TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technology

Rudolfs Kelle IVCM 214148

FURTHERING INDUSTRIAL CONTROL SYSTEM INTRUSION DETECTION SYSTEMS' ADVANCEMENT BY ANALYZING THE IEC 60870-5-104 PROTOCOL & ITS TRAFFIC

Master's Thesis

Supervisor: Bernhards Blumbergs
Dr.comp.sc. (cyber-security)

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Rudolfs Kelle IVCM 214148

TÖÖSTUSLIKU TEHNOJUHTIMISSÜSTEEMI SISSETUNGITUVASTUSE SÜSTEEMI EDENDAMINE IEC 60870-5-104 PROTOKOLLI JA SELLE LIIKLUSE ANALÜÜSIMISEL

Magistritöö

Juhendaja: Bernhards Blumbergs
Dr.comp.sc. (cyber-security)

Tallinn 2023

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Rudolfs Kelle

15.05.2023

Abstract

The IEC 60870-5-104 (IEC 104) protocol is widely used in Industrial and Automation Control Systems (IACS) for communication between various devices such as Remote Terminal Units (RTUs), Supervisory Control and Data Acquisition (SCADA) systems, and Programmable Logic controllers (PLCs). The IEC 104 protocol was looked at from a cyber-security perspective exploring ways to secure Operational Technology (OT) networks that use it by developing a Network Intrusion Detection System (NIDS) prototype. That began with an overview of IACS and their importance in critical infrastructure as well as risks associated with their compromise. Important cyber-security concepts were introduced in context with IACS and the IEC 104 protocol. Relevant research was examined regarding different aspects of IACS security, such as Intrusion Detection System (IDS), NIDS, and IEC 104 role in such systems. Additionally, the developed NIDS prototype was examined by exploring all topics related to its development including the validation of the developed NIDS prototype, which was accomplished by experimentation in the IACS laboratory located at CERT.LV [1] headquarters in Riga, Latvia. Finally, the results that were gathered during the validation phase were presented and discussed. In conclusion, this research provides an overview and analysis of the developed NIDS prototype. It serves as a valuable resource for researchers, engineers, and security experts who are interested in extending the security capabilities of IACS systems.

The thesis is written in English and is 57 pages long, including 6 chapters, 18 figures, and 2 tables.

Annotatsioon

Tööstusliku tehnojuhtimissüsteemi sissetungituvastuse süsteemi edendamine IEC 60870-5-104 protokolli ja selle liikluse analüüsimisel

IEC 60870-5-104 (IEC 104) protokolli kasutatakse laialdaselt tööstus- ja automaatikajuhtimissüsteemides (IACS) erinevate seadmete, näiteks kaugterminalseadmete (RTU), järelevalvekontrolli ja andmehõive (SCADA) süsteemide ja programmeeritavate Loogikakontrollerite (PLC) vaheliseks suhtluseks. IEC 104 protokolli vaadeldi küberturvalisuse vaatenurgast, uurides võimalusi seda kasutavate operatiivtehnoloogia (OT) võrkude kaitsmiseks, töötades välja võrgu sissetungimise tuvastamise süsteemi (NIDS) prototüübi. Käesolev töö algab ülevaatega IACS-ist ja selle tähtsusest kriitilises infrastruktuuris ning nende kompromiteerimisega seotud riskidest. IACSi ja IEC 104 protokolli kontekstis tutvustati olulisi küberturvalisuse kontseptsioone. Analüüsiti asjakohaseid uuringuid seoses IACS-i turvalisuse erinevate aspektidega, nagu sissetungimise tuvastamise süsteem (IDS), NIDS ja IEC 104 roll sellistes süsteemides. Lisaks uuriti väljatöötatud NIDS-i prototüüpi, käsitledes kõiki selle arendamisega seotud teemasid, sealhulgas väljatöötatud NIDS-i prototüübi valideerimist, mis viidi läbi eksperimenteerimise teel Lätis, Riias CERT.LV [1] peakorteris asuvas IACS-i laboris. Lõpuks esitati ja arutati valideerimisetapi käigus kogutud tulemusi. Kokkuvõttes annab käesolev uurimus ülevaate ja analüüsi väljatöötatud NIDS-i prototüübist. See on väärtuslik ressurss teadlastele, inseneridele ja turbeekspertidele, kes on huvitatud IACS-süsteemide turbevõimaluste laiendamisest.

Lõputöö on kirjutatud inglise keeles ja on 57 lehekülge pikk, sisaldab 6 peatükki, 18 joonist ja 2 tabelit.

List of Acronyms

APCI	Application Protocol Control Information
APDU	Application Protocol Data Unit
ASDU	Application Service Data Unit
CERT.LV	Information Technology Security Incident Response Institution of the Republic of Latvia
DC	Double Command
DMZ	Demilitarized Zone
DNP3	Distributed Network Protocol 3
DNS	Domain Name System
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
I-format	Information transfer format
IACS	Industrial and Automation Control Systems
ICS	Industrial Control System
IDS	Intrusion Detection System
IEC 101	IEC 60870-5-101
IEC 104	IEC 60870-5-104
IOA	Information Object Address
IT	Information Technology
ITACA	Internet Traffic and Content Analyzer
LAN	Local Area Network
MiTM	Man-in-the-middle
NIDS	Network Intrusion Detection System
OSI	Open System Interconnection
OT	Operational Technology

PLC	Programmable Logic controller
POC	Proof of concept
RTU	Remote Terminal Unit
S-format	Numbered supervisory function
SC	Single Command
SCADA	Supervisory Control and Data Acquisition
SIEM	Security Information and Event Management
TCP	Transmition Control Protocol
U-format	Unnumbered control function

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 Motivation

Industrial Control System (ICS), specifically the Industrial and Automation Control Systems (IACS), play a crucial role in the operation of critical infrastructure sectors such as energy, water, transportation, and manufacturing. These systems are responsible for controlling and monitoring various processes and functions, including power generation, water treatment, transportation systems, and production lines. As such, the protection of IACS from cyber threats has become a paramount concern for nations worldwide. Cyber attacks targeting IACS can result in the disruption, destruction, or manipulation of critical processes and infrastructure, leading to widespread consequences and significant impact on national security, economic stability, and public safety.

Given the risks associated with cyber threats, safeguarding IACS from cyber threats has become a top priority for governments, organizations, and security professionals. Ensuring the security and resilience of IACS is a critical aspect of a nation's defense plan, as the protection of critical infrastructure is essential for maintaining the stability and security of a country's economy, society, and national security.

Similar to conventional Information Technology (IT) networks, Operational Technology (OT) networks on which IACS are present also require robust cybersecurity measures to protect them from cyber threats. Similar to IT networks, a multi-layer security approach is commonly used to safeguard OT networks. This approach typically involves the implementation of various cybersecurity measures, including network segmentation, host-based security monitoring, and network monitoring [2].

Network segmentation and host-based security monitoring are crucial cybersecurity measures for protecting OT networks. Segmentation divides the OT network into isolated segments or zones to restrict unauthorized access. Host-based security monitoring con-

tinuously analyzes devices within the OT network for vulnerabilities and patches. These measures enhance cybersecurity, ensuring the secure and reliable operation of critical industrial processes.

Network monitoring is also an essential aspect of OT network security. It involves the continuous monitoring and analysis of network traffic for any signs of anomalous or malicious activity. Network monitoring tools and techniques, such as Network Intrusion Detection System (NIDS), Security Information and Event Management (SIEM) systems can help detect and respond to potential cyber threats in real-time, allowing for immediate incident response and mitigation.

To facilitate efficient operation and effective incident detection, NIDS rely on rule sets that require protocol dissection to interpret network traffic. These rule sets, created from existing intrusions, enable the NIDS system to detect anomalies and known attacks. However, to advance the capability of NIDS, further research on the underlying protocols is required to develop improved rule sets.

The research conducted in this study focused on the IEC 60870-5-104 (IEC 104) protocol, which is widely used in IACS. The goal of this research was to gain a comprehensive understanding of the IEC 104 protocol and its associated network traffic patterns, behaviors, and characteristics. This knowledge was utilized to develop effective rule sets and strategies with a goal of developing a working IEC 104 prototype to recognize malicious commands that use the IEC 104 protocol. The findings from this research are expected to contribute to the advancement of open-source NIDS.

Data acquisition, testing, and other practical needs for this research were obtained from Industrial and Automation Control System Security Laboratory located at CERT.LV headquarters in Riga, Latvia. The laboratory was built to model and replicate parts of the electricity generation, supply, and distribution across Latvia. It is described in more detail in Chapter 4

1.2 Hypothesis

The experimental analysis of malicious IEC-60870-5-104 traffic, conducted using a Network Intrusion Detection System prototype based on IEC-60870-5-104 Single and Double command detection, will be able to detect potentially harmful traffic.

