICSs.
Managed Support: Service support for ICSs is often via a specific vendor. In IT systems, there are often diversified support possibilities.
Component Lifetime: ICS components, such as PLCs and mainframes, often have a much longer lifetime than components in an IT system. IT system components generally have 3-5 years lifetime, while ICS components often have 15-20 years of lifetime, and sometimes even longer.
Access to Components: ICS components might be isolated and hard to reach, for example far into a restricted area in a service station somewhere, while IT components usually are easier to reach.

4.2.1.2 Threat actors

Another important aspect to keep in mind when developing systems is what threats are present for the system, and what kind of threat actors that exist. There are several kinds of threat actors, seen in the list below [1].

- Attackers that try to break into ICSs for fame and glory.
- Criminal groups that do it for monetary gain.

- Insiders that have been working inside the system that are disgruntled in some way.
- Foreign states that want to keep the upper hand in case of a conflict.
- Terrorists that want to disrupt operation for some political reason or just to create chaos.

These are just some of all threat actors that might compromise the security in an ICS, but are deemed to be the biggest ones according to an industry expert, during the workshop about risks. There are several kinds of threats and vulnerabilities that these threat actors might use in order to compromise the system. The list of vulnerabilities can be long and is often dependent on the actual installation of the ICS.

4.2.1.3 Signs of intrusion

Even though administrators and operators deploy certain security features, there is always a risk that an ICS system might be compromised in some way. There are several signs of intrusions that administrators should look for [1]. These signs might indicate an intrusion or an unintended incident or accident. However, depending on the skill of the adversary these symptoms may not be obvious.

There are several signs related to the amount of traffic, if the traffic level is unusually high for a longer time this might indicate that something is wrong. If the disk space suddenly decreases something might also be wrong. Also, it is suspicious if the CPU usage is unusually high.

There are also several signs that include the topic of user accounts; If new user accounts are created, accounts are locked out, if accounts are used when users are away from work or if someone attempts to use administrator-level accounts. If any of the above signs of an intrusion are detected one should be suspicious and check the account history and take actions.

The log files can also tell if something is wrong. If the log files are cleared one might suspect that someone tried to hide their traces. Also, if the log files are unusually large because of numerous events something might be wrong as well.

These signs are taken into account during the experiments in Chapter 5 and when proposing methods for an IDS in Chapter 7.

4.2.2 Workshop with industry expert

We present the results a workshop with an industry expert about risk management and how actions against risks should be taken. This workshop corresponds and connects to the guide to SCADA systems by NIST [1], summarized in Section 4.2.1. The workshop with an industry expert gave us an inside look to what the expert believe to be risk factors when it comes to the security surrounding an ICS. Below are what the expert believe to be the major risks toward an ICS.

Mobile Net: Using wireless communication in order to communicate with distributed stations is a good option when setting up an ICS. However, it comes with a drawback that it is often controlled by a third party supplier. When the communications leave the local network to traverse the wireless network it is out of the owner's control what happens. Due to varying loads on the network it could cause latency on time critical processes.
The Human Factor: Humans pose the greatest threat to a system. Social engineering is a frequently used tool to gain information or even access to systems. Knowledge about certain parts of a system can go lost when employees quit their jobs. Employees with a lot of knowledge about the system could also be the target for extortion in order to gain access to said system.
Physical Sabotage: Possibly the least complicated attack on a system. It does not require any expert knowledge about any IT system or security. If an attacker successfully trespass on for example a water treatment plant the attacker could poison the water supply. There is also a risk of physically sabotaging piping or the hardware attached to a station should an attacker gain access to them.

Even though parts of the connected network were to temporarily go down, such as the wireless network, every station is capable of fending for themselves automatically, which mitigates some of the attack risks. Moreover, the expert speak about the problem of security versus the price of that security. It would be possible to spend a lot of money in order to have an almost impenetrable system from an IT perspective. This would most certainly only lead to attackers, who are set on attacking the system, to resort to more primitive methods, such as physical sabotage mentioned earlier. This is because it would not be worth the time investment to break into such a system when it is easier to just break it physically. Thus, the impenetrable IT system would be rendered more or less useless and a lot of the invested money and time would be wasted.

Another security risk that the expert mentioned was about third party providers. Using commercial off-the-shelf products could be a risk in that the users do not have full control of the software or hardware. This could lead to backdoors being introduced into a system without the users knowledge because the product has been compromised.

4.3 Summary

This chapter presents a risk analysis, containing both guidelines from a published source as well as what a domain expert deems to be the top risks to ICSs. The first part containing guidelines from a published source, contains some general factors to consider when securing an ICS. This includes differences between normal IT systems and ICSs, threat actors and what signs of an intrusion one might observe. The workshop with a domain expert resulted in three major things to consider when securing an ICS: Using mobile nets, the human factor and physical sabotage.

Based on some of the signs of an intrusion and risks presented in this chapter, we move on to the next chapter where we conduct experiments in order to further understand the network data we have been handed, from the ICS.

5

Experiments and system model

Based on the information yielded from Chapter 4, Risk analysis, we perform experiments to verify some of the risks that we found. We also investigate if the results and the discussions from previous papers that we surveyed in Chapter 3, Related work, are relevant and applicable to our data as well.

The experiments in this chapter should answer questions such as: How much of the data is regular and automatically generated? How much of the data is generated by human interaction? Can we categorize between different types of events, e.g. time-triggered, user-triggered, event-triggered? Can we characterize device interaction patterns based on a data-driven approach?

We have performed these experiments on data that is supplied from the ICS that we have monitored. We have 106 days of traffic from a certain set of hosts. Each capture file consists of a full day's worth of network traffic originating from the SCADA network. The system model and the data is further explained later in this chapter, in Section 5.2, System model and data.

5.1 Experiments

We present the three performed experiments in this section, shown in Table 5.1. As a starting point the data is analyzed using various network analysis applications, such as Wireshark, tshark and tcptrace, which were further explained in Section 2.3. This is done in order to get some basic understanding of the data, how it looks like, who communicates with whom and how often it occurs.

When this is done, we perform an analysis of the data using two already established network monitoring programs, which also can act as IDSs, namely Bro and Snort. This was done as the second experiment so that we might better understand the traffic that originates from the network, also, to see if an off the shelf IDS would detect any suspicious traffic in the network without any extra configuration.

The last experiment is to analyze the actual semantics of the data, which is extracted from the Modbus protocol. We extract various data, such as register values, masters and slaves in the network, time between messages, number of messages sent and what kind of Modbus commands that are used.

Table 5.1: Overview of what experiments are conducted in this chapter.

Experiment	**Purpose**
Exp 1: *Investigate network traffic*	Gaining a greater understanding for how the network traffic looks like is very important in order to get a better overview of the communication. Using network analysis applications to see how often stations communicate, what they are sending and who is sending messages to whom.
Exp 2: *Evaluate off-the-shelf IDSs:* - Bro - Snort	We wanted to know if two off-the-shelf IDS applications would be able to detect anything of interest, without any special configurations, about our dataset.
Exp 3: *Modbus analysis:* - Modbus commands - Classification of Modbus registers - Modbus register values	Investigating the process semantics is the final experiment conducted. We analyze the Modbus protocol by looking at the types of commands, the number of commands and classify registers using the classification scheme by Hadžiosmanović et al. Lastly we look at the values of each Modbus register in the network.

5.1.1 Experiment 1: Investigate network traffic

In order to get an understanding of how the network traffic looks like we performed an analysis with three network analysis applications: Wireshark, tshark and tcptrace. The expected outcome of this step was to answer questions such as how much data that is flowing in the network, which hosts communicates with whom, when and how data is sent and what kind of data that is sent.

The analysis using **Wireshark** is done in order to get some general understanding and graphical view of how the data looks like. Various analysis tools in Wireshark are used in order to answer some basic questions. The analysis should give some information about the response trip time (RTT), source and destination IP-addresses in the network, throughput, time sequence graphs and window scaling over time.

The analysis in **tshark** is done in order to get a better view of how the traffic looks like over days, and should provide viable output that can be processed using scripts. This analysis should provide extensive information that can be used in order to track how the network changes over time, how much traffic is sent between which hosts or what kind of protocols are used. The analysis using tshark has been performed with four certain scans: conversations scan which includes information about who communicates with whom, endpoints which shows information about all the endpoints in the network, non_tcp which shows information about all traffic in the network which is not TCP traffic and non_502 which shows information about all traffic that does not go over port 502 (the default Modbus port). Listing 5.1 shows what commands we run on the data.

Listing 5.1: tshark commands issued on the capture files.

```
tshark -q -z conv,eth -z conv,ip -z conv,tcp -z conv,udp
tshark -q -z endpoints,ip -z endpoints,tcp -z endpoints,udp
tshark -Y "not tcp"
tshark -Y "not tcp.port == 502"
```

The analysis in **tcptrace** is performed as a secondary option to tshark, in order to both verify the data from tshark and to provide some complementary information.

5.1.2 Experiment 2: Evaluate off-the-shelf IDSs

In order to further understand how off-the-shelf IDSs can detect intrusions and other suspicious traffic, we run Bro and Snort on the dataset, which is further explained in Section 5.2. Both IDSs are run on the dataset without any additional configuration, thus only showing warnings according to the default Modbus ruleset.

5.1.2.1 Bro

Bro, which has been introduced and explained in Section 2.2.1, is used in order to get some early insights in what an off-the-shelf IDS could say about the traffic that we have. Bro can analyze the Modbus protocol, which is used as the application protocol in our data. Bro outputs five different log files that are going to be interpreted in order to understand whether traffic is harmful according to the default settings in Bro. These are the different log files that Bro generated:

- conn.log - Contains connection information, such as source address, timestamps, port and protocol.
- dns.log - Contains much of the same information as conn.log and it also contains queries regarding DNS information with query class and class name.
- modbus.log - Contains useful information about which commands sent over the network, for example commands from the master to the slave.

- packet_filter.log - Contain information about the packet filters used.
- weird.log - Logs entries from the capture file that Bro finds odd.

In order to gain further information useful for data analysis, Bro was also run using two Bro scripts that probe traffic. The first command that is shown in Listing 5.2 will generate, after a successful analysis, a log file that contains all the register values and the changes to register values of slaves in the Modbus network in the given capture file.

Listing 5.2: Bro script that will log register value changes over the network traffic.

```
bro -r *pcap-file* track-memmap.bro
```

The other command shown in Listing 5.3 will generate a log file containing all known Modbus masters and slaves in the network.

Listing 5.3: Bro script to capture masters and slaves in the network traffic.

```
bro -r *pcap-file* known-masters-slaves.bro
```

5.1.2.2 Snort

Snort, which is discussed in Section 2.2.2, is commonly used as an IDS that runs like a daemon in the background. When running as a daemon it produces alerts and logs strange behavior according to a set of rules defined when first started. It can also run a capture file through its set of rules and analyze it for any possible intrusion, where an example is shown in Listing 5.4.

Listing 5.4: Snort command to read a pcap file and analyze it according to rules defined in snort.conf

```
sudo snort -c /etc/snort/snort.conf -r *pcap-file* -l *logdir* -b -q
```

5.1.3 Experiment 3: Modbus analysis

Process semantics is deemed to be an important part of ICSs, since many authors analyze it, according to the survey presented in Section 3.2. Since the application protocol used in the acquired dataset is Modbus, a Modbus analysis is performed. The analysis consists of three separate experiments; An analysis of Modbus commands is performed, further explained in Section 5.1.3.1, in order to see how commands are exchanged in the network, which gives a better understanding of the dataset. Also, a classification of Modbus registers is performed, further explained in Section 5.1.3.2, in order to better understand how different registers look like and how they act. This is also partly done in order to ease the last experiment. Lastly, an analysis of Modbus register values is performed. This step is performed in order to get further understanding of how the process semantics of the system acts, as well as to see if data-driven methods used by previous authors presented in the survey, in Section 3.2, is applicable to this dataset.

5.1.3.1 Modbus commands

In order to further understand Modbus, an analysis of the Modbus commands is performed. The expected outcome of this step is a list of all the Modbus commands that is used within the system and to see how many of each command that are executed each day, for all the substations in the ICS network. This is useful in order to see the normality of the data, especially the normality of the process semantics and how communication between the Modbus masters and the Modbus slaves is performed.

5.1.3.2 Classification of Modbus registers

Classifying registers that are used in the Modbus protocol can be convenient in order to better understand how the registers look like and how they act. However, there are several ways to classify registers. Hadžiosmanović et al. [4] classify registers into three different categories:
Constants - Registers that never change their value. Hadžiosmanović et al. [4] claim that this is common for process settings, such as setpoints.
Attributes - Registers that stay within a certain enumeration of values. Hadžiosmanović et al. [4] claim that this one is common for program state and reporting.
Continuous - Registers that continuously change values over time. Hadžiosmanović et al. [4] claim that this one is common for various measurements, such as sensor values.

Hadžiosmanović et al. [4] use a heuristic approach in order to classify variables into the three different categories, which according to them and their interviews with field experts should give a good enough estimation of the class of a register. Their approach is to count the number of distinct values that a register holds for a certain time period. If the register only contains one value during the classification period, it is deemed to be a constant register. If the register holds 2^k or fewer values during the time period, they choose $k = 3$, the register is deemed to be an attribute register. If the register holds more than 2^k values during the time period the register is deemed to be a continuous register. We have chosen to follow their approach.

5.1.3.3 Modbus register values

Many of the authors in the performed survey in Chapter 3, used data driven approaches, such as using machine learning, on the register values in the system. They did this to see that the registers followed a stable and consistent flow because of the very consistent and homogeneous nature of ICSs. The expected outcome of this step is to get an understanding of how register values change over time and see if the methods proposed by other authors are applicable to register values found in our data set as well. The aim is to understand the behavior of several registers, and especially see the differences between constant registers, attribute registers and continuous registers.