1.3 Research Questions

1. What solutions, insights, and suggestions have been explored regarding IACS NIDS that look at the IEC 104 protocol?

   Sub-questions:

   (a) What IEC 104 NIDS and rule sets are available for ICS systems?

   (b) How well do the current open-source NIDS solutions support the IEC 104 protocol and what could be done to improve their capabilities?

   The methodology in this research question context involves conducting a comprehensive literature review.

2. What aspects of the IEC 104 protocol and its traffic need to be examined more closely in order to support open-source NIDS solutions?

   Sub-questions:

   (a) What would be the most effective approach for detecting anomalies in traffic generated by the IEC 104 protocol?

   (b) How can a deeper understanding of the IEC 104 protocol contribute to the development of more effective rules for open-source NIDS solutions?

   (c) Do the newly developed rules effectively detect known attacks in the context of the IEC 104?

   The methodology employed for this research question encompasses data analysis, experimentation, and validation.

1.4 Research scope, gaps, novelty, and methods

Research Scope covers the IEC 104 protocol and devices used in the previously mentioned IACS laboratory described in more detail in Chapter 4 which includes but is not limited

to devices such as Remote Terminal Unit (RTU)s, Routers, and switches. The IACS laboratory mimics many IACS used in the field making the key assumption that both systems work in the same manner.

The identified research gap consists of attack/anomaly detection capabilities of NIDS implementing real-time dissectors of IEC 104 protocol.

This research gap is a summary of findings and is discussed in more detail in chapter 2.

Novelty

Analysis of IEC 104 protocol and its traffic in combination with comparisons with different open-source NIDS solutions in a known attack scenario will bring more insight into crafting a NIDS solution for OT networks.

Research methods

Research methods will consist of literature analysis, where all relevant research regarding the IEC 104 protocol will be examined. After literature analysis, the identified gaps will be explored and considered to create a NIDS prototype based on the IEC 104 protocol. Finally, the prototype will be tested and validated as described in chapter 4, and the results analyzed with the goal of understanding future development options.

2. Background / Literature review

This section looks at relevant background information.

To comprehensively survey the relevant literature, a systematic approach was employed, involving the following workflow:

1. Utilize a set of carefully selected keywords to search for a seed set of papers from reputable journals that are relevant and align with papers published within the past 5 years (10 years for fundamental knowledge), as the field is fast-changing and dynamic.
2. Use backward snowballing technique to find more papers which were then analyzed in two parts. First, by reading the Abstract to assess the relevance or applicability and later the whole paper when redeemed that it might be useful.
3. Additionally, whitepapers and specifications were also included in the literature review to capture the current state of knowledge in the field. They were sourced by utilizing different search engines (Google search, DuckDuckGo) using the same keywords.

Keywords	ICS, IDS, SCADA, Intrusion Detection rules, Modbus, IEC 104, IEC-60870-5-104
Keyword combinations used for searching in Google scholar, Mendeley, and Semanticscholar	ICS IDS ICS IDS SCADA Intrusion detection rules ICS Intrusion detection rules SCADA SCADA IDS Modbus IDS IEC 104 IEC-60870-5-104 IEC 104 dissector

Table 2. Keywords and their combinations

Digital libraries:

Google Scholar, Mendeley (https://mendeley.com), and Semanticscholar

(https://semanticscholar.org) were primarily utilized for paper retrieval, with inclusion/exclusion criteria based on publication date, relevance, and source. Papers published within the last 5 years were favored, with some exceptions made for fundamental knowledge, such as understanding the inner workings of Modbus, IEC 60870-5-104 (IEC 104), and other protocols and concepts, where the age of the paper is less relevant due to the immutable nature of the protocols.

2.1 Background

To properly design a Network Intrusion Detection System (NIDS) for Industrial and Automation Control Systems (IACS), one must consider the fundamental differences between the industrial and conventional networks. Galloway et al. [3] summarizes these differences as displayed below in Figure 1.

	Industrial	**Conventional**
Primary Function	Control of physical equipment	Data processing and transfer
Applicable Domain	Manufacturing, processing and utility distribution	Corporate and home environments
Hierarchy	Deep, functionally separated hierarchies with many protocols and physical standards	Shallow, integrated hierarchies with uniform protocol and physical standard utilisation
Failure Severity	High	Low
Reliability Required	High	Moderate
Round Trip Times	250 μs - 10 ms	50+ ms
Determinism	High	Low
Data Composition	Small packets of periodic and aperiodic traffic	Large, aperiodic packets
Temporal Consistency	Required	Not required
Operating Environment	Hostile conditions, often featuring high levels of dust, heat and vibration	Clean environments, often specifically intended for sensitive equipment

Figure 1. *Main differences between Operational Technology (OT) -industrial and Information Technology (IT) - conventional networks [3]*

Since a NIDS is built on network traffic analysis, a deep understanding of the protocol suite used is required to successfully catch malicious network requests/commands. The literature analysis and research will give an overview of NIDS developed for OT networks with a focus on the IEC 60870 standard, with particular emphasis on the IEC 104 protocol due to its extensive use in Europe [4] and its availability in IACS laboratory located at CERT.LV headquarters in Riga, Latvia. It is important to note that the scope of this research covers NIDS which rely on analyzing and interpreting the IEC 104 protocol.

Network Intrusion Detection Systems

This section provides a contextual framework for NIDS that are designed for OT networks and Industrial Control System (ICS) by presenting previous research efforts in the field. It

is important to note that the content of this section is beyond the scope of the current thesis and is included to provide readers with additional background information.

To develop a NIDS, it is necessary to be able to differentiate between regular and malicious traffic. To accomplish this, researchers have experimented with numerous methods when attempting to simulate such traffic, they have used honeypots [5], [6] as well as simulating attacks such as command injection [7] in cases where the OT network might be compromised. In such cases a NIDS is required to recognize the breach. There are numerous types of NIDS which are used to help securing the ICS network - NIDS aided by deep learning models [8] as well as flow-based monitoring solutions [9] that help operators understand the traffic and detect anomalies thus providing valuable data for the NIDS. There are even attempts to generate NIDS rules from code [10] where the researchers developed a tool that generates 5 NIDS rules using Programmable Logic controller (PLC) code as the main input. There exist NIDS solutions for ICS (Suricata, Bro, Snort) but most of which only support the Modbus, and Ethernet/IP protocols [11].

Network Intrusion Detection Systems based on IEC 60870-5-104 protocol

IEC 104 is as Matoušek [12] summarizes: "part of IEC Telecontrol Equipment and Systems Standard IEC 60870-5 that provides a communication profile for sending basic telecontrol messages between two systems in electrical engineering and power system automation." As Pliatsios [13] clarify, it supports various network types within TCP/IP and defines the application layer of the Open System Interconnection (OSI) model. Due to its widespread usage across Europe, IEC 104 has gained the attention of security researchers who have identified several shortcomings in its security measures, particularly the lack of proper authorization mechanisms, which makes it vulnerable to various cyber attacks [14] such as Man-in-the-middle (MiTM) and unauthorized access attacks [15].

There are few different methods exerted into securing OT networks that base their protection mechanisms on the IEC 104 protocol, but they have numerous shortcomings. Wang et al. [16] proposed an Intrusion Detection System (IDS) model based on graphical features of IEC 104 operation data, which they then test by creating an IDS in Matlab, however, upon inspection, no code was made public by Wang et al for this project, proving difficult

to reproduce the results. Furthermore, Matlab's main functions include programming, app building, and other development/research based applications [17] and since it is not maintained as a dedicated IDS, it should not be used as one.

Researchers have also proposed more conventual methods like signature-based IDS [18], where they used a model-based approach by crafting a multitude of Snort NIDS rules. These Snort rules are based on examining specific components of the IEC 104 packet contents, which works in principle but each time one wants to create a new NIDS rule, one must closely re-examine the IEC 104 packet which might slow the development process. Yang et al. also created deep packet inspection frameworks to parse network traffic - Internet Traffic and Content Analyzer (ITACA)[19] and have used it to develop stateful NIDS with claims of it outperforming Zeek and Snort [20]. Although these works seem promising it was not possible to test the success of Yang et al., because of the unavailability of both - the ITACA tool (it seems that it is proprietary to Queen's University Belfast) and the code that Yang et al. developed. Additionally, the applicability and relevance of the ITACA is debatable, as the research conducted by Hurley et al. [19] and Yang et al. [20] is nearly a decade old.

Research approaches have drifted more towards using Scapy - a packet manipulating Python library [21]. Grammatikis et al. created an anomaly detection IDS by utilizing deep packet inspection with Scapy, see Figure 2 for an overview of their IDS [14].