5.2 System model and data

In order to get realistic results from the experiments, we run the experiments on data from a real world ICS. This section presents how the network, where the captured data originates from looks like, both from a logical perspective as well as a network map. The data captured is a 105.2GB large dataset, and consists of 106 days worth of network traffic.

How the SCADA system communicates can be seen in Figure 5.1. The system has an HMI where operators can observe values, like pressure and the water level of a tank, from different substations or issue commands to substations. The SCADA system runs almost exclusively automatically without any human intervention.

5.2.1 Data statistics

The data that has been used in the analysis is network traffic captured from a water treatment plant, where the logical structure can be seen in Figure 5.1, and was captured over a period of 106 days which adds up to 105.2 GB of data. The data captured per day decreases over this period of 106 days due to the fact that the system the captured data originates from is being transferred to be placed into a new system. The first couple of days the number of sent packets are about 26 million and after the first month it is down to about 9 million packets. The number of hosts connected in the captured network is 24, as shown in Figure 5.2.

5.2.2 Network model

A general picture of how the network model is set up can be observed in Figure 5.1. This is how the part of the complete ICS looks like from where all the captured data originates. The figure shows an example where the network contains 4 substations. This amount varies during the acquired capture period.

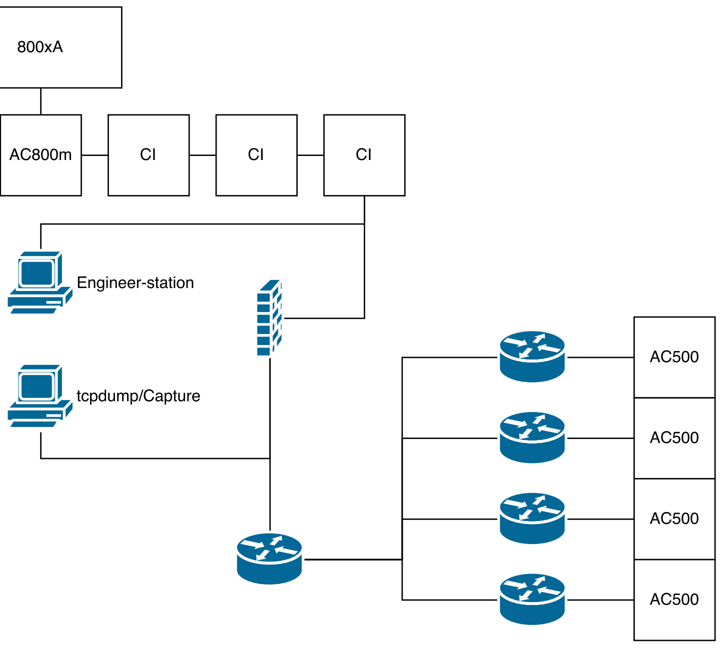


Figure 5.1: Construction of ICS network in general.

The *800xA* seen the at top part of the figure is the control server in the ICS where the HMI is located. Below that is the *AC800m*, which is the control server relaying commands to all the slave PLCs in the network. The *Communication Interfaces (CIs)* are used to send the Modbus commands over the network to all the slaves. There are three CIs in Figure 5.1, which each enables different data link protocols. An engineer-station is connected to the network in order to troubleshoot or run diagnostics on the network. Then, there is a firewall between the *CIs* and the router. The captured data originates from the link between the firewall and the router. The router to router communication is wireless and lastly is the *AC500* which is the PLC at each substation in the SCADA network.

5.2.3 Modbus model

The Figure 5.2 shows all the connected hosts which are present at the start of the 106 days of network captures. It shows that there are two Modbus masters and the rest are slaves. The IP-address of each slave represents a substation. Both masters are placed in subnet 10.34.211, and most slaves are placed in subnet 192.168.211, except for 2 nodes that are placed in 192.168.212 and 1 node placed in 192.168.215, for reasons unknown to us. The size of the network varies throughout the acquired capture period, as seen in Figure 5.3. There is a drastic change in the beginning of the capture period, and then it stays rather static for a couple of weeks, and then all hosts are phased out completely.

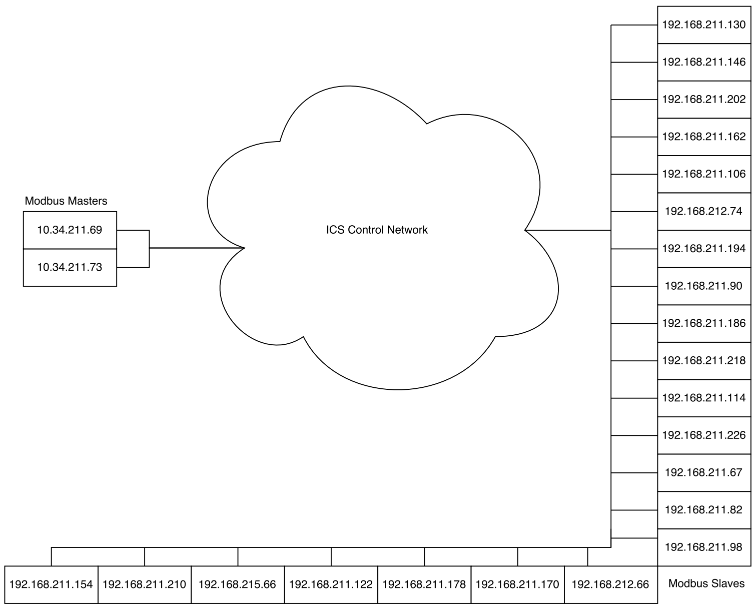


Figure 5.2: Logical distribution of Modbus nodes across the network.

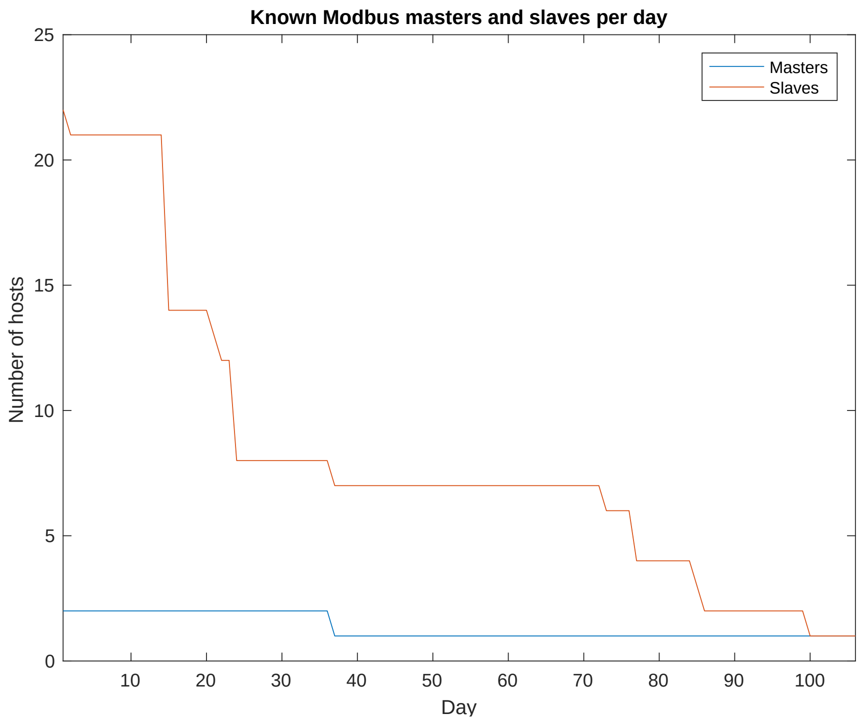


Figure 5.3: How the number of Modbus masters and slaves change over time.

6

Results from experiments

This chapter gathers the results from the experiments described in Chapter 5. These experiments can also be observed in Table 5.1. Each section in this chapter describes the results of a specific part of the experiments performed.

First, we present results from the first experiment, an investigation of network traffic, in order to get further understanding of how data flows through the network. Secondly, we evaluate off-the-shelf IDSs in order to find out if they detect suspicious behavior in the dataset used. Finally, we perform analysis on process semantics, in this case, the Modbus protocol. Here we look at commands sent and received in the network, classify registers and look at the behavior of Modbus registers.

The findings presented in this chapter will be used in the upcoming chapter, Chapter 7, where methods that can be used in an IDS for ICSs is presented and evaluated.

6.1 Experiment 1: Investigate network traffic

This section contains the results from the first experiment presented in Section 5.1.1. We only focus on Wireshark and tshark, since tcpdump yielded the same kind of results as tshark. As stated in Section 5.1.1, these experiments are designed to show how data flows through the network, which is described in Section 5.2.

The results shown in Figure 6.1 and in Figure 6.2, are based on an analysis between two hosts in the network using Wireshark, namely one Modbus master and one Modbus slave. There were a total of two Modbus masters at the start of the data capture and the master with the most activity is chosen for analysis. The slave is chosen arbitrarily because the behavior between slaves are similar.

In Figure 6.1 we see how the round trip time changes over time during the first day, which is when the network is most busy. As seen in the picture, the deviation is rather large in percent, but quite small considering how round trip times could deviate. The peaks are few, and the distribution is mostly located within the interval of 5 to 13 ms. The graph begins at noon and continues for 24 hours.

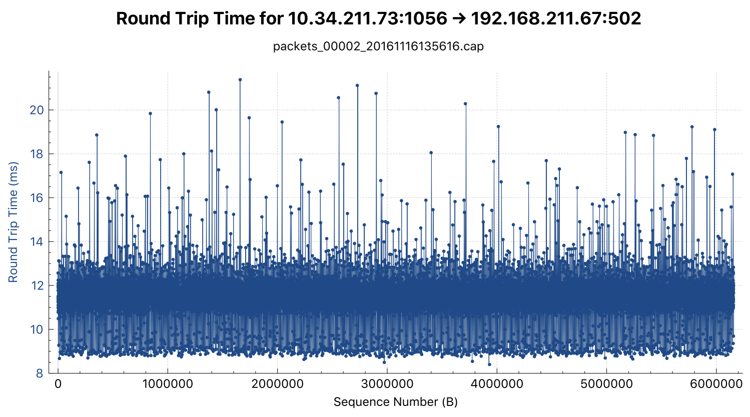


Figure 6.1: Round trip time within the network during one day.

The deviation in window scaling is low, both when considering the deviation in percentage and values. This shows that the size of packets are very static, since the TCP/IP stack never try to raise the window scaling to high values.

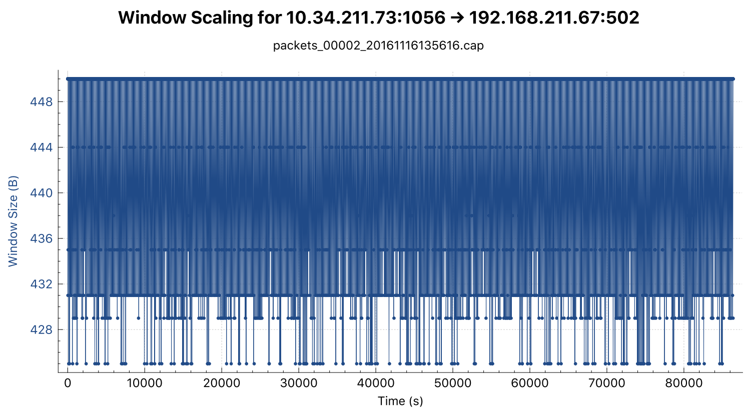


Figure 6.2: Window size within the network during one day.

The tshark results focus on the amount of data that is transferred each day for the first week of data. As seen in Figure 6.3 the number of bytes sent per day per host to the control center, on a conversation basis, is very static.

Figure 6.4 also shows the same kind of results, but consider the amount of data transferred per endpoint instead of per conversation to the control center. This means that we consider the total amount of traffic transferred per host, rather than the total amount of traffic transferred to the control center. These figures are very similar.

As seen in both Figure 6.3 and Figure 6.4, one could group certain nodes in the network together into 3 groups. At the top we have one node which is handling a considerably larger amount of data than the other nodes. Then we have a cluster of nodes around $1.3 * 10^6$. Below that we have some scattered nodes. The node that sends most data is a Modbus master, namely the control center, which indeed should transfer the highest amount of data, since all traffic should go either from it or to it.

The deviation that is shown in Figure 6.5 also supports the claim that data flow within the network is static, since the deviation is only a few percents day to day.

Finally, Figure 6.6 shows how the distribution of bytes transferred one day looks like for the entire capture period, 106 days, for the substation that is located in the network for the longest duration. As seen in this figure, most of the occurrences are located in the interval of 64 MB to 71 MB, with a strong majority of size 65 MB and 66 MB.

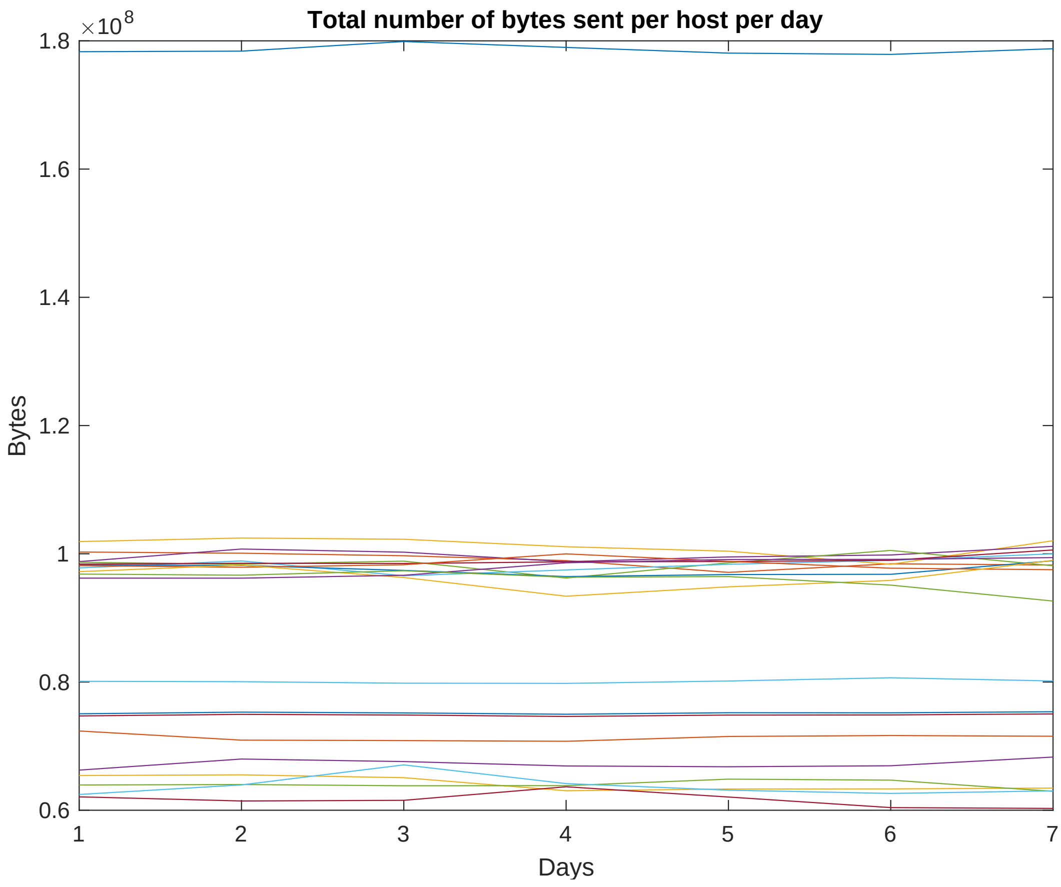


Figure 6.3: Total bytes sent per host per day to the control center for a duration of 7 days.

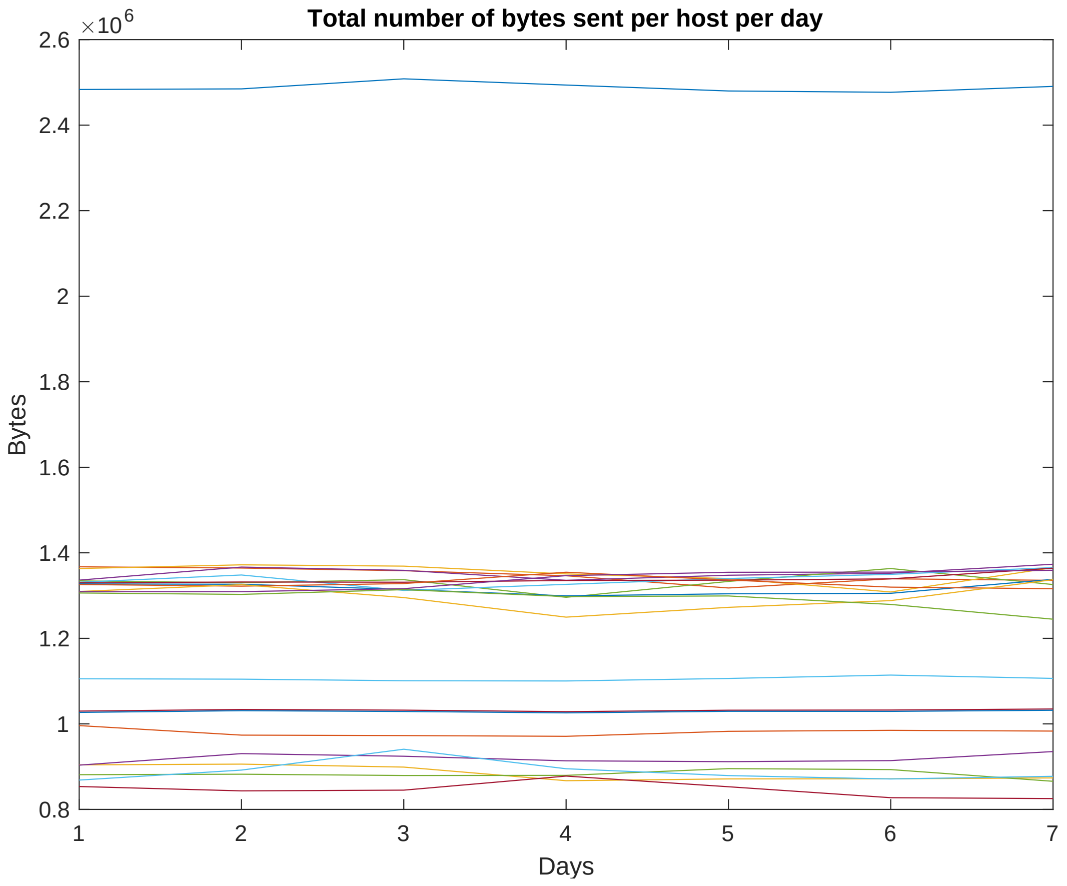


Figure 6.4: Total bytes sent per endpoint per day for a duration of 7 days.

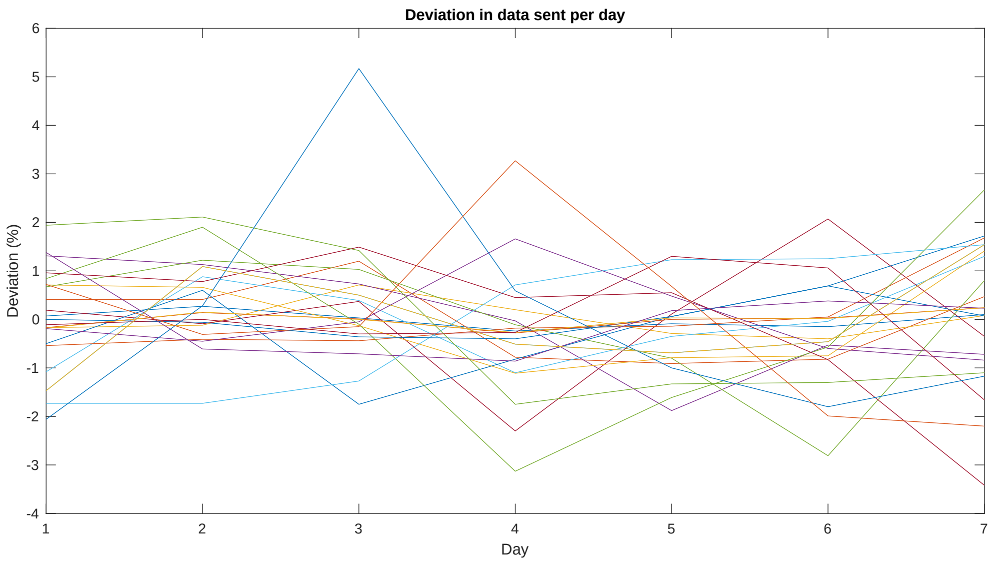


Figure 6.5: Deviation per substation in percentage, considering total bytes sent per day. Each color is a substation.

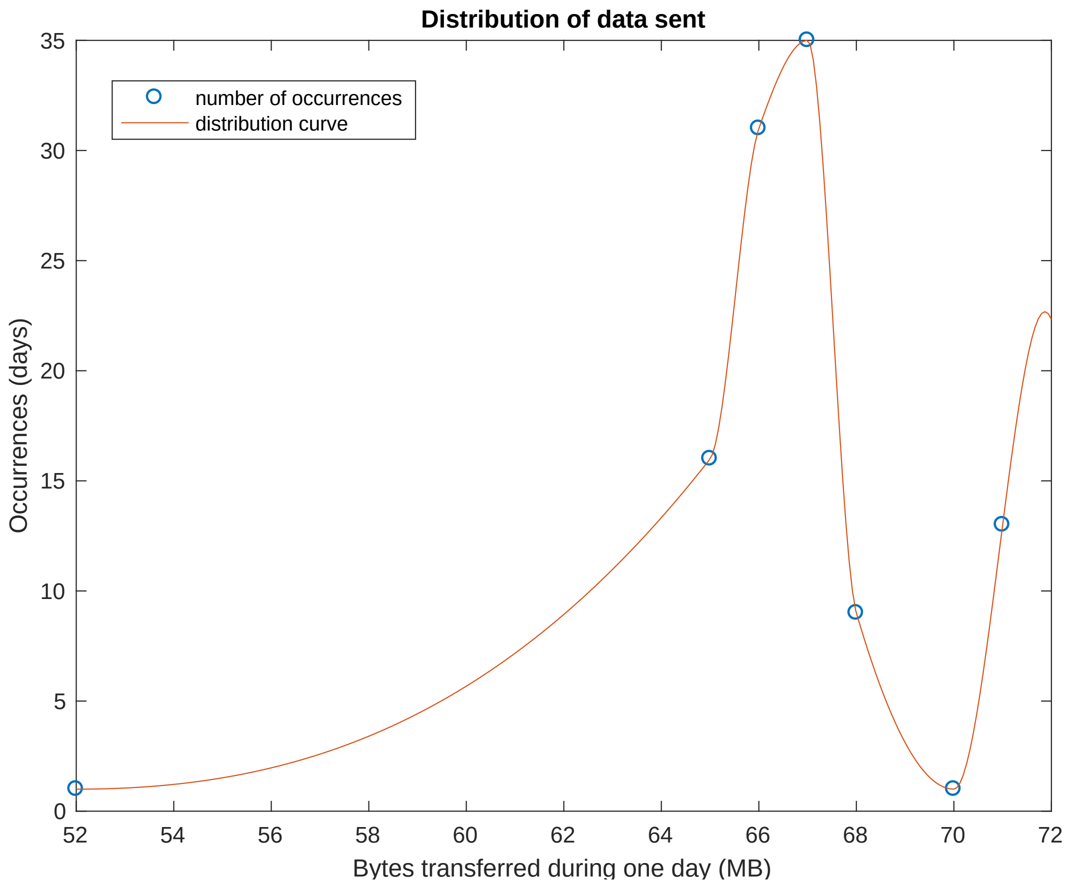


Figure 6.6: Distribution that describes the occurrences of a specific amount of data transferred in MB per day. The data is from the entire capture period for the substation that is in the network for the longest time.

In the tshark analysis we can also see that the duration of each conversation between the control center and the substations is 86,000 seconds per day, which exactly accounts for one day. This means that the conversations are active all the time as long as no errors occur within the network.

The purpose of this first set of experiments is to confirm any assumptions about the static nature of an ICS network traffic. In order to confirm this we looked at six different properties; round trip time in the network communication, window size of packets, deviations in amount of bytes sent per day, distributions over the amount of bytes sent per day, bytes sent per day per endpoint and bytes sent per day per host to the control center. As seen in the figures mentioned above, the amount of bytes does not change over the captured period per host and the deviations in the amount of data sent is also static. This shows that the network behavior from the ICS network where our data originates from is static.

6.2 Experiment 2: Evaluate off-the-shelf IDSs

This section showcases the results from an evaluation of two off-the-shelf IDSs, explained in Section 5.1.2, Bro and Snort. The purpose of this experiment is to see if the IDSs would find anything useful in the dataset used, without any extra prior configuration.

6.2.1 Bro

The results in Figure 6.1 shows the "weird" log from Bro, which outputs any unusual or suspicious traffic that it does not recognize. All "dns_unmatched_msg" warnings are actual valid Modbus traffic. All "unknown_packet_type" warnings are ordinary ARP messages that request the MAC-address of the router or firewall the is present in the part of the network where the data capture was performed. The false positive rate is therefore 100% in the weird log from Bro in this case, because all network traffic here is legit.

Listing 6.1: Bro weird output for analysis of data from the first day of capture

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   weird
#open   2017-02-15-17-24-46
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p       name    addl    notice  peer
#types  time    string  addr    port    addr    port    string  string  bool    string
1478703211.133785       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478703647.008897       -       -       -       -       -       unknown_packet_type     -       F       bro
1478703816.381725       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478704432.546827       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478704840.059512       -       -       -       -       -       unknown_packet_type     -       F       bro
1478705039.804364       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478705653.073625       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478706033.109686       -       -       -       -       -       unknown_packet_type     -       F       bro
1478706260.357534       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478706376.377928       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478706874.628679       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478707226.160264       -       -       -       -       -       unknown_packet_type     -       F       bro
1478707481.881383       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478708094.136706       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478708419.210516       -       -       -       -       -       unknown_packet_type     -       F       bro
1478708704.395119       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478709316.649905       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478709612.261538       -       -       -       -       -       unknown_packet_type     -       F       bro
1478709934.892092       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478709976.405575       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478710536.136381       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478710805.312401       -       -       -       -       -       unknown_packet_type     -       F       bro
1478711145.341326       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478711757.328604       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478711998.364288       -       -       -       -       -       unknown_packet_type     -       F       bro
1478712366.845634       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478712979.086906       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478713191.414869       -       -       -       -       -       unknown_packet_type     -       F       bro
1478713576.421757       -       -       -       -       -       dns_unmatched_msg       -       F       bro
1478713602.544837       -       -       -       -       -       dns_unmatched_msg       -       F       bro
#close  2017-02-15-17-25-02
```

The majority of the traffic that can be observed from the conn.log file is plain Modbus traffic passing through on port 502 which is the standard port for the protocol. This is just the normal communication between master and slaves in the SCADA network. There are mostly persistent connections between the substations and the control center, However some substations seem to lose their connection to the control center and must therefore reconnect which can also be observed in the log file. All of this traffic is send over TCP although there are some UDP traffic in the network. The UDP traffic is exclusively used over port 137 and 138 and these are used in the NetBios protocol. 137 is used for DNS and 138 is used for datagram services.

From the known_modbus.log one can observe the relationship between devices. The captured data shows that there are two masters and the rest are slaves. This is something that also can be observed from the conn.log due to the heavy traffic outgoing from two specific IP-addresses.

6.2.2 Snort

Out of the 106 days of data that is subject to analysis Snort reacted to 4 of them. The packets that were caught by Snort is ICMP packets, simple ping request and replies. All the Snort alerts are from ICMP requests and replies, always from the same originating computer. In Figure 6.2 you can see how the alerts look like in Snort. These alerts are confirmed to be from an operator of the ICS and are therefore considered harmless.

Listing 6.2: Snort alert on ICMP packets.