Chromnik et al. have also developed deep packet inspection models based on Spicy [22], a plugin for protocol dissection for real-time packet processing in Zeek [23]. Udd et al. began the work in 2016 by creating a IEC 104 parser in BinPac (now Spicy)[24] and their work has since then been extended by Chromnik et al., where they tested their solution with two operator procedures: "(i) changing the state of one switch from connected to disconnected, and (ii) changing the set point (threshold) value of the maximum current on one power line, significantly"[25]. Researchers also made their efforts open-source by placing their code on GitHub [26]. The prototype developed by Chromnik et al. [25] was created in 2019 and currently is outdated. It was developed for Zeek (then called Bro) version 2.5.1 and does not work with the newest version of Zeek 5.2.0. It is noteworthy that Chromnik et al. describe and discuss various network-level policies in their research

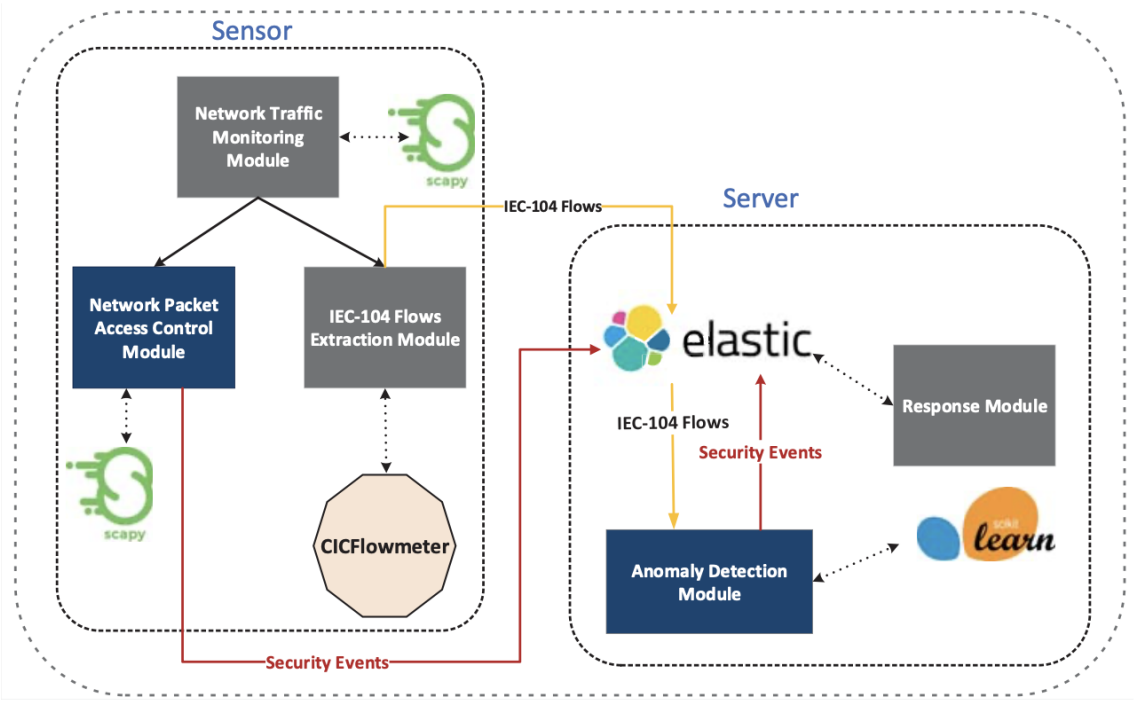

Figure 2. *Anomaly detection IDS using Scapy[14]*

paper for improving the monitoring of IEC 104 traffic using Zeek [25]. However, these policies are not present in the Zeek script and Interface Definition files available in the researcher's repository [26].

Although Scapy is well suited for testing, research, and development, it is not applicable for production environments because of the overhead which comes with processing packets with Python. On the other hand, Spicy is developed for Zeek – a real-time packet processing tool and is better suited for production environments. The speed differences are due to language specifics - Zeek and thus Spicy is based on C++ which is faster by design since it is statically typed and does not use an interpreter as does Python[27].

Taking into account the before-mentioned differences between Zeek and Scapy(Python), this research will concentrate on comparing the developed prototype with one created by Chromnik et al. [25].

IEC 104 attacks

Securing IACS and Supervisory Control and Data Acquisition (SCADA) systems that use the IEC 104 protocol requires consideration of potential attack vectors. These include

known vulnerabilities or weaknesses in the inner workings of the protocol as well as historical instances of observed attacks in the real world. The IEC 104 protocol has not been designed to be inherently secure and it lacks proper authorization mechanisms [14] as mentioned earlier in this section.

Erdodi et al. explored this design flaw by attempting numerous attacks, such as passive and active reconnaissance, operation failure, and denial of service attacks. Out of a total of 14 attacks, 11 were successful including operation failure attacks in which Erdodi et al. captured the traffic between the control center and the Remote Terminal Unit (RTU) that it communicated with and inserted a rogue Single Command Application Service Data Unit (ASDU) type thus changing the state of the circuit [28]. See Table 3 for an overview of different attack vectors used by researchers.

Attack vectors	**Tools used**	**Year**	**Reference**
Reconnaissance, Unauthorized Acc.	NA	2022	[28]
Unauthorized Access	*Python*	2021	[29]
Unauthorized Access, injection	NA	2020	[30]
DoS, Unauthorized Acc., MiTM	*hping*, *OpenMUC j60870*	2019	[31]

Table 3. *Different IEC 104 attack vectors*

To perform an Unauthorized Access attack researchers in both, [29] and [28] created a Transmition Control Protocol (TCP) packet with a Single Command ASDU type to attack the RTU. This packet was then sent to the RTU thus changing the state of a switch by closing/opening it. While this and other attack vectors, methods, and tools have been extensively described in existing literature, the underlying code base that facilitates these attacks is often left unaddressed, which was the case with methods utilized by researchers in [31], [30], and [28]. The actual code used for the attacks was only available in [29].

On December 17, 2016 a transmission substation in northern Kiev lost power and it was due to a cyber attack [32]. This attack was later investigated by Dragos - a US based threat intelligence company concentrating on ICS OT networks [33] and ESET - a Slovakian company focusing on anti-virus development [34]. White paper created by Dragos details that the attack occurred in multiple stages by the attackers first obtaining access to the internal IT network, establishing persistence, and pivoting to IT data historian, which provided the attackers access to the OT network. After learning the specifics of the

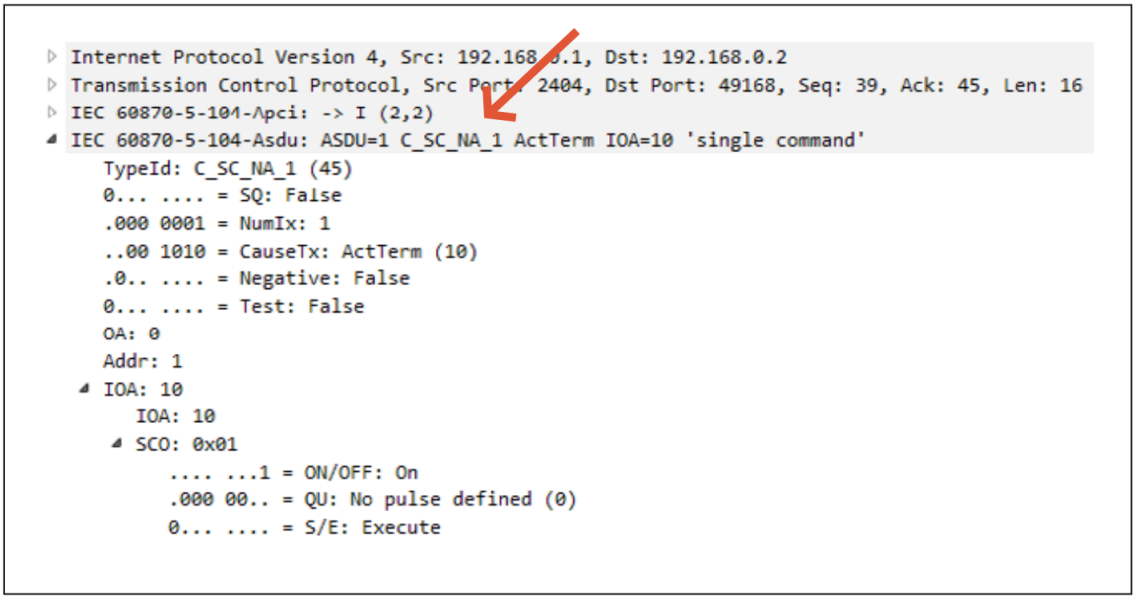

Figure 3. *IEC 104 packet sent to the RTU [36]*

network and developing malware which Dragos named *CRASHOVERRIDE* - first ever framework developed to attack electric grids, the attackers were ready for the final stage - execution. The framework included a module tailored for the IEC 104 protocol, which had the following capabilities - setting of specific values, Information Object Address (IOA)s enumeration, setting IOAs to the *open* state or continuously toggling the IOA between *open* and *closed* states. These commands can then be sent to a RTU [35]. ESET also published a white paper which includes an important detail concerning the specific command sent to a RTU - they provide a screenshot of the specific packet which triggers the *open* state, see Figure 3. As one can observe, the packet consists of the Single Command information object S_SC_NA_1 (highlighted in Figure 3) [36].