```
[**] [1:384:5] ICMP PING [**]
[Classification: Misc activity] [Priority: 3]
12/09-12:42:13.427034 10.34.211.71 -> 192.168.211.82
ICMP TTL:128 TOS:0x0 ID:9184 IpLen:20 DgmLen:60
Type:8  Code:0  ID:512   Seq:2560  ECHO

[**] [1:408:5] ICMP Echo Reply [**]
[Classification: Misc activity] [Priority: 3]
12/09-12:42:13.436225 192.168.211.82 -> 10.34.211.71
ICMP TTL:251 TOS:0x28 ID:9184 IpLen:20 DgmLen:60
Type:0  Code:0  ID:512  Seq:2560  ECHO REPLY
```

6.3 Experiment 3: Modbus analysis

This section will go through results yielded from Modbus related experiments. We present results from an analysis of Modbus commands that shows how the rate of Modbus commands change from day to day. Also, we present a classification of Modbus registers that is useful when determining the behavior of registers. Lastly, we present an analysis of how different classifications could look like from the real world data that we have.

6.3.1 Modbus commands

While analyzing the capture files through Wireshark only focusing on the Modbus traffic, there were only three different commands that were sent in the network. These signals are further explained below. The three commands are sent from the master device with either a command to write values to registers, or for the slave device to report its register values, or in order to test communication.

WRITE_MULTIPLE_REGISTERS: This command is sent to a slave device from a master device. The signal contains information about how many registers to write to, which registers to write to and which values to write to each register [33].
READ_HOLDING_REGISTERS: A master device sends this command to a slave device. The request contains how many registers to be read from and from which address to start reading from. The slave responds with the values of each of these registers [33].
DIAGNOSTICS: A master device sends this command in order to troubleshoot the communication between a master device and a slave device [33].

Using the approach mentioned in Section 5.1.3.1 to extract information about the Modbus commands sent in the network, seen in Figure 6.7. The figures depict the total amount of commands (write, read, diagnostics) that are sent to each substation over a period of 14 days. Each line on the figure represents a substation. The amount of commands sent to each substation is consistent up until day 13 when the amount of traffic goes down before day 14 where several substations stop receiving any data. This is because some substations from the ICS where the data is captured from is being transferred over to another network.

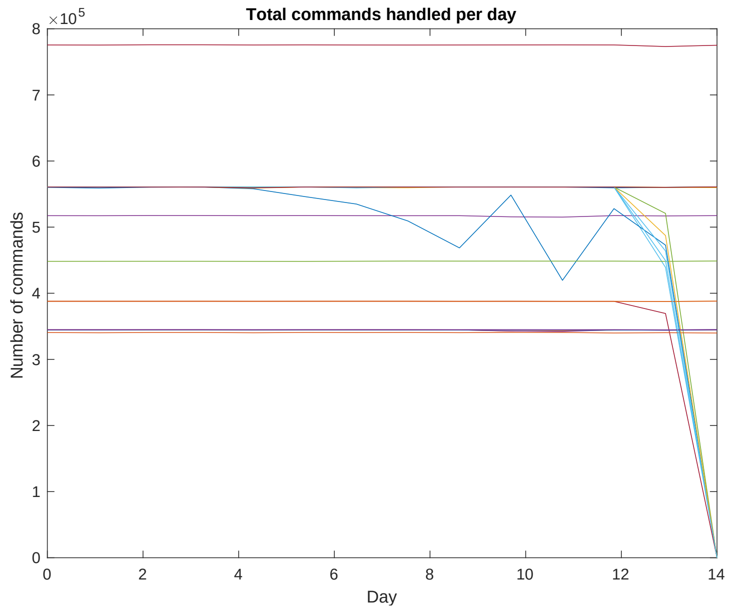


Figure 6.7: Total commands for all the available substations over a period of 14 days.

As seen in the figure above, total amount of Modbus commands, the network behavior is static. There is little deviation up to day 13, which is, as stated earlier, because some substations are removed from the ICS network here. In order to further show the static behavior of the ICS network, Figure 6.8 is presented below. As seen in this figure the deviation for most of the substations is minimal. However, two stations experience quite a lot of deviation compared to the other stations, but only +-1.5%.

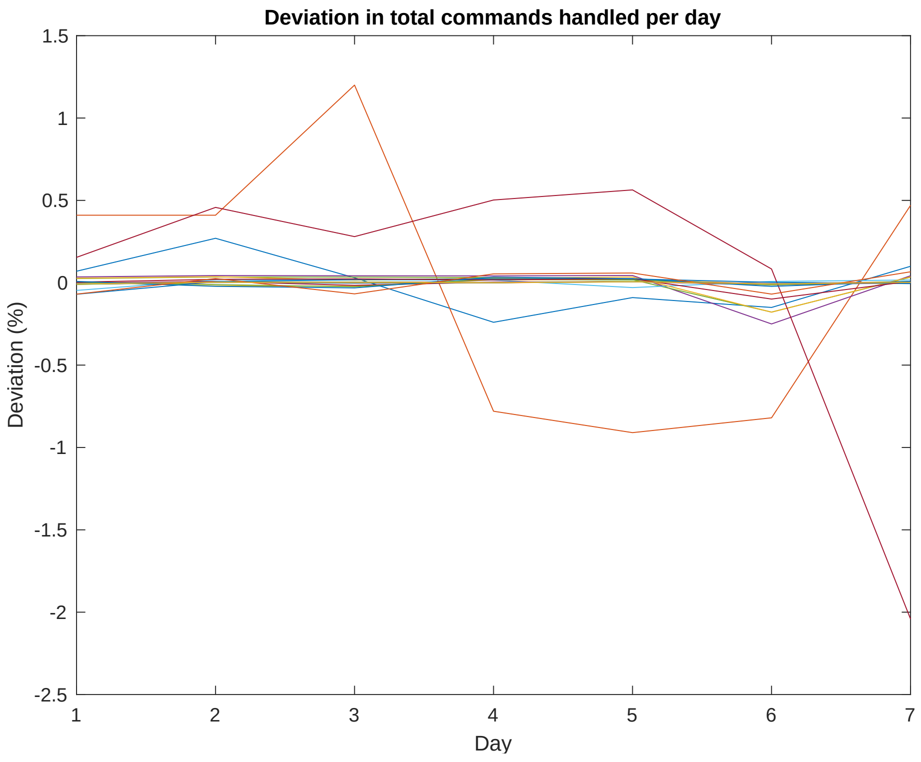


Figure 6.8: Deviation per substation in percentage, considering total commands handled per day. Each color is a substation.

Figure 6.9 shows how many commands a certain substation of the ICS gets per hour for the duration of one day. As can be seen, the amount of traffic sent to the substation is consistent throughout the given day.

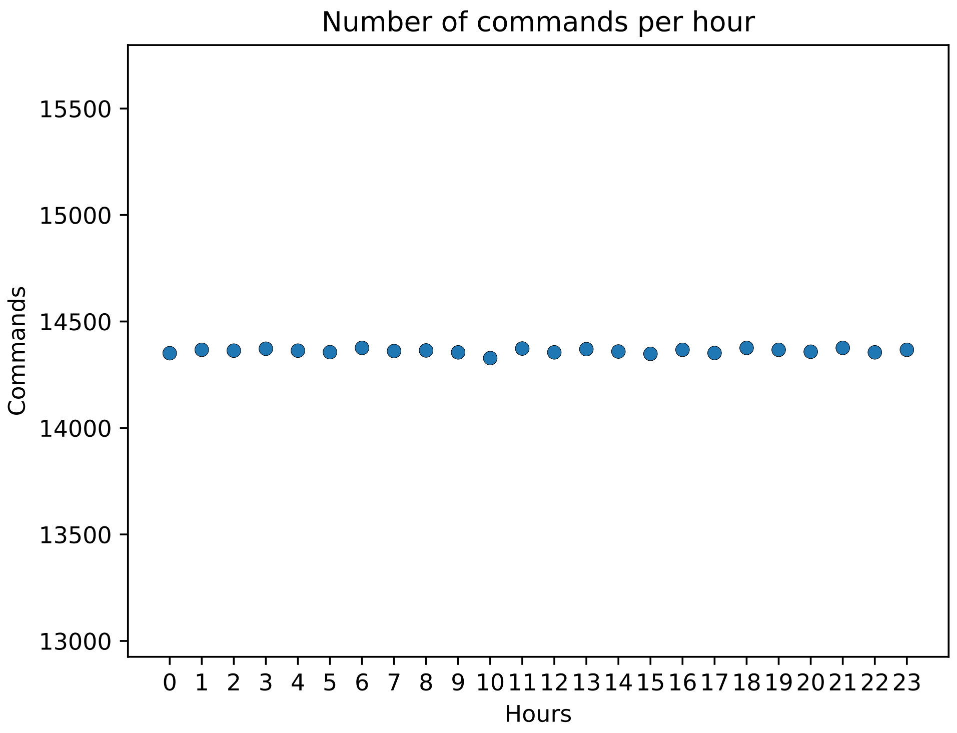


Figure 6.9: Commands per hour from one substation.

Correlated to the network traffic results from Section 6.1, the amount of commands sent each day in the network does not fluctuate, the deviation per substation is also minimal and the commands per hour remains approximately the same. As can be observed in the three previously showcased figures.

6.3.2 Classification of Modbus registers

In order to get further understanding of how Modbus registers act, a classification is performed. The approach when classifying registers is further explained in Section 5.1.3.2.

When Hadžiosmanović et al. [4] use their approach on their own data, they achieve the classifications shown in the second column of Table 6.1. As seen in the table they have a large amount of constant registers in their data set, and only a few occurrences of attributes registers and continuous registers. When we apply their classification approach on our data for one day, we achieve the classifications shown in the third column of the same table. Another thing to note is that we receive another classification when we run the classification for a period of 14 days instead, which can be seen in the last column of Table 6.1. The difference between the table that shows a classification over 1 day compared to the one with a classification of

14 days is that some of the attribute registers have become continuous registers, and some of the constant registers have become attribute registers and continuous registers.

Table 6.1: Result of the classification algorithm used by Hadžiosmanović et al. [4]

Classification category	Hadžiosmanović (3 days)	Our data (1 day)	Our data (14 days)
Constants	95.5 %	5.95 %	3.75 %
Attributes	1.4 %	34.17 %	32.24 %
Continuous	3.1 %	59.88 %	64.01 %

Our result of the classification shows the opposite, since we only have a few constant registers in our data set and a majority of attribute and continuous registers, where continuous registers stand for most of them. As seen in the tables above, our result clearly differs from their result. Our number of constant registers are only 3.75 %, while theirs is 95.5%. Also, our number of attributes and continuous signals are far higher than theirs. As stated earlier, a noteworthy observation is that our results also differ between a 1 day evaluation, compared to a 14 day evaluation. This shows that registers that are constant during 1 day might be attribute based over 14 days, and registers that are attribute based for 1 day might become continuous after 14 days. Examples of how the registers look like are given in the upcoming Section 6.3.3.

6.3.3 Modbus register values

Here in this section we present graphical representations of how register values behave in our real world data. This is important in order to further understand what kind of methods that can be used to detect anomalies and deviations from a desired behavior, as explained in Section 5.1.3.3. Based on the classifications performed in the previous Section 6.3.2, we show how the three types of classifications look like by means of register values.

Figure 6.10 shows how a continuous register behaves. The graph showcases a static behavior, apart from the three anomalies that occur.

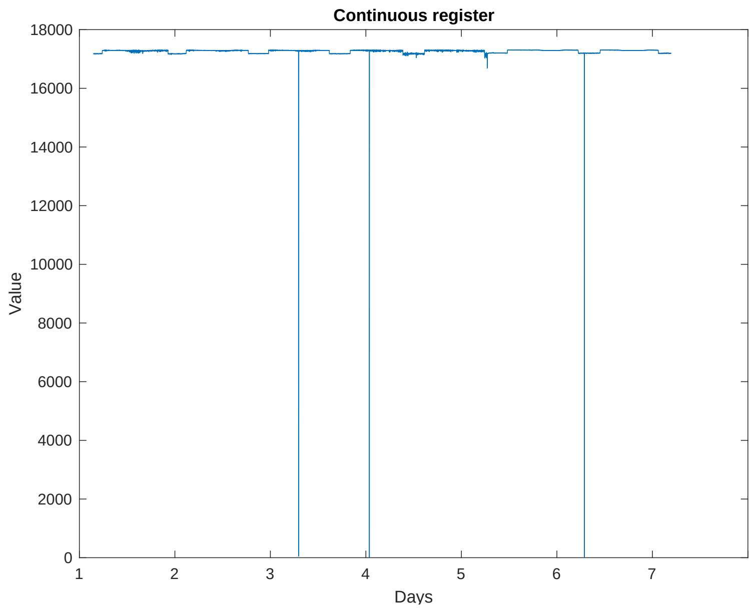


Figure 6.10: Values from a continuous register for a duration of 7 days.

Figure 6.11 shows how another continuous register behaves. Compared to the previous continuous register, this one is not very static but quite predictable within certain intervals. In the end one can see a change in the behavior of the values, which could be due to a change in the ICS.

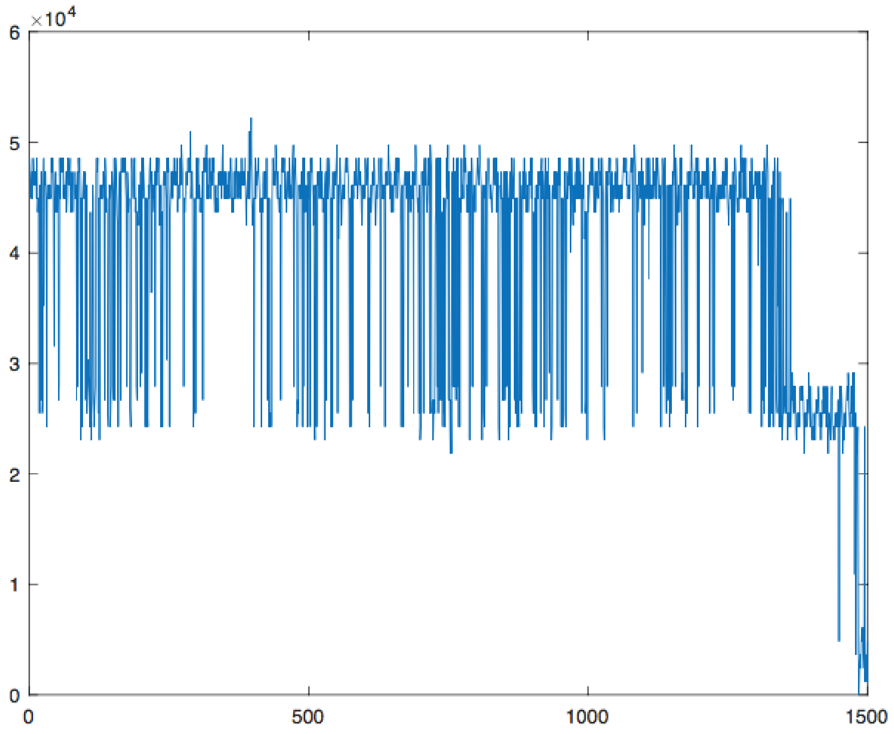


Figure 6.11: Values from a continuous register for 1500 register changes.

Figure 6.12 shows how another continuous register could behave. This register also shows a static behavior, except for two anomalies. This register is similar to the behavior of the register shown in Figure 6.10.

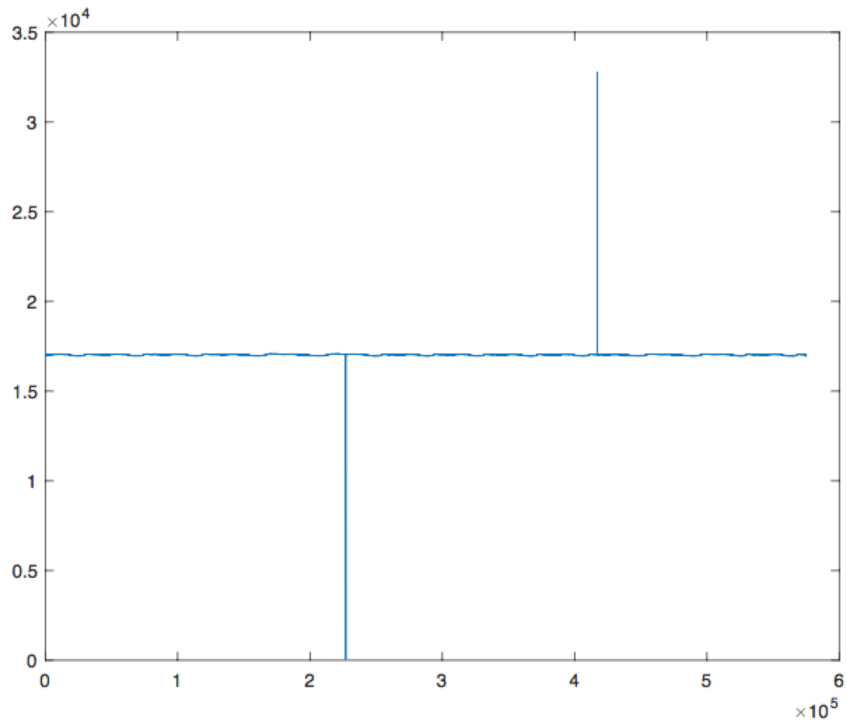


Figure 6.12: Values from a continuous register for 10^5 register changes.

Figure 6.13 shows how an attribute based register behave. This register only change between values from a set of size 3.

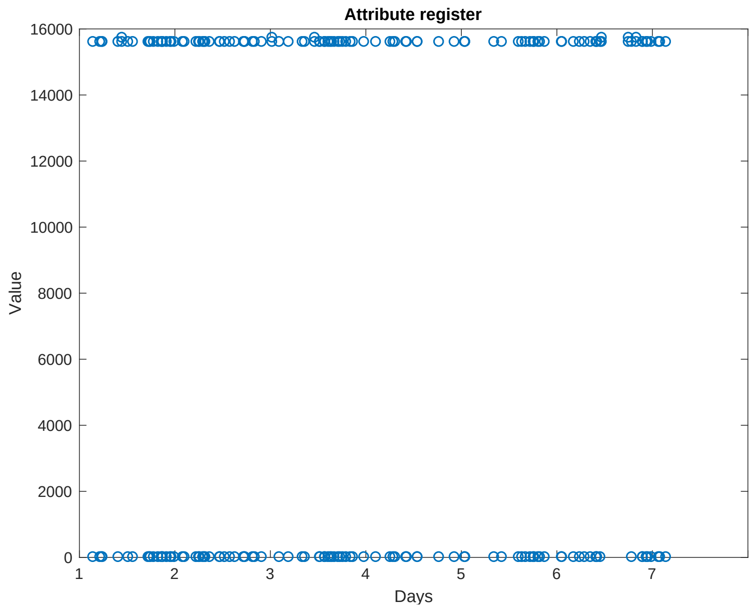


Figure 6.13: Values from an attribute register for a duration of 7 days.

Figure 6.14 shows another attribute based register. This register also go between 4 different values.

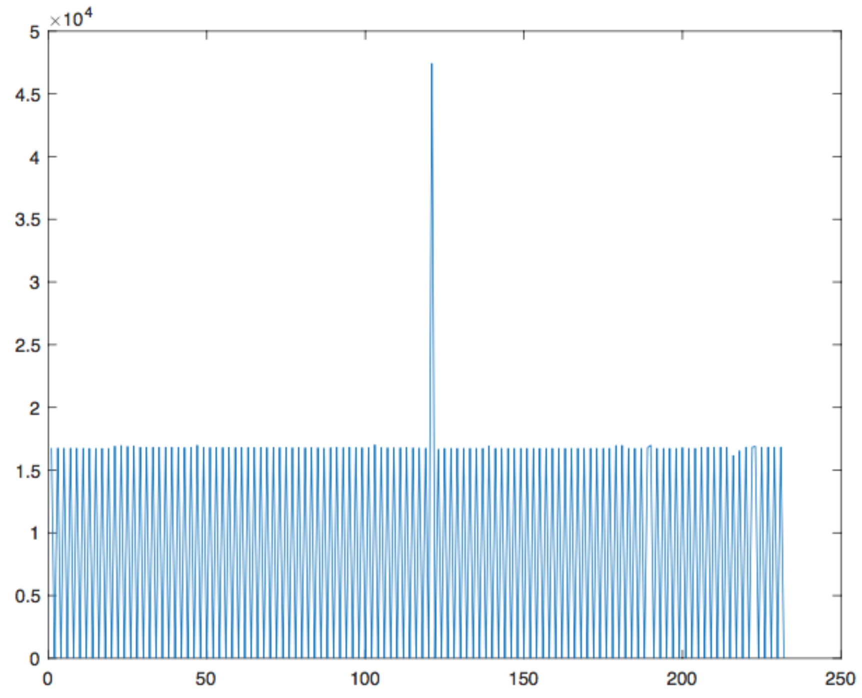


Figure 6.14: Values from an attribute register for 250 register changes.

Figure 6.15 shows how a constant register behaves. As seen in the figure, the value never changes.

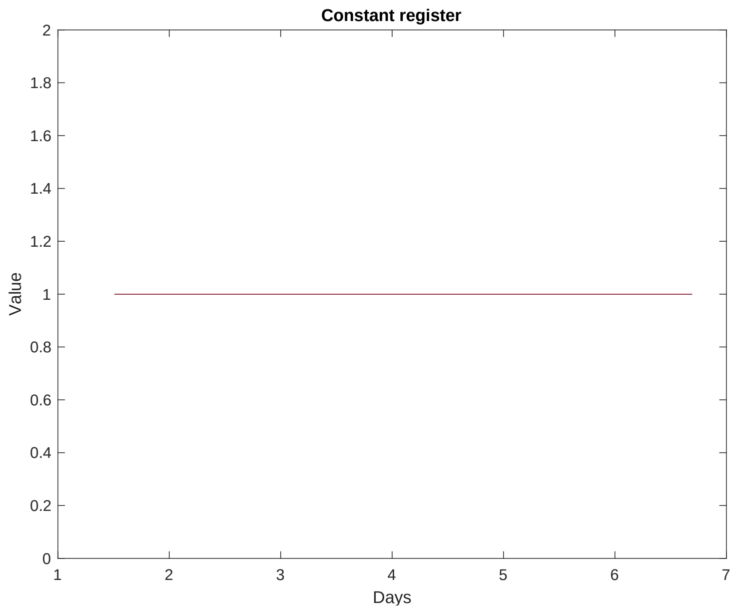


Figure 6.15: Values from a constant register for a duration of 7 days.

6.4 Summary

Here we present results from the three performed experiments. In the first experiment we investigate network traffic and the graphs show that the behavior of the network is static. The total amount of bytes sent each day rarely differ or deviate from day to day. The second experiment is utilizing off-the-shelf IDSs, trying to detect any strange behavior in our dataset without any extra configuration, do not, as expected, show any indication at all, except for some false positives. In the final experiment we look at the process semantics of the system, namely the Modbus protocol. Here we can see from Modbus commands that the amount sent daily also showcase a static behavior, much like the result from experiment one. The final experiment focus on the register values, but we start by classifying the registers according to Hadžiosmanović et al. [4]. The result from the analysis of register values shows predictable or static behavior. It also shows that the amount of data included in analysis can have an impact on the results. For example, during the classification, our results showcase completely different behavior compared to the results presented by Hadžiosmanović et al. [4]. Also, we achieve different results when running the classification algorithm for a period of 1 day, compared to a period of 14 days.