3. Prototype

In this chapter, the developed prototype will be discussed starting by providing background information on Transmition Control Protocol (TCP), IEC 60870-5-104 (IEC 104), and Zeek and Spicy followed by the development process and rationale for implementing a rule-based Network Intrusion Detection System (NIDS).

3.1 Context & Background information

To analyze and understand the developed NIDS prototype, it is important to discuss and understand some background information. That includes the following topics: selected technical inner-workings and specifics of the TCP and IEC 104 protocols, the framework of open-source NIDS, and Spicy, which is a plugin utilized for developing custom protocol dissectors in Zeek.

TCP and IEC104

The IEC 104 protocol delivers information as application data (Layer 7) over the TCP protocol using port 2404 [12]. This implies that all communications using the IEC 104 protocol are established on top of the TCP protocol and therefore, must adhere to TCP specifics. Most notably this means that all IEC 104 communications need to establish a TCP session by utilizing a three-way handshake with the target device.

The IEC-60870-5 standard specifies that every station/device is either a controlling station or a controlled station. This in turn means that the IEC 104 protocol works on a master-slave basis with the master usually being a computer with a Supervisory Control and Data Acquisition (SCADA) system installed and the slave being the Remote Terminal Unit (RTU) controlled by the computer.

Whenever a SCADA system needs to send a command to a RTU, it needs to know the RTU's IP address as well as the Information Object Address (IOA) for the specific

command/information request. The IOA address represents a unique address that identifies a specific physical device, such as a relay or a switch, within the substation automation system which is controlled by the RTU. A wide range of commands and information requests can be transmitted to a RTU, varying from requests for information that prompt the end device to send physical readings back to the SCADA system to commands that open or close switches, thus enabling or stopping the flow of electricity.

IEC 104 packets

The IEC 104 network traffic consists of Application Protocol Data Unit (APDU) packets each of which containing a Application Protocol Control Information (APCI) section and, optionally, an Application Service Data Unit (ASDU). See Figure 4 for an illustration of the packet structure. Note that a APDU does not always include an ASDU.

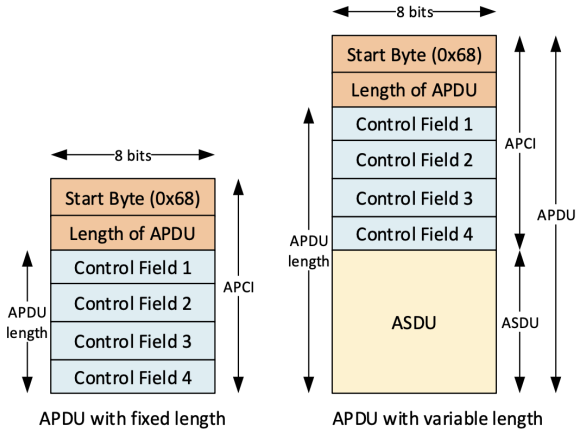

Figure 4. *IEC 104 packet structure* [12]

The described packet structure is determined by its classification into one of three subgroups - Unnumbered control function (U-format), Numbered supervisory function (S-format), and Information transfer format (I-format). Both U-format and S-format packets have an APDU with a fixed length of 4 bytes consisting of four one-byte long Control Field as seen in Figure 4. S-format packets exert a supervisory function and U-format packets provide a control function by specifying the beginning or end of data transfer, which provides context but does not give any information on the type of data transferred. I-format packets transfer data, which can have several functions ranging from voltage readings to controlling Programmable Logic controller (PLC)s by sending them commands, such as

Single Command (SC) or Double Command (DC) and are explained in more detail in the Prototype Development part of this chapter - Section 3.2. For more information and specifics regarding the IEC 104 protocol see work done by Matousek et al. [12] or the IEC 104 standard [37].

Zeek and Spicy

Zeek is a versatile open-source network analysis framework that can be utilized for both - to analyze network traffic and as a NIDS [23]. It incorporates a range of built-in analyzers for protocols commonly employed in Information Technology (IT) networks, for example, Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), Domain Name System (DNS), BitTorrent. A comprehensive listing of these analyzers is available at [38]. These analyzers can then be leveraged to analyze network traffic and limit malicious requests by utilizing Zeek's built-in NetControl framework. See Figure 2 for an overview of Zeek's architecture.

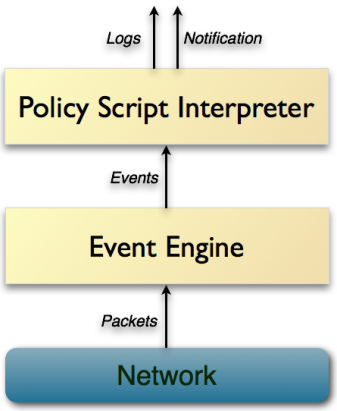

Figure 5. *High-level Zeek architecture [39]*

Despite being a powerful network analysis tool, Zeek does have certain limitations. One such limitation is the lack of built-in support for a multitude of Industrial Control System (ICS) protocols, such as Distributed Network Protocol 3 (DNP3), IEC 60870-5-101 (IEC 101), and IEC 104 protocol. The likely reason for this limitation is the disparity in use cases, with IT networks having more use cases compared to Operational Technology (OT) networks. Additionally, IT networks are generally more widespread than OT networks. However, Spicy, a Zeek plugin for protocol dissection, can be used to address this limitation. Spicy is a programming language/wrapper that allows users to analyze non-supported

protocols by generating C++ code from a more simple and understandable Spicy language. This C++ code can then be compiled and later interpreted by Zeek, extending the platform's capabilities.

According to Spicy guidelines, one must complete the following steps to be able to create an analyzer that can parse and analyze any IT or OT network protocol's traffic:

1. Create a Spicy file describing the grammar of the protocol. This file serves as a crucial resource in the analyzer since it gives important context to raw bytes received from the network packets.
2. Create an Interface Definition file describing conditions at which the Spicy plugin must be activated. This file is responsible for creating events that the Zeek Engine can understand. This file often is described as "glue" that sticks the Spicy plugin and Zeek together serving as a vital intermediary component.
3. Construct the Zeek script file which describes the actions that Zeek will take once it sees traffic described by the Interface Definition file. This script file has all the capabilities available by the Zeek project from custom logging to network traffic flow control with the NetControl framework. In the context of this research, this file should be viewed as the NIDS rule-file. Further discussion can be found in the prototype development section - 3.2.

Spicy and Interface Definition files are fed into the Zeek event engine, which in turn processes the network packets and finally forwards the events to the Policy script interpreter for which the Zeek script file describes the appropriate actions. This process can be seen represented in Figure 6.

3.2 Prototype development

Spicy script file for IEC 104 dissection

The Spicy file created during this research, which describes the grammar of the IEC 104 protocol follows Chromnik et al. work[25] available on GitHub [26] and aims to improve it due to obsolescence and incompatibility with the latest version of Zeek. Chromnik et

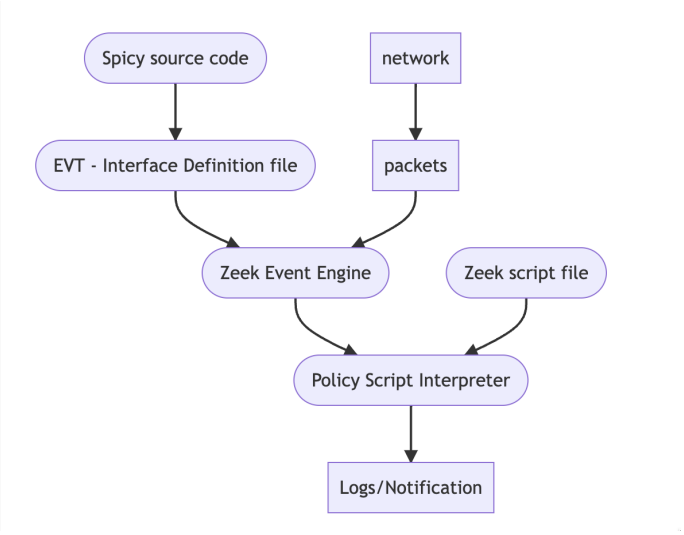

Figure 6. *Integrating custom Spicy dissector into Zeek*

al. work was developed for Zeek version 2.5.1 (then called *Bro*). The current version of Zeek is 5.2.0 and it does not support the Spicy code developed by Chromnik et al. The Spicy code created during this research addresses multiple issues with Chromnik et al.'s code including a missing step in the code's logic which is discussed in more detail later in this section. As discussed in Chapter 2, work done by Chromink et al. was the only one available that met both - the requirements of implementing an NIDS based on the IEC 104 protocol as well as having the source code of their efforts available for inspection.

The developed Spicy file, which accounts for dissecting the IEC 104 grammar adhered to the following sequence of steps when parsing one packet:

1. Recognize the IEC 104 protocol packet by seeing an "x68" byte. This byte, when converted to an integer value, is 104 and signals the application layer, which processes the TCP connection, that the packet received is part of the IEC 104 protocol. The "x68" byte is the very first byte of IEC 104 protocol and it is present in all IEC 104 packets.
2. Record and analyze the following five bytes at offset 1. As it can be seen in Figure 4, the sequential five bytes consist of one byte specifying the length of the APDU and the remaining four bytes to four Control Fields, which define the APDU type of which there are three: I-format, U-format, and S-format. The code analyzes these five bytes to determine whether the packet is in I-format. If true, the code continues

to the appropriate section to process I-format packets.

3. Process the I-format packets. I-format packets contain an ASDU which is responsible for issuing commands to the end device with which the RTU is communicating. See Figure 7 for an overview of the ASDU packet structure. An ASDU can contain up to 127 objects, where an object corresponds to one of 127 different types of information objects. It is important to note that an ASDU can only carry one type of object meaning that if an ASDU has 100 objects, all of which would be of the same object type. These objects can transmit various types of information or commands, such as "single point information", "measured value", "single command", or "double command".
4. If the IEC 104 packet is not in an I-format, it will be either a S-format or an U-format packet. Both types are not as crucial from a network security standpoint and were discussed in more detail in Section 3.1

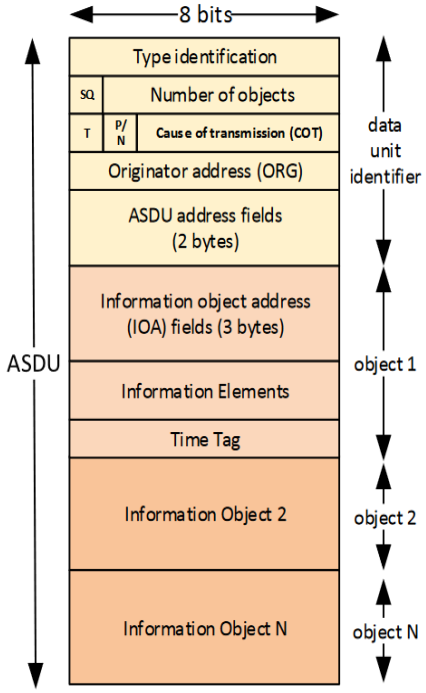

Figure 7. *IEC 104 ASDU packet structure [12]*

Chromnik et al's work had a multitude of issues when executed with the newest version of Zeek. Because of that, the Spicy code developed during this research was restructured and built from scratch. It followed the main flow of Chromnik et al's work but virtually every line had to be examined to be sure that the prototype's Spicy code will work. The

Spicy code was written in small steps, continuously writing few lines, and then testing the grammar's capability to understand raw input bytes. Testing was done by using Linux built-in *printf* tool to feed raw bytes into the *spicy-driver* utility present in the Spicy plugin. An example of this can be seen in Figure 8. Note that this example is meant as a visualization of the development process and in it one can see a simple U-format packet of APDU length equal to 4. The raw bytes were built one by one assembling the IEC 104 packet. Once the stage of parsing a complete packet was complete, the other two files necessary for the dissector were created and the prototype was tested with actual traffic gathered from the Industrial and Automation Control Systems (IACS) laboratory, which is described in more detail in Chapter 4.

```
root@1430b1eb09c9:/zeek/spicy_lietas/spicy-IEC104# printf '\x68\x04\x83\x00\x00\x00' | spicy-driver I104.spicy
ALL, [$start=104, $apdu_len=4, $ctrl=(0, 0, 0, 0, 0, 0, 16768), $asdu=(not set)]
DONE APCI
```

Figure 8. *Spicy example*

Although most of the improvements to Chromnik et al's work were made to ensure that the prototype's Spicy code would be compatible with the newest version of Zeek, it was apparent that there was an issue with the underlying logic present in the Spicy code created by Chromnik et al. [25]. The issue was related to the second step in the dissector logic, which was described earlier in this section. Specifically, the logic of the code developed by the researchers mishandled the first control field of the APCI header (Control field 1 in Figure 4) which prevented it to process I-format packets - line 195 in the *t104.spicy* file in the *IEC104_data* folder [26]:

```
asdu : Asdu if (self.ctrl.not_i_mode == 0);
```

This seemingly minor issue made it impossible to dissect network traffic containing IEC 104 packets since the logic issue prevented the Spicy grammar file to differentiate between I-format, U-format, and S-format packets. It was difficult to pinpoint this issue since analyzing network traffic directly with Chromnik et al.'s work was successful, but the prototype was failing.

After extensive debugging it became apparent that an extra conditional clause was required and added by the author to allow for appropriate I-format packet processing. The added clause to the *if* statement solidifies the logic required to recognize the I-format packets.

While Chromnik et al. logic relies solely on the last bit of the Control Field 1 seen in Figure 4, the grammar file developed in this research provides an extra clause to be certain that the packet bytes following the 4 Control fields are indeed of the I-format type thus containing an ASDU. This change (adding "self.apdu_len > 4") can be seen in line 58 in the *I104.spicy* file of this research's GitHub [40]:

```
asdu : Asdu if (self.ctrl.not_i_mode == 0 && self.apdu_len > 4);
```

The exact cause of this logic issue in Chromnik et al.[25] remains unclear. These issues are most likely due to the use of different Spicy versions when compiling the Spicy files (Spicy source code and EVT file) to create a binary file that Zeek uses at runtime to parse network traffic.

There were numerous other changes and adjustments needed to ensure that the developed Spicy code runs with the latest version of Zeek. For example, the IEC 104 protocol specifies that one APDU packet can have numerous (up to 127) information objects in its ASDU. To accommodate this, Chromnik et al. had the following in their *t104.spicy* file in the *IEC104_data* folder [26] (line 172):

```
:  list<Apci> &transient &until(False); #TODO better solution needed
```

which aims to create a list of APCI data type units. As Chromnik et al. specify themselves, their implementation could be improved and as it became evident during this research, it did not work with the newest version of Spicy and had to be adjusted to the following (line 20 in the *I104.spicy* [40]):

```
apcis :  Apci()[];
```

Interface and Zeek script (rule) files

This file is the NIDS rule file and sets the work done in this research apart from Chromnik et al.'s. The main differences are in the Interface Definition and Zeek script files. The Zeek script file assumes a paramount role in the NIDS framework, as it allows to specify the appropriate action(s) that should be taken by Zeek upon detecting specific network traffic patterns. Essentially, this file defines the conditions under which certain actions should

be executed and provides a roadmap for detecting and responding to potential security threats. As such, the development and maintenance of an effective rule/Zeek script file is of paramount importance in ensuring the overall effectiveness of the NIDS framework. In short, the Zeek script file should be viewed as the rule file in the NIDS context. It is viewed as one of the main contributions of this research.

Most of the advancements were made in the Zeek script file, *I104.zeek* since the Interface (*I104.evt*) file does not do any data analysis or processing but rather acts as an in-between layer to pass information and relevant events between the Spicy and Zeek files, as described in Zeek and Spicy part of Section 3.1. The Zeek script developed in this prototype extends the capabilities of the prototype to move towards the goal of furthering open-source NIDS development by providing more contextual information when looking at IEC 104 traffic.

The Zeek script file accomplishes that by analyzing the Information Objects present in every ASDU, which exists in the I-format packets. See Figure 7 and Figure 4 for a visualization. More particularly, the Zeek script looks for C_SC_NA_1 and C_DC_NA_1 Information Object codes which translate to Single and Double commands, respectively. It then stores the bits describing which command is sent and logs it together with other connection-related details such as IP addresses and ports. The code snippet, which logs the C_SC_NA_1 command can be seen here (lines 31-51 in the *I104.zeek* file [40]):

```
event I104::single_command(c: connection, info_obj_addr: int, scs: int)
{
        local command: string = "";
        if (scs == 0)
                {
                        command = "OFF";
                }
        else if (scs == 1)
                {
                        command = "ON";
                }
        else
                {
                        command = "ON/OFF";
                }
```

```
        # Create Info object defined in lines 9-14
        local rec: I104::Info = [$ts=network_time(), $id=c$id,
                                 $command_single=command];
        # Log the rec object
        c$I104 = rec;
        Log::write(I104::LOG, rec);
}
```

A sample log file created during the validation of the prototype described in Chapter 4 can be seen in Figure 16 and is present in the Appendix. These commands are usually sent by the SCADA system to the RTU to open/close a switch or, in other words, turn the flow of electricity OFF/ON. The *I104.zeek* logic described in this paragraph can be seen in Figure 9. This allows Zeek to log ON/OFF commands that a RTU receives. Note that the *if* statement defaults to the command being in ON/OFF state, which is not possible in a real-world scenario but was added to the rule-set for debug purposes.