From these three experiments we now move on to make informed and motivated decisions as to what the best methods will be to use when trying to secure an ICS using anomaly based IDSs.

7

Proposed methods

In this chapter we propose three different methods that can aid in finding intrusions in an ICS. The choice of these methods are based on the results from the four previous chapters. The survey in Chapter 3 provided various methods, assumptions and important parameters to think about when designing an IDS for ICSs. After that, in the Risk analysis in Chapter 4, we gain extensive knowledge about various risks, threat actors and signs of intrusions. This information comes from a paper as well as a workshop with an industry expert. Lastly, in Chapter 5 and Chapter 6, we present and perform experiments and analysis in order to see which methods from the survey show promise on our real world data. Also, what kind of signs of intrusions that we could be able to find in our dataset. After the proposed methods are explained and motivated, we also perform an evaluation of them. The evaluation is also further discussed in Chapter 8.

7.1 Methodology

As stated in Chapter 1, we present methods that can be used in an IDS in order to further increase security in an ICS. We setup to choose three methods based on the following:
Survey: The survey performed in Chapter 3 provides information about what has been done previously, which methods were used, the validation methodology and if other articles validate their methods on simulated or real data.
Risk analysis: The risk analysis presented in Chapter 4 gives us a deeper insight into what kind of risks that exist in ICSs. The risk analysis consists of an extensive analysis of standards published by NIST, as well as a workshop with a local expert with insight into ICS security and IT security overall. The risk analysis helps us to understand what kind of threats there are, threat actors and what to look for by means of signs of intrusion.
Experiments performed on our dataset: Various experiments explained in Chapter 5 are performed, and their results presented in Chapter 6. These experiments help us to validate assumptions and methods proposed by other authors in the survey, as well as to give us an understanding whether we can look for signs of

intrusions from the risk analysis. In this chapter we get further understanding as to what kind of methods we can propose for our dataset from a real world ICS.
Feasibility: Another important criteria when selecting methods from the survey, is that the papers and articles need to provide information about the algorithms and methods used. Articles that fail to provide well explained methods will not be considered as feasible methods.

7.2 Proposed methods

In this section we present the three methods that we find suitable when implementing an anomaly-based IDS in ICSs, based on the methodology that was presented above. These methods either work on a semantic based level or on a network based level, or both. Method 1, seen in Section 7.2.1, uses an autoregression algorithm to predict the next continuous value of a register, as well as control limits for constant registers, attribute registers and continuous registers, to check that they follow the actual process semantics. Method 2, presented in Section 7.2.2, analyses the network flow and adds flow records that keep track of properties such as IP-addresses, ports or time since last message. The final method in Section 7.2.3, combines two learning algorithms in order to train a neural network to cluster normal network traffic behavior.

The algorithms are explained in detail in this section, in order to give further understanding of the evaluation performed in Section 7.3, as well as the discussion and conclusion of this thesis.

7.2.1 Method 1: Autoregression with control limits

This method is based on the paper by Hadžiosmanović et al. [4]. Their approach is to watch the semantic behavior of an ICS, namely the register values of PLCs that are located within the ICS. They have divided their approach in a couple of steps; First of all, they extract data. Then they characterize the data, which we also call classification of registers. Lastly, they model the data and propose methods that can be used in order to detect deviations from normal behavior.

Data extraction: The first step they do is to parse and extract the Modbus commands from the network traffic. The data include the operation code such as a read command or update command with the value that will be updated. This is the key for it all to work and they developed a Modbus analyzer in order to achieve this. When they have extracted the data from Modbus traffic, they construct memory shadow maps. When these are constructed, they then proceed with the next step, and that is to characterize registers based on register values.

Data characterization: They characterize the data into three different categories: constant registers, attribute-based registers and continuous registers. How this is done is further explained in Section 5.1.3.2. Each category is handled separately by different methods in the next stage, Data Modelling and Detection.

Data Modelling and Detection: After the process of characterizing the three different types of registers they can easily monitor abnormalities in the process semantics of both constant registers and attribute registers. Constants need to be checked as to never change its value and from the learning phase make sure that an attribute register never assumes a value outside of its learned set of values. The final register, the continuous one, proves to be more difficult. In order to check the semantics of this type they use two different approaches: one autoregressive algorithm, and one that checks that the process never go out of its control limits. The autoregressive algorithm predicts a registers next value. In order to estimate deviations in the model they compare a residual variance, which is observed during the training phase, to a prediction error variance, which is observed during the testing phase. These variance variables are produced using a method called two variance hypothesis tests, or F-test. Should the prediction error variance be higher than the residual variance, then the values have deviated from the model and the algorithm will treat this as an anomaly. The autoregressive algorithm of order p is defined as follows:

$$x_i = \phi_0 + \phi_1 x_{i-1} + \phi_2 x_{i-2} + ... + \phi_p x_{i-p} + \epsilon_i$$

To estimate the coefficients, $\phi_1, ..., \phi_p$, used in the algorithm, they make use of Burg's method. Estimating the order of their model they use the common Akaike information criterion. ϵ_i is a normally distributed error term with zero mean and non zero variance σ^2. Using this method they are able to predict the next value of a register. The Figure 7.1 shows their predicted series using the above method and how they violated the series using false commands in their testbed environment.

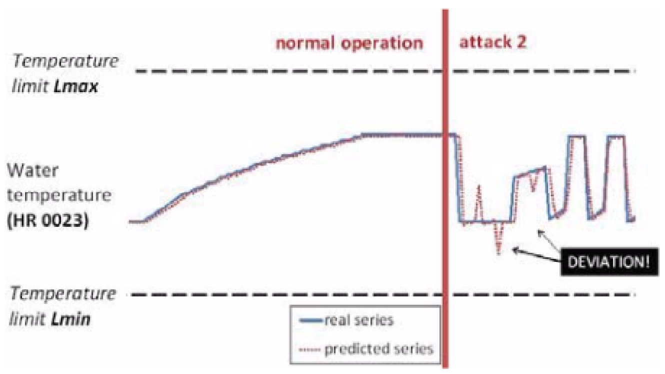


Figure 7.1: Example of violation of the autoregressive calculation, as shown in the paper by Hadžiosmanović et al. [4].

In order to calculate the control limits they use Shewhart control limits. This method uses a pair of values, one max value and one min value. Should the continuous value go beyond any one of these limits set up it would detect this as an anomaly, since the process is going outside of its operational limits. Figure 7.2 shows Hadžiosmanović et al. violating their set up control limits in their testbed environment.

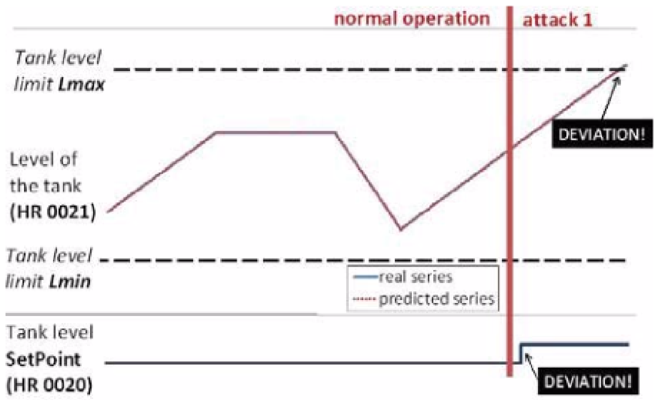


Figure 7.2: Example of violation of the control limits as well as a configuration change, as shown in the paper by Hadžiosmanović et al. [4].

7.2.2 Method 2: Flow-based anomaly detection

This method is based on a paper by Valdes and Cheung [27]. Their approach is to analyze patterns in a network that consists of Modbus over TCP/IP traffic. Their IDS focus mainly on network flow information, such as endpoints in the network, rate of packet flow between the endpoints and other parameters that can be useful in order to determine if traffic is anomalous. When analyzing network flows, they look for anomalies, such as new flows, flows that cease to exist and flows that change. They have two different approaches, Pattern-based anomaly detection and Flow-based anomaly detection, which are further explained below.

Pattern-based anomaly detection

Their pattern-based anomaly detection observes a datastream and examines patterns in it. They describe a pattern as a vector that contains source and destination IP address, and destination port. Since they deem the source port to be ephemeral, they only include the destination port in the pattern. When they observe patterns they evaluate them against a pattern library. If a pattern match an existing pattern in the library, they use a similarity function and similarity threshold in order to determine the best-matching pattern in the library. They claim that one of the key strengths of this approach is that the data is not required to be attack-free, since they use historical probabilities and similarities.

By looking through all patterns in the library, they choose the one that has the best match according to some similarity function, which is then matched against a match threshold. In their test runs they choose $T_{match} = 0.6$. If this returns a value greater than the threshold then this pattern is a winner. If not the pattern is inserted into the library of patterns. The formal algorithm can be observed below:

```
Algorithm to pick winner:
Find K s.t.
$Sim(X, E_K) \geq Sim(X, E_k) \forall k$
$X$ = observed pattern
$E_k$ = $k$th pattern exemplar in library
$If\ Sim(X, E_K) \geq T_{match}$, $E_K$ is the winner
$Else$ insert $X$ into the library of pattern exemplars
$T_{match}$ = Minimum match threshold
```

When a pattern has won they proceed to modify and update the observed pattern. The updated pattern will be a result of the amount of times the pattern has been observed times the pattern value added with the currently observed pattern. All of this is then divided with the amount of times E_K has been observed added with one. They call this the *Adaptive modification of the winning pattern*:

$$E_K \leftarrow \frac{1}{n_K + 1}(n_K E_K + X)$$

$$n_K = \text{Historical (possibly aged) count of observances of}\, E_K$$

For an anomaly to be detected they first determine the probability of the winning pattern by dividing the amount of occurrences of that pattern with all the occurrences in the pattern library. This is then used as a limit when calculating the *Tail probability*. The Tail probability is calculated by adding all the probabilities which is lower than the probability of all other patterns. The Tail probability is then checked against the alert threshold, in their experiment they used a value of $T_{alert} = 0.7$, if it is greater than this value an anomaly has been detected. Below is the formula used:

$$\Pr(E_K) = \text{Historical probability of pattern}\, K = \frac{n_K}{\sum_k n_k}$$

$$Tail\_\Pr(E_K) = \text{Historical tail probability of pattern}\, K = \sum_{\Pr(E_k) \geq \Pr(E_j)} \Pr(E_j)$$

$$If\, Tail\_\Pr(E_K) \leq T_{alert}, \text{ generate alert}$$

$$T_{alert} = \text{alert threshold}$$

Flow-based anomaly detection

Their flow-based anomaly detection approach is to keep a database of active and historical flow records. Flow records are generated or updated on the fly as packets are observed. When a flow record is updated by new traffic, it is also evaluated against previously experienced behavior from historical flow records. They also implement a global update which transform all flow records since the last global update into historical statistical profiles. The idea is to detect anomalies such as new flows, changes in rate of data transmitted and absence of an existing flow. As stated earlier, there are two types of records, further explained below.

A flow record contain the following elements: Flow Record = {Source IP and port, Destination IP and port, Time since last packet, Number of packets since global update, Average number of bytes per packet since global update, Variance of bytes per packet, Average interval between packets since last global update, Variance of interval between packets since last update, Anomaly score for the record}.

The Historical record contains the following elements: Historical Record = {Source IP and port, Destination IP and port, Time of last global update, Historical number of packets, Historical average number of bytes per packet, Historical variance of bytes per packet, Historical average interval between packets, Historical variance of interval between packets}.

As stated earlier, Valdes and Cheung look for flows that are new, flows that disappear and changes in the flow. The algorithm will generate an alert in case a new flow record were to appear when there is no corresponding historical record or when so many flows have been observed that a new one would be highly unlikely. The anomaly score is used when looking for deviations and changes in flow. The scoring uses the bytes per packet and inter-arrival time of a flow record relative to the historical record, based on the formula below where T is a test:

$$T_X = \frac{Avg(X) - HAvg(X)}{\sqrt{\frac{Var(X)}{Packets}}}$$

The anomaly score for a flow is obtained using the formula below, if bytes per packet and the inter-arrival time *(DT)* between packets is treated independently.

$$Score = \sqrt{{T_{NumBytes}}^2 + {T_{DT}}^2}$$

This scoring algorithm is used to find anomalies if the anomaly score is greater than a given threshold. As mentioned before they also raise alerts on new flow records and also if there is a missing flow record for a historical record after a global update period.

The authors also introduce an aging concept to the algorithm, intended to allow it to adapt to changing environments. So after each global update the historical records are updated with the corresponding flow records plus the aging and the flow records are reset.

To test the flow-based anomaly detection approach, the authors conduct three different experiments: increasing the flow rate of a Modbus connection, reducing the flow rate of a Modbus connection and finally they use a Modbus-TCP scanner. In all three experiments the algorithm was successful in detecting the changes in the network.

7.2.3 Method 3: IDS-NNM algorithm

The method described in the paper written by Linda et al. [31], uses neural networks to cluster normal behavior and to detect anomalies. They combine two neural network learning algorithms; The Error Back-Propagation and the Levenberg-Marquardt algorithm. These two algorithms are used to model the boundaries of the clusters representing normal network traffic behavior when training the artificial neural network. These algorithms are not analyzed in-depth by us in this thesis.

The authors mainly focus on so called windows of packets of a certain length. In their experiments they chose the length of the window to be 20 packets long. From these 20 packets currently inside the window, a feature vector, $\overrightarrow{r_j}$, is calculated. During one window, they extract certain attributes, which they later use to classify traffic as either anomalous or normal. The most significant window based attributes that they look for are the following: number of IP-addresses, number of packets with window size of 0, average interval between packets, number of packets with data length of 0, number of protocols used, number of flag codes used, average window size and average data length.

According to the results from the different architectural setups that were tested, it was concluded that the optimal setup was to use a two-layer feed forward neural network with 10 neurons in the first layer, 6 neurons in the second layer and one output neuron. The proposed algorithm consists of two phases used in order to train the network; the first one is to construct a training set to be used in the network and the second one is the supervised training session of the network using the constructed training set.

The training set is constructed by extracting a sequence of the window based feature vectors $\overrightarrow{r_j}$ using the attributes mentioned earlier. Each feature vector is assigned a normal class label. Next is to construct a randomly generated set of intrusion vectors uniformly distributed over the windows based attribute space and labeling each feature vector with an intrusion class label. Lastly, they combine each of these two sets into a single training set, T, used in the second phase.

The training set, T, is used as an input vector for the neural network. The following algorithm is used to calculate the net input of the ith neuron in layer $k+1$:

$$n^{k+1}(i) = \sum_{j=1}^{Sk} w^{k+1}(i,j)a^{k}(j) + b^{k+1}(i)$$

Sk, which is the amount of neurons used in the current layer, $w^{k+1}(i,j)$, is the weight of the connection between neuron j and this neuron in the current layer. The bias of neuron i is $b^{k+1}(i)$ and the output from neuron j in the current layer is $a^{k}(j)$. The value from $n^{k+1}(i)$ is then used as input for the activation function. This is the output value from the ith neuron:

$$a^{k+1}(i) = f^{k+1}(n^{k+1}(i))$$

The Levenberg-Marquardt algorithm's task is to minimize the total error of the neural network. P is the amount of patterns, M is the amount of outputs in the algorithm and d_{pm} is the desired output:

$$E = \sum_{p=1}^{P} \sum_{m=1}^{M} (d_{pm} - a_{pm}^{L})^2$$

A Jacobian matrix is computed according to the modified Error Back-Propagation algorithm, combining this with the Levenberg-Marquardt algorithm fixes ill-defined Jacobian matrices by introducing an identity matrix I and a learning parameter μ. μ is initialized to be 0.001 and should the total error increase from the above algorithm then μ is multiplied by 10, should the total error decrease then μ is divided by 10 instead. By solving the following weight update vector they are able to update the network weights and the learning parameter μ, where $\overrightarrow{e}$ is an error vector:

$$\Delta\overrightarrow{w} = [J^{T}J + \mu I]^{-1}J^{T}\overrightarrow{e}$$

This training algorithm is done in order to minimize the classification error by eventually finding the boundary of the normal behavior class. If anything happens to be located outside of this boundary then it is considered an intrusion.

The authors generated intrusions using the tools nmap, Nessus and MetaSploit and combined this with their testbed data. Using the architecture with two layers with 10 and 6 neurons respectively they were able to identify 100% the intrusion attempts created in the sample data.

7.3 Evaluation

In this section we present results from the evaluation of methods described in Section 7.2.1, Section 7.2.2 and Section 7.2.3. Each section also explains how the evaluation is performed. Method 1 and Method 2 are mostly implemented as described in the papers. Since some details are left out in the papers, these sections only provide proof of concept, rather than exact results. Method 3 is not implemented at all because of its complexity and that details are left out of the paper, making it difficult to implement it.

The results from this evaluation is further discussed in Chapter 8 and later concluded in Chapter 11.

7.3.1 Method 1: Autoregression with control limits

We have implemented most parts of the algorithm presented by Hadžiosmanović et al. [4]. Since several parts of the implementation is not described in the paper, especially when it comes to the autoregression algorithm, we choose to only implement the functionality for constant and attribute based registers. Constant registers may never change, and attribute registers may only switch between a certain set of values. We present the analysis with different learning and analysis periods, as seen in Table 7.1. First, we perform an analysis using a learning period of 3 days, and an analysis period of 3 days as well, which is the same approach as the authors of the algorithm. Secondly, we perform analysis using a learning period of 3 days, but a analysis period of 1 day. Lastly, we extend the learning period to 7 days and perform analysis on 3 days.

Table 7.1: Result from evaluation of Method 1: Autoregression with control limits

	3 day learning 1 day analysis	3 day learning 3 day analysis	7 day learning 3 day analysis	10 day learning 3 day analysis
Occurrences of constants	0	6	106	1273
Constant mismatch (#)	0	5	82	1200
Constant mismatch (%)	0 %	83 %	77 %	94 %
Occurrences of attributes	1364342	4436827	3455747	3030904
Attribute mismatch (#)	1048	593770	110834	4700
Attribute mismatch (%)	0.07 %	13 %	3.2 %	0.15 %

As seen in Table 7.1, the result clearly vary between different approaches, both when it comes to learning period and analysis period. When using a learning period of 3 days and an analysis period of 3 days as well, the algorithm yields a large percentage of mismatches considering both constant and attribute registers. In the second analysis with a learning period of 3 days and an analysis of 1 day, the algorithm does not yield a single constant mismatch since there are no occurrences of constant in that period, and a very low amount of attribute mismatches as well. In the third analysis, with a learning period of 7 days and an analysis period of 3 days, the algorithm yields far better results than the first analysis, especially when considering attributes. This is due to that there are many registers that does not classify as continuous when using a learning period of 3 days, but do classify as continuous when using a learning period of 7 days, since more values are observed.

7.3.2 Method 2: Flow-based anomaly detection

In the article by Valdes and Cheung [27], they describe their algorithm Flow-based anomaly detection. We implemented a simple system trying to replicate the method in the article as closely as possible.

The system reads a network capture file and converts it into flow records according to the authors methods. After each period a global update is issued. In our implementation we choose the period length to be 180 seconds long. When a global update is performed all flow records updates their corresponding historical records or creates one if no one exists for it yet.

When a historical record exists for a flow record an anomaly score can be calculated. If this score exceeds a given threshold an anomaly is reported. However, picking a suitable threshold is troublesome. There are records that have a score as low as around 0.05 while others scores around 0.7, making it hard to pick something that detects anomalies for all flows.

In Table 7.2 examples of anomaly scores on two flow records from our dataset can be seen. The anomaly scores in the table accounts for 1 hour of running the algorithm. In Figure 7.3, the anomaly scores for the same two flow records can be seen, but visualized in a graph, and for a duration of 12 hours instead.