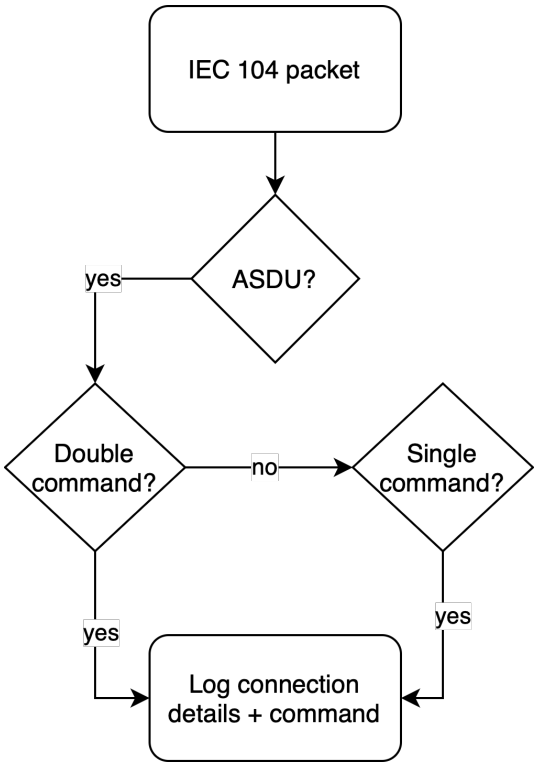

Figure 9. *High-level Zeek script logic [40]*

As discussed in more detail in Chapter 5, Chromnik et al.'s script file provides a statistical-oriented approach when seeing network traffic data and it does not provide any contextual information when seeing Information Objects.

The functionality of the NIDS prototype is not limited to one RTU and can be applied in situations where an OT network has more than one RTU to account for. An example use case can be seen in Figure 10. In this case, the NIDS prototype would be able to detect the threat actor's IP address if they would start to communicate with any of the RTUs. At the current development stage, the prototype would only log the incoming connections, but this functionality can be extended to dropping requests based on some characteristics (e.g., unrecognized IP address).

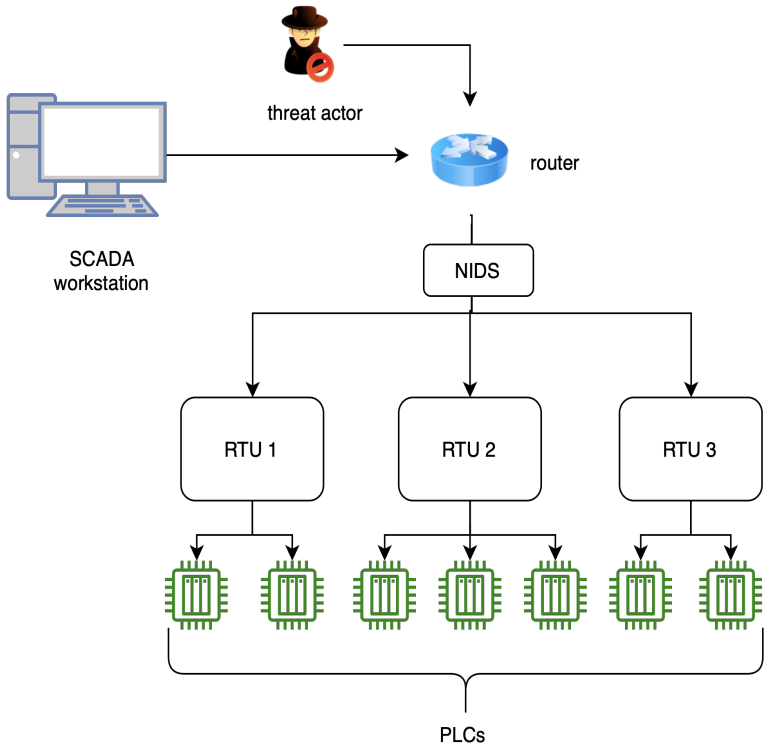

Figure 10. *Example use case of the NIDS prototype*

The prototype was developed and tested using Docker to ensure the prototype's portability, reproducibility, and isolation. The docker image used for testing and development is a modified version of the official Spicy docker image available on Docker Hub [41]. The official docker image was modified to install the Spicy plugin as well as Zeek. It is available on Docker Hub [42].

3.3 Context for rule-based NIDS development

The security engineering team at CERT.LV had conducted thorough research and development with respect to both IT and OT NIDS. This involved a comprehensive analysis of potential security threats, the identification of key network traffic patterns, and the subsequent development of an NIDS framework that is capable of effectively detecting and responding to these threats. The OT sensors employ Zeek to process and analyze network traffic. The scope of the OT sensor covers the electrical energy grid, which includes a substantial amount of different systems and protocols with the IEC 104 protocol representing a fraction of the overall network traffic. Using Zeek gives a good insigth into the network traffic and potential threats but it has its weaknesses, specifically, one key limitation of Zeek is its inability to effectively process ICS traffic, which includes IEC 104, as discussed previously in this chapter, in Section 3.1. This limitation hindered the development of the OT sensor and provided additional incentive to create a IEC 104 dissector, which could be integrated into the OT sensor.

The main goal of this study in the context of the OT sensor is to expand its capabilities by detecting, recognizing, analyzing, and reacting to IEC 104 traffic. It is also expected for the study to aid at advancing the OT sensor in other ways by providing a template for other protocol dissection with Spicy, for example.

A rule-based NIDS prototype was chosen to provide a Proof of concept (POC) view of a NIDS, which is based on the IEC 104 protocol for the OT sensor. At its current development stage it is not ready to deploy in a production environment and will be further designed to incorporate more rules and Zeek actions. The development of other NIDS methods (anomaly-based, machine-learning models) is out of scope for this research and will be examined and compared to the developed prototype more closely in the future. See further discussion of future work in Chapter 5.

As discussed in Section 3.2, the Zeek script file, which acts as the main NIDS file, looks at two IEC 104 commands to detect potentially malicious network traffic - Single and Double commands. These two commands were chosen based on previous research done on the topic and the availability of tools and methods used by the researchers. The Single

command was also used in a real world attack to a transmission substation in Kiev. See end of Chapter 2 for further discussion and more details.

4. Prototype Validation

To verify the effectiveness and reliability of the Network Intrusion Detection System (NIDS) prototype, an experiment was carried out to test and validate its functionality. The experiment was designed to evaluate the prototype's ability to detect specific types of attacks. This chapter provides an overview of the laboratory used in the study, including the equipment and architecture, threat scenario, which is emulated for the experiment and discusses the methods used to gather data, such as the tools used to capture live network traffic and tools that were used to execute the attacker commands.

Overall, this chapter serves as a foundational introduction to the Information Technology Security Incident Response Institution of the Republic of Latvia (CERT.LV) Industrial and Automation Control Systems (IACS) laboratory and the methods used to test the developed prototype. It sets the stage for the presentation and discussion of the results in Chapter 5, which will provide a more detailed and comprehensive analysis of the experiment's findings.

4.1 IACS Laboratory Overview

The laboratory was built to mimic parts of the Latvian electrical production and distribution network, which includes gas operators. The laboratory uses protocols, devices, and tools used on-site in real life. For example, Station 2 from Figure 11 mimics Inčukalns - the only natural gas storage in the Baltics which, if necessary, can increase its capacity to 3.2 billion cubic meters of active natural gas. That is enough to cover Latvia's and nearby region's need for fuel [43].

For this research, permission was obtained to utilize a laboratory facility that the author did not construct. It was granted specifically for this purpose by CERT.LV [1] and provided a suitable environment to conduct the necessary experimentation.

The laboratory was built with modularity in mind which means that each of the seven

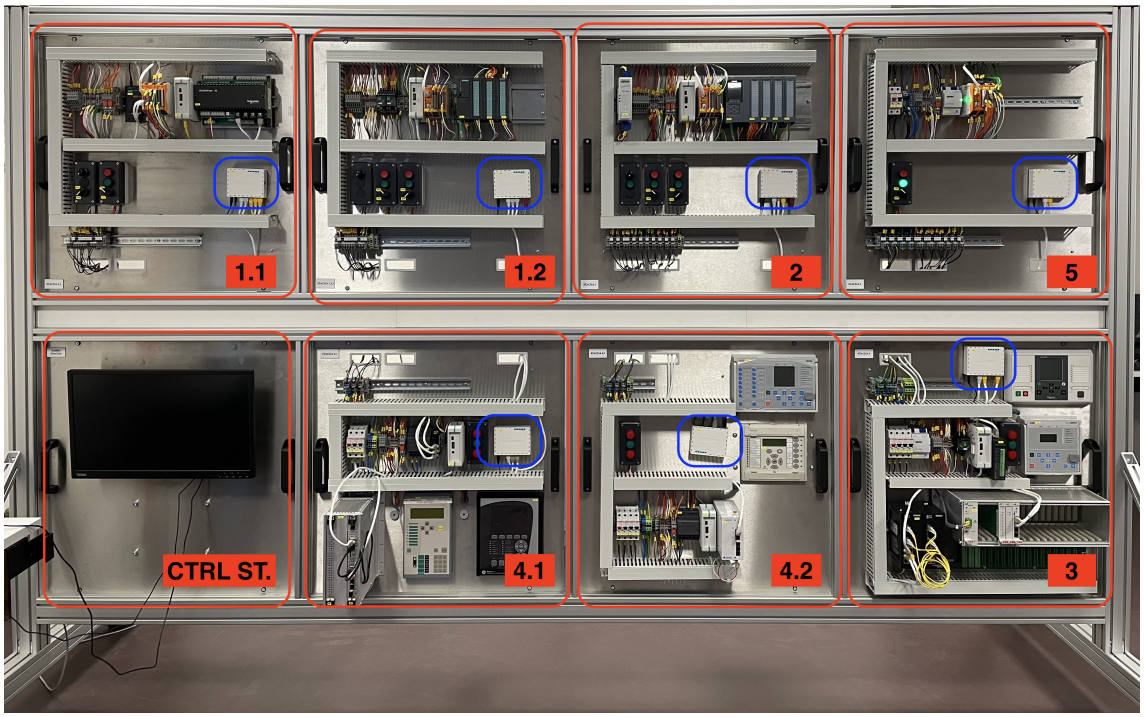

Figure 11. *IACS lab with stations*

blocks can be removed and run separately. The seven blocks as well as the computer monitor used for controlling the lab system can be observed in Figure 11.

4.1.1 Stations and functional blocks

This section details the functional blocks that comprise the IACS laboratory, providing insights into its inner workings.

Station 1.1

This station controls V2 valve which, together with valve V1, controls the gas supply to Station 5 which mimics a Thermal Power Plant thus powering the whole IACS lab system. It uses Modbus Remote Terminal Unit (RTU) and Modbus TCP/IP protocols to accomplish this.

There are no physical valves to open/close, that is simulated with two industrial Raspberry Pis acting as RTUs. The valves then can be controlled manually with a physical switch or with software available on the lab computer.

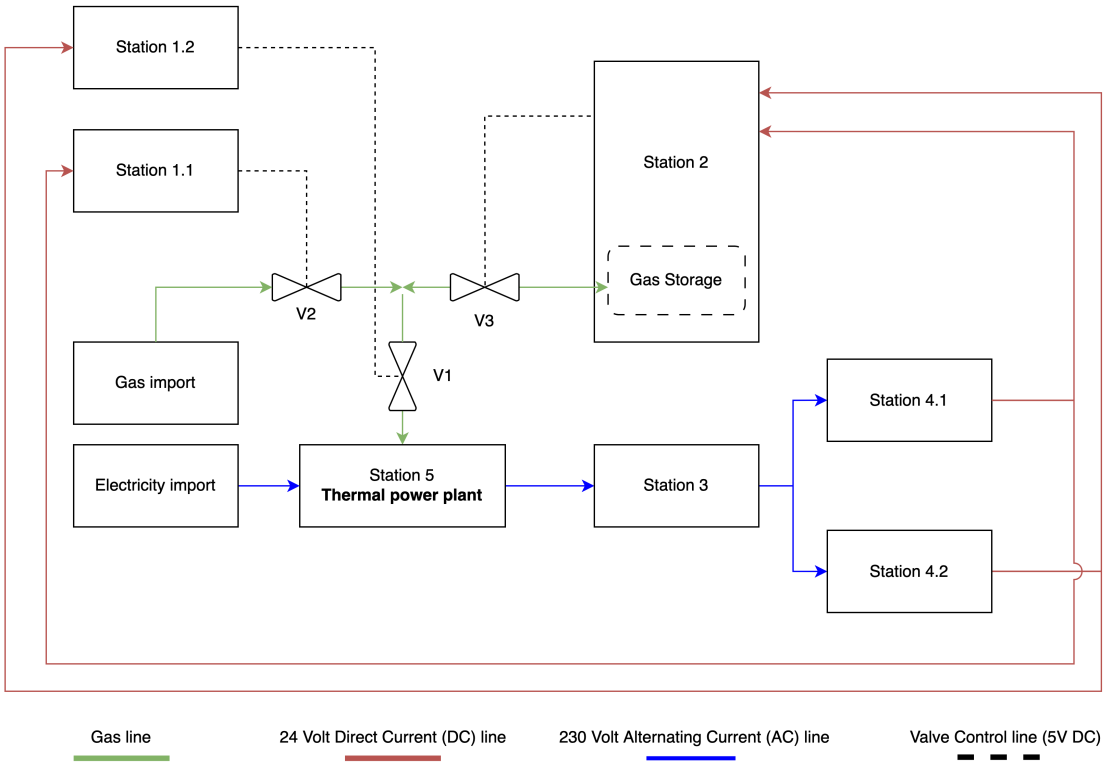

Figure 12. *IACS lab Functional Block Diagram (FBD)*

Station 1.2

This station controls valve V1 thus determining whether gas will reach Station 5 which is the thermal power plant. It uses Siemens S7 and Profinet protocols. Valve operations are simulated similarly to Station 1.1 by using Raspberry Pis.

Station 2

This station represents the Incukalns gas storage facility. It uses Siemens7 or Profinet protocols as well as industry-level Raspberry Pies to simulate the valve controller for valve V2. It also monitors the gas level inside the storage facility which can be seen in the Supervisory Control and Data Acquisition (SCADA) system.

Station 3

This is a sub-station that splits power generated by station 5 (from burning gas) to stations 4.1 & 4.2. It uses IEC 60850, GOOSE, and IEC 60870-5-104 (IEC 104) protocols with redundant communication links as it is commonly done in the real-world environment.

Station 4.1

Main elements are two protective relays that are connected to one RTU using IEC 61850 using two different channels. The RTU device is integrated into the main IACS lab control system using IEC 104 protocol. Station's main power switch controls power to stations 1.1, and 2 via the simulated 24V DC line. Station also provides a visual indication of the main power switch.

Station 4.2

This station is similar to Station 4.1 but it uses IEC-60870-5-103 instead of IEC-61850 to communicate with the RTU which then communicates with the main SCADA control system. It gives power to Stations 1.2 and 2.

Station 5

This is the Thermal power plant Station which simulates Electricity generation. It uses gas from a gas storage facility (Station 2) or from a gas import location. This station has the main power switch that can turn the whole IACS lab system on/off.

Laboratory networking

The IACS laboratory can be controlled manually by pressing either of the two (red/green) buttons seen on each Station or via the SCADA software on the *CTRL ST*, which can communicate with all Stations. The *CTRL ST* is connected to all other stations via a switch and seven Mikrotik Routerboards, which are connected to the switch and configured in bridge mode (i.e., they act as a switch) bridging each Station's Local Area Network (LAN) with the *CTRL ST*. They can be seen in Figure 11 - they are the small white boxes present in every Station and have a blue square around them.

Each station's LAN operates differently due to the presence of various components and each of the Mikrotik Routerboards can be accessed and configured as needed via the web console. See figure 13 for an overview of the network topology.

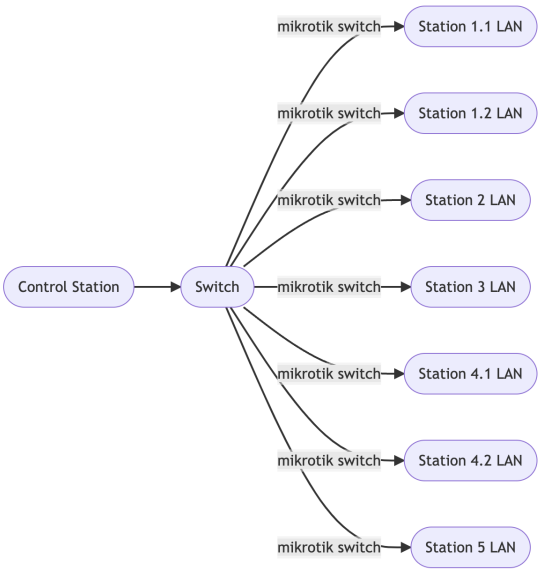

Figure 13. *High-level network graph of the IACS laboratory*

4.2 Threat scenario

This part of Chapter 4 details the threat scenario emulated by the experiment, including an examination of the attack surface and scope.

The scope of the simulated attack was concentrated around Station 4.2 and assumed that the attacker had obtained access to the internal Operational Technology (OT) network and could communicate with any device situated in the station's LAN. Essentially this means that the threat actor had gotten to the second stage in the Purdue Model for Industrial Control System (ICS) networks [44]. The chain of events performed by the attacker that could have led to this stage are not trivial and include breaching both - the Information Technology (IT) and OT networks. A hypothetical chain of events include the following: initial access to the IT network via a supply-chain attack, pivoting to the OT network due to poor network segmentation (the IT and OT networks are not separated properly) thus circumventing any firewalls or Demilitarized Zone (DMZ) that might be present. Finally, due to poor practices implemented by the IACS engineers, the attacker was able to see and capture network traffic which gives them the ability to map the OT network and gain important details to expand the attack to affect the operation of IACS. Details, such as IP addresses, ports, and Information Object Address (IOA) for a RTU are presumed to have been acquired by the threat actor.