Table 7.2: Examples of two flow records from 1 hour of analysis.

Global Update Time (seconds)	Anomaly Score
540	0.183864584290828
900	0.4730531934381516
1440	1.1238482003913244
1980	1.1461997076188528
2520	0.8619689062598697
3420	0.7439147654465018
540	0.20244827206402996
900	0.1368861120224177
1440	0.21961507274170547
1980	0.04703492147976729

2520	0.044110492017037956
3420	0.069266703332644

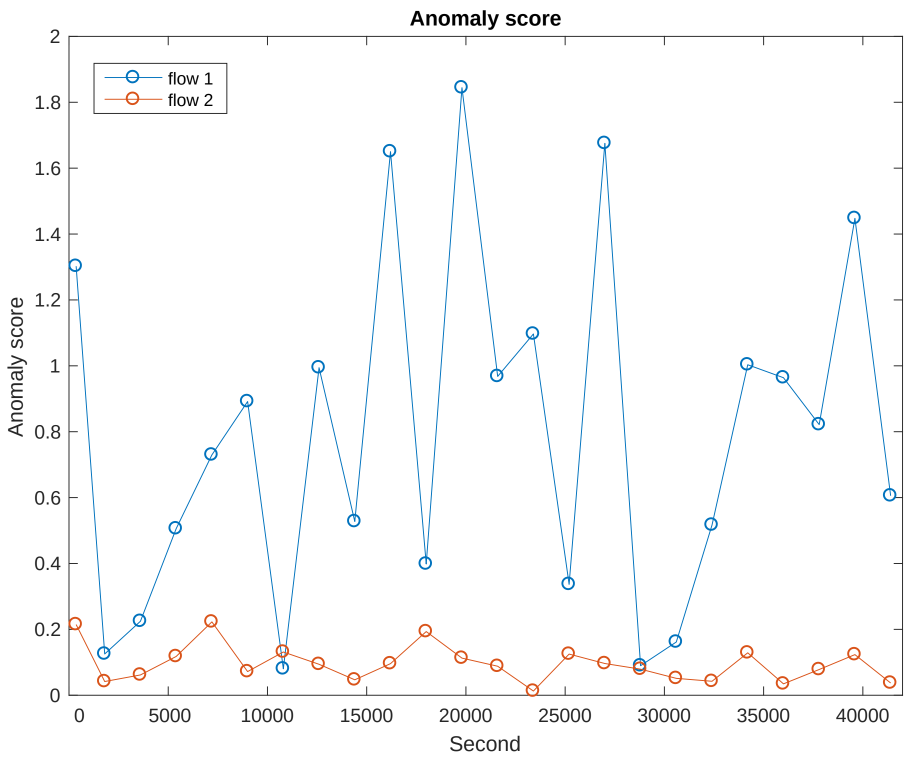


Figure 7.3: Examples of two flow records from 12 hours of analysis.

Another property to be accounted for is that a few flow records' anomaly score fluctuates a bit from the start to finally reach a stable score. As Table 7.2 shows, one flow record switches scores quite greatly while the other stays stable though out the hour. This is something that also can be seen the graph from Figure 7.3. The algorithm benefits from running and learning a bit before going online to make it less prone to erroneous label flows as anomalies.

7.3.3 Method 3: IDS-NNM algorithm

This section present Method 3, but because difficulties implementing the algorithm due to the lack of repeatability and lack of implementation details in the authors article, we only present an analysis of the method with reasoning. The method is, as stated in Section 7.2.3, based on neural networks. Table 7.3 shows the network attributes that the algorithm perform analysis on in order to detect anomalies in behavior.

Table 7.3: 8 window based attributes that are extracted and used.

Number of IP addresses	Number of packets with window size 0
Average interval between packets	Number of packets with 0 data length
Number of protocols	Average window size
Number of flag codes	Average data length

Much like Method 2, this algorithm looks at certain network attributes. However, unlike Method 2, this algorithm does not look at individual conversations but checks a group of packets at a time. Each window consists of a certain amount of packets, with their extracted properties, as input for the neural network.

As stated earlier, we do not implement and test this algorithm. However, it does show promise. Having a window based extraction technique looking at the properties from Table 7.3, or at least similar properties could work well given our dataset, given that the results from the experiments in Section 6.1, shows that the network traffic is static. Since their data is similar to our dataset and static, at least to the best of our knowledge we deem it applicable to our dataset as well. Their tests also yield a very good result, in the best case 100 % detection rate and 0 % false positives. Therefore, we assume that the algorithm would also perform well for our data.

7.4 Summary

In this chapter we present, explain and evaluate three different methods found during our survey, presented in Chapter 3. These methods are chosen based on the results of the survey and risk analysis, as well as the outcome of the experiments performed. These methods are supposed to aid in finding intrusions in an ICS.

We have chosen to partly implement two of these methods. The level of implementation depends on how detailed the methods and algorithms are explained by the authors, which means that the implementations should be seen as proof of concepts builds, rather than an implementation that yields exactly the same result as the authors. However, these results should provide some guidance at least, showing what kind of algorithms that would prove useful for ICS.

The evaluation is further discussed in Chapter 8, and concluded in Chapter 11.

8 Discussion

This chapter brings up discussions about each part of the report. We include discussions from Chapter 3, Chapter 4 and Chapter 6 even though we already discuss these chapters individually. However, repeating this and including further reasoning is useful in order to understand the discussion about the last chapter, Chapter 7, and an overall discussion of the entire thesis.

This chapter is divided as follows: Section 8.1 contains a short wrap up of results from the Survey, as well as some discussion about work performed in the same area before. Section 8.2 presents the Risk analysis in short, as well as the discussion from that chapter. Section 8.3 discusses the various experiments explained in Chapter 5, and their results presented in Chapter 6. Finally, in Section 8.4, we conclude the discussion with reasoning about the final chapter, Chapter 7.

This chapter aims to discuss all findings in the report, before presenting a discussion about ethics and sustainability in Chapter 9, as well as future work described in Chapter 10, and finally concluding the thesis in Chapter 11.

8.1 Survey

The performed survey, seen in Chapter 3 provides two different results, one quantitative and one qualitative. The quantitative result does not bring much to the discussion since it is more of an introduction to the papers selected than an actual analysis. However, the qualitative analysis is worthwhile discussing. In Table 3.5 we summarize the qualitative analysis.

We see that most authors indeed have data that originates from real systems (A.1). Despite this, most of the authors do not provide any information of how much data that they have. The amount of data that is subject to analysis can be seen as an important parameter to examine and compare, since systems might behave differently during the weekdays, during different weeks and during different times of the year.

We can also see that the level of analysis (A.2) differs between the papers. One article focus solely on process semantics, three articles focus on network analysis and the last three articles perform analysis on both process semantics and network traffic. This is indeed a feature, since the focus of this thesis is on both of these fields.

When it comes to the algorithm (A.3), we can observe a difficulty when trying to re-implement the authors' approaches. Only two authors provide mathematical formulas, and only one author provide both sample code and mathematical formulas. Thus, it is rather difficult to repeat their approach on a completely new dataset. As seen in Chapter 7 where we propose methods for an IDS, the lack of information about algorithms and mathematical formulas complicates the evaluation step.

Level of the data (A.4) and capture position (A.5) are generally included in the articles. However, test validation method (A.6) is not well motivated in some cases. Some authors consort to simple network probing tools, such as nmap scans. Generally, these network probing tools generate a large amount of traffic over a small period of time, which would indeed trigger an alarm, in a method that performs analysis on network parameters such as network flows. However, we deem this as an unfair validation since it does not provide any result of how good the method would detect more sophisticated attacks.

8.2 Risk analysis

In the risk analysis performed in Chapter 4, we investigate what threats and risks there are to ICSs according to NIST, as well as according to an industry expert. The published work from NIST acts as a set of guidelines to consider when securing an ICS from threats and risks. The workshop with the industry expert resulted in a different kind of insight as to what the threats that face ICSs are.

Some of our initial thoughts about the risks to ICSs were from hacker groups or government actors because these systems often control the infrastructure and is therefore a suitable target to attack in times of conflict. These thoughts were confirmed by the work from NIST with the threat actors and also by the industry experts opinion. NIST also provided some insight as to what signs of intrusions could look like, something that is taken into consideration when performing experiments on our dataset.

8.3 Experiments and results

The experiments performed, explained in Chapter 5, with results in Chapter 6, answers the questions that they were supposed to. How much of the data is regular and automatically generated? How much of the data is generated by human interaction? Can we categorize between different types of events, e.g. time-triggered, user-triggered, event-triggered? Can we characterize device interaction patterns based on a data-driven approach?

Our initial hypothesis was that the data would be very regular and automatically generated without human interaction. This was proven to be true when examining our dataset. We also saw that the data could be categorized into categories.

The experiments also show different findings compared to some other authors. For example, our classification of registers, as seen in Table 6.1, showed that most of the registers in our dataset are continuous, while a majority was constant in the article by Hadžiosmanović et al. [4]. Thus, an approach that neglect the importance of accurately identifying deviations in continuous registers would be nearly useless when used on our dataset.

We also see that two off-the-shelf IDSs, Bro and Snort, does not trigger on any data in our dataset. This is probably due to that we do not specify any additional configuration parameters, but we can not know for sure since we only can assume that our dataset does not contain any direct or indirect control attacks on the process semantics. Hadžiosmanović et al. [4] have configured Bro in order to check for register values as well, but their code is not public. Thus, it is hard for us to test a similar solution with Bro.

8.4 Proposed methods

When evaluating the methods proposed in Section 7.2, trying to replicate the methods as closely as possible given the information from the articles we found it quite hard to get it exactly correct. Everything was not clear as crystal and certain values for example threshold limits were not declared in the clear, which complicated the process of repeating the authors' approach.

When trying to evaluate the first method proposed by Hadžiosmanović et al. it quickly became apparent that the amount of data is a crucial part of the method. The register classifications varied quite a bit, so learning the algorithm required a large dataset for our classification to be as true as possible to the real world. Hadžiosmanović et al. only used 3 days of data to teach their algorithm, which in our case yields many mismatches, mostly due to registers that are classified as attribute-based, when they truly are continuous. This shows the true importance of using a large dataset when performing analysis, mostly due to that the algorithms have to learn the correct behavior of an ICS.

For the second method by Valdes and Cheung, we replicated and evaluated their Flow-based anomaly detection scheme. As it turned out it was quite easy to follow what they had done at first but when getting down to the details it got quite confusing. How often and when the testing and anomaly scoring was done for example? They mention scoring is not possible without any historical records for a certain flow record. Did this mean the scoring would be used every time a flow record was updated when a historical record existed for the flow record or as they also mention, when the historical packet count is large? The big feature for this detection scheme was the anomalous flow alert which was based on the anomaly score. If the anomaly score exceeded some threshold then an alert would be raised, however, they seem to fail to mention how this threshold should be chosen. Should the threshold be chosen after a couple of global update periods as to confidently pick a low enough threshold?

This threshold plays a big role for this scheme to work as intended, flow records differed from record to record when it came to the anomaly score. Some scored very low (>0.05) while some scored high (~ 1.1) making a decision for a fair threshold a challenge in itself. Perhaps having an individual anomaly threshold on a per record basis could mitigate the challenge. However, this method showed promise when it came to having analyzing intrusions on a network basis.

When comparing the learning time for Method 1 and Method 2 respectively, we observe that Method 2 has a much shorter learning time before it can perform reasonable well. Method 1 required a learning time of 7 days before becoming stable, as seen in Table 7.1, while Method 2 exhibits a very similar pattern after just one hour, as seen in Table 7.2. Figure 7.3 also exhibits this stable behavior over the time period of 12 hours.

For the last method, using neural networks by Linda et al., we only evaluate it based on the results and methods from their article and draw conclusions as to how well it would suit our own dataset. They provide well explained mathematical models but do not explain any implementation specific details. This is why we could not provide any proof of concept system using the particular method. When it comes to the supervised training of the neural network they state that it needs to be trained with both regular traffic and traffic containing intrusion signatures in order for it to learn its cluster boundaries. For intrusion vectors they fail to mention details as to what the intrusions are only that they use tools like nmap, Nessus and MetaSploit, which is something we have discussed earlier in that these attacks may not be all that representative when it comes to detecting intrusions.