After sufficient analysis of the network traffic seen, the attacker then had the opportunity to try to access any Station's RTU and the threat-scenario assumed that they chose Station 4.2 since its RTU had the weakest host-based protection enabled. An engineer had added multiple IP addresses to the RTU firewall's whitelist allowing the attacker to effortlessly communicate with it.

4.3 Experiment

This section details how the experiment to test and compare the developed prototype to the solution developed by Chromnik et al. [25] was carried out. It elaborates on the setup, tools, and methodology used in the experiment.

Setup

The experimental setup followed the scope detailed in section 4.2. Most importantly it:

- Stayed in the network bounds of Station 4.2
- Only accounted for network traffic related to the IEC 104 protocol
- Covered the network traffic concerning the RTU

A vital aspect of a NIDS is to be able to see raw network traffic. In most cases, this is achieved by placing the NIDS in such a way that it sees all network traffic, as shown in Figure 14[45].

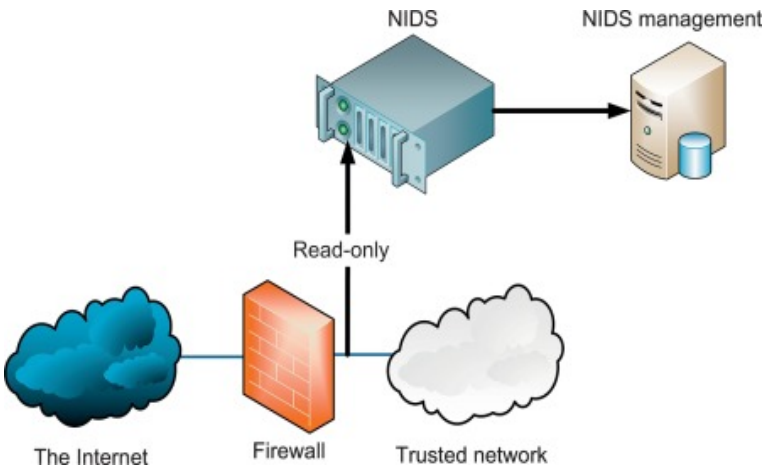

Figure 14. *Standard NIDS placement [45]*

In the experiment this was achieved by mirroring network traffic that was transmitted or received by RTU in Station 4.2. Port mirroring was configured on Station 4.2's Mikrotik, it was configured to mirror all ingress and egress traffic from port 3 to port 2. This enabled network traffic analysis on port 3 by connecting a computer to port 2 and configuring Zeek/Wireshark to analyze/capture the traffic.

Most importantly, this allowed all vertical traffic originating from and going to the main *CTRL ST.* to be mirrored thus linking the module used for experimentation to the IACS laboratory. See Figure 15 for a visualization of the described setup.

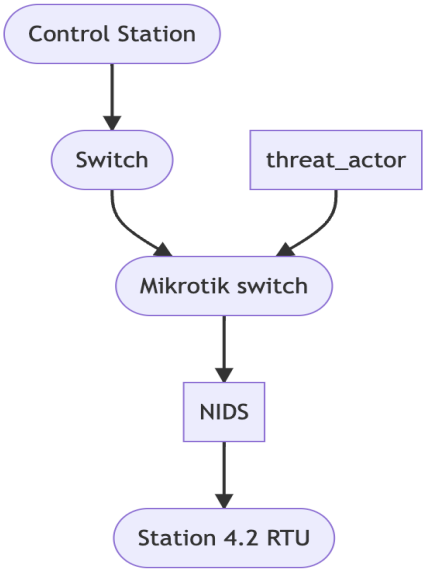

Figure 15. *Experimental NIDS setup*

Tools used

The following tools were utilized during the experiment:

- Zeek, which was used in conjunction with the Spicy script described in Section 3
- Wireshark, which was used to capture traffic in each experimental run to validate and compare the output generated by both Zeek/Bro and Spicy scripts.
- iec104.py - a python script, which injected the malicious commands. Explained in more detail later in this section [29].
- Docker with Bro (older version of Zeek) used in conjunction with the Spicy script developed by Chromnik et al. This older version of Zeek was necessary to run

Chromnik et al.'s script and was obtained from their GitHub [26].

The Python script used in the experiment was a slightly modified version of IEC 104 script developed by Blumbergs [29], which leverages the IEC 104 security flaws discussed in Section 2. The script was utilized to send ON/OFF commands to the RTU. See *Interface and Zeek script files* section of Chapter 3 for more information regarding these commands and their appearance in network traffic.

Finally, the mirrored traffic captured with Wireshark was exposed to both - the prototype's scripts and the scripts developed by Chromnik et al. [25]. This allowed to compare both solutions using the gathered data.

Methodology

The Methodology of the experiment was split into the following steps:

1. Setting up the research environment, which involved the following sub-steps:
   (a) Turning on the IACS laboratory using the *CTRL ST* computer
   (b) Connecting a computer to port 2 which mirrors all traffic going to/from the RTU on Station 4.2 using an Ethernet cable.
   (c) Emulating an attack in the second level of the Purdue model by the threat actor to the IACS laboratory's OT network. In practice this was done by connecting the attacking machine with the laboratory's WiFi network.
2. Commencing the data gathering process with Wireshark on the network interface which was connected to the mirrored port detailed in Setup section of this chapter.
3. Injecting both legitimate and malicious traffic into the network, achieved through the following steps:
   (a) Sending 1 OFF and 1 ON command from the *CTRL ST* to the RTU and waiting 3 minutes in between to simulate regular engineering work and leaving the station in the ON state.
   (b) Sending 1 OFF command from the attacking machine using the script described previously to simulate malicious behavior. (1.5 min after the RTU returned to ON state). The command executed was the following:

```
python3 2_iec104.py -e en0 -t 10.10.42.252 -i 8001 -s 0
```

where *-e en0* specifies the correct network interface, *-t 10.10.42.252* specifies the IP address of the RTU, *-i 8001* specifies the IOA of the RTU, and *-s 0* tells the script to send an OFF command.

(c) Sending 1 ON command from the *CTRL ST* to return the system to its baseline 30 seconds after the RTU turned OFF from the previous step

4. Ending the data gathering process by stopping Wireshark and saving the data in to a *pcap* file for analysis.
5. Analyzing the captured *pcap* file with the solution provided by Chromink et al. [25] and the developed prototype discussed in Chapter 3

5. Results

As discussed in the Methodology section of Chapter 4, the generated *pcap* file was analyzed with both - the developed Network Intrusion Detection System (NIDS) prototype and the tool created by Chromnik et al. [25]. The analysis generated three files in total - two when using the tool created by Chromnik et al. - "typeid_count_output.txt" and "lvl_isu_-output.txt" and one when analyzing the *pcap* file with the developed prototype - "I104.log". All the before-mentioned files are available in the Appendix.

The main conclusions that can be drawn from the resulting text files are the following:

- The created prototype works as intended and provides the expected results. That can be observed in the log file generated by the NIDS prototype - ON/OFF commands as well as other information, such as IP addresses, ports, and time are apparent in the analyzed network traffic, see the last two columns in Figure 16 - the command columns.
- The prototype shows more relevant information regarding the IEC 60870-5-104 (IEC 104) protocol traffic security events. When observing Figure 16, different IP addresses are seen sending commands to the Remote Terminal Unit (RTU), which signals that more than one device is communicating with it. In most cases, only one Supervisory Control and Data Acquisition (SCADA) station should be able to control RTUs, and the log file generated by the NIDS prototype clearly shows two source IPs, which indicates that a threat actor is present.

When observing the *I104.log* file obtained from analysis the *pcap* file discussed in the Methodology section in Chapter 4 one can see duplicates of OFF commands (see Figure 16).

The cause of this is the RTU sending ActCon and ActTerm Application Service Data Unit (ASDU) in separate packets instead of one. This is normal IEC 104 protocol behaviour that was also observed by researchers in [12]. Similar behavior can also be observed

I104.log - Notepad

File Edit Format View Help

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   I104
#open   2023-04-12-13-26-25
#fields ts      id.orig_h       id.orig_p       id.resp_h       id.resp_p       command_single  command_double
#types  time    addr    port    addr    port    string  string
1681228247.200431       10.10.0.100     50441   10.10.42.252    2404    -       OFF
1681228247.462364       10.10.0.100     50441   10.10.42.252    2404    -       OFF
1681228247.488081       10.10.0.100     50441   10.10.42.252    2404    -       OFF
1681228429.168951       10.10.0.100     50441   10.10.42.252    2404    -       ON
1681228519.568603       10.10.0.240     57065   10.10.42.252    2404    OFF     -
1