9

Ethics and sustainable development

In this chapter we gather ethical questions that arose during the survey presented in Chapter 3, and the risk analysis presented in Chapter 4, as well as questions that arose when examining the dataset given to us in Chapter 5 and Chapter 6.

We have chosen to divide this chapter into two key sections, Sustainability and Surveillance. In the section about sustainability we speak about how the topic of security in ICSs is important for a sustainable society, since ICSs often control systems of high importance. In the section about surveillance we reason if the dataset, together with analysis, can intrude on people's privacy.

9.1 Sustainability

Security in ICSs, especially those that handle critical functions to society, is a very important topic. It would not be sustainable for society if these system go down for a longer period of time, since this probably would cause harm to the society. Since methods that look for malicious activity by watching process variables would detect both security breaches, and other incidents that might alter the functionality of the system, we deem that the area of IDSs in ICSs is important by means of sustainability.

9.2 Surveillance

We have been given a large dataset from a water distribution network in a large municipality in Sweden. This data contains a lot of information that can reveal information about water consumption. However, it is hard to make something useful out of this information, since the only way to identify areas is by pairing the IP address of a substation together with geographical information, which we do not have.

Also, each substation distribute water to large areas, so it is deemed to be impossible to actually connect the information to specific households or customers.

10

Future work

This chapter brings up potential future work that can be performed in the area of IDSs in ICSs, based on the work that we present. The future work is based on problems and potential improvements that can be done based on our work.

Generate believable and accurate network traffic: Since it is quite hard to acquire real world data from actual ICSs, one approach could be to try to generate this in a believable and accurate fashion. Simple testbeds that do not resemble real systems can provide false results. Therefore, it would be good to have some kind of framework that can generate network data based on a certain set of parameters and configurations.

Create a virtual ICS: In order to find more realistic attack vectors, a virtual ICS environment can be created, which could be used to replay captured data. This could create an environment that can be used as a honeypot ICS, which possibly would yield realistic attack vectors if actual threat actors try to attack it. A simple honeypot might provide attack vectors of some degree, but a system that is being run using real world data could provide more fair and true attack vectors. This could also give a hint of what kind of consequences there could be based on an actual attack.

Packet injection framework for ICSs: An extensive framework that can be utilized in order to inject packets into datasets would be useful when testing different methods and measuring their effectiveness. This could provide more suitable and realistic attack vectors, unlike running simple nmap tests.

Test methods on multiple datasets: It would most likely be beneficial and yield a greater understanding of how different kind of ICSs differ from each other, and how methods can be applied to multiple types of ICSs. Therefore, it would be good to test methods on multiple datasets from multiple sources, e.g. another water distribution system or similar. Those datasets could be used to test methods for a more accurate and generic result.

Fully implement an IDS: Because of the scope of this work, we do not implement all proposed methods fully. Proof of concept builds are made in order to perform simple tests in order to evaluate efficiency of the methods. However, it would be useful to fully implement all the proposed methods and actually implement it as a real time IDS that watch network traffic.

11

Conclusion

In this thesis we set out to conduct analysis on network traffic from an ICS in order to gain a greater understanding as to what patterns could be observed, in order to make motivated proposals of anomaly detection methods. One of our key benefits is that we were lucky to have been given access to a large amount of data originating from a real world ICS. Given this data, and the understanding we gained from our network and semantic analysis, we research data-driven methods for discovering anomalies in the ICS. These were methods that we proposed in Chapter 7, given their potential to detect anomalies based on our dataset.

In order to understand the field of IDSs in ICSs and get inspiration of anomaly-based methods that could prove useful, we present a survey. Also, another important factor when developing IDSs is to understand the threats and risks that a system is exposed to. Therefore, we conduct a risk analysis using a paper and hold a workshop with an industry expert. With the understanding from the survey and the risk analysis, we perform experiments on our dataset in order to get a further understanding of how the data looks like and what kind of methods that could be useful. Lastly, we perform evaluation of three methods; One method that perform analysis on process semantics in order to see that the actual industrial process proceeds undisturbed. One method that perform analysis on network flows in order to search for anomalies, such as new flows, changes to existing flows or flows that disappear. One method that take advantage of neural networks in order to also find anomalies in network traffic.

One conclusion we can draw from both the experiments and when evaluating the three methods, is that the amount of data available for both learning and for analysis is a crucial parameter. In our case, more data has generally led to more accurate results. We also see that we receive other results when classifying registers compared to other authors, which means that there are differences between different ICSs. Therefore, we think that it is important to have access to a large amount of data when teaching the data driven methods when employing them in an IDS.

Another important thing is to conduct fair tests. By fair, we mean that the tests should be at least inspired by potential real world attack scenarios. An extensive risk analysis aids the understanding of such threats and signs of intrusion. Flooding the network using simple network tools have been used in previous related work, but this could be circumvented by e.g. throttling the speed of the analysis.

Our biggest contribution is that we provide detailed information about our methods of choice, as well as our dataset. This could in hand contribute to future efforts in the same field. Based on previous work, threats and sign of intrusions, and since our experiments on the dataset confirms that the network traffic is static, we deem that anomaly-based methods are well suited for detection intrusions in ICSs. We also deem it very important to perform analysis on both process semantics and pure network analysis, since they might reveal different signs of intrusion.

Bibliography

[1] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to industrial control systems (ICS) security," *NIST Special Publication 800-82*, May 2015.

[2] MODICON, Inc., Industrial Automation Systems, "Modicon modbus protocol reference guide." [Online]. Available: http://modbus.org/docs/PI_MBUS_300.pdf

[3] K. A. Scarfone and P. M. Mell, "Sp 800-94. guide to intrusion detection and prevention systems (idps)," National Institute of Standards & Technology, Gaithersburg, MD, United States, Tech. Rep., 2007.

[4] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *Proceedings of the 30th Annual Computer Security Applications Conference*, ser. ACSAC '14. New York, NY, USA: ACM, 2014, pp. 126–135. [Online]. Available: http://doi.acm.org/10.1145/2664243.2664277

[5] M. Mantere, M. Sailio, and S. Noponen, "A module for anomaly detection in ics networks," in *Proceedings of the 3rd International Conference on High Confidence Networked Systems*, ser. HiCoNS '14. New York, NY, USA: ACM, 2014, pp. 49–56. [Online]. Available: http://doi.acm.org/10.1145/2566468.2566478

[6] D. Hadžiosmanović, D. Bolzoni, S. Etalle, and P. Hartel, "Challenges and opportunities in securing industrial control systems," in *2012 Complexity in Engineering (COMPENG). Proceedings*, June 2012, pp. 1–6.

[7] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May 2011.

[8] Wired, "Everything we know about ukraine's power plant hack," 2016. [Online]. Available: https://www.wired.com/2016/01/everything-we-know-about-ukraines-power-plant-hack/

[9] PowerMag, "What you need to know (and don't) about the aurora vulnerability," 2013. [Online]. Available: http://www.powermag.com/what-you-need-to-know-and-dont-about-the-aurora-vulnerability/?pagenum=1

[10] Wired, "A cyberattack has caused confirmed physical damage for the second time ever," 2015. [Online]. Available: https://www.wired.com/2015/01/german-steel-mill-hack-destruction/

[11] W. Bolton, *Programmable Logic Controllers.* Elsevier Science, 2015. [Online]. Available: https://books.google.se/books?id=sDqnBQAAQBAJ

[12] AMCI: Advanced Micro Controls Inc., "What is a PLC?" 2017. [On-

line]. Available: https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/what-plc/

[13] Modbus Organization, “Modbus protocol,” 2017. [Online]. Available: http://www.modbus.org/specs.php

[14] E. Alperovits and U. Shamir, “Design of optimal water distribution systems,” *Water Resources Research*, vol. 13, no. 6, pp. 885–900, 1977. [Online]. Available: http://dx.doi.org/10.1029/WR013i006p00885

[15] C. Rowland, “Intrusion detection system,” Jun. 11 2002, US Patent 6,405,318. [Online]. Available: https://www.google.com/patents/US6405318

[16] H. Debar, M. Dacier, and A. Wespi, “A revised taxonomy for intrusion-detection systems,” *Annales Des Télécommunications*, vol. 55, no. 7, pp. 361–378, 2000. [Online]. Available: http://dx.doi.org/10.1007/BF02994844

[17] Bro, “The bro network security monitor,” 2016. [Online]. Available: https://www.bro.org/

[18] M. Roesch, “Snort - lightweight intrusion detection for networks,” in *Proceedings of the 13th USENIX Conference on System Administration*, ser. LISA ’99. Berkeley, CA, USA: USENIX Association, 1999, pp. 229–238. [Online]. Available: http://dl.acm.org/citation.cfm?id=1039834.1039864

[19] Snort, “Snort - network intrusion detection & prevention system,” 2017. [Online]. Available: https://www.snort.org/

[20] R. Sommer and V. Paxson, “Outside the closed world: On using machine learning for network intrusion detection,” in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP ’10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 305–316. [Online]. Available: http://dx.doi.org/10.1109/SP.2010.25

[21] Wireshark, “Wireshark - go deep.” 2017. [Online]. Available: https://www.wireshark.org/

[22] ——, “tshark - the wireshark network analyzer 2.0.0,” 2017. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html

[23] S. Ostermann, “Tcptrace manual,” 2017. [Online]. Available: http://www.tcptrace.org/

[24] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, “Anomaly-based network intrusion detection: Techniques, systems and challenges,” *Computers & Security*, vol. 28, no. 1–2, pp. 18 – 28, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167404808000692

[25] A. Valdes, R. Macwan, and M. Backes, “Anomaly detection in electrical substation circuits via unsupervised machine learning,” in *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, July 2016, pp. 500–505.

[26] H. Lin, A. Slagell, Z. Kalbarczyk, P. Sauer, and R. Iyer, “Runtime semantic security analysis to detect and mitigate control-related attacks in power grids,” *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2016.

[27] A. Valdes and S. Cheung, “Communication pattern anomaly detection in process control systems,” in *2009 IEEE Conference on Technologies for Homeland Security*, May 2009, pp. 22–29.

[28] RhysU, “Autoregressive process modeling tools in header-only c++,” 2017. [Online]. Available: https://github.com/RhysU/ar

[29] T. Onoda and M. Kiuchi, *Analysis of Intrusion Detection in Control System Communication Based on Outlier Detection with One-Class Classifiers.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 275–282. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-34500-5_33

[30] M. Mantere, M. Sailio, and S. Noponen, “Network traffic features for anomaly detection in specific industrial control system network,” *Future Internet*, vol. 5, no. 4, pp. 460–473, 2013. [Online]. Available: http://www.mdpi.com/1999-5903/5/4/460

[31] O. Linda, T. Vollmer, and M. Manic, “Neural network based intrusion detection system for critical infrastructures,” in *2009 International Joint Conference on Neural Networks*, June 2009, pp. 1827–1834.

[32] P. Simões, T. Cruz, J. Gomes, and E. Monteiro, “On the use of honeypots for detecting cyber attacks on industrial control networks,” in *Proc. 12th Eur. Conf. Inform. Warfare Secur. ECIW 2013*, 2013.

[33] Modbus, “Modbus application protocol specification v1.1b3,” 2012. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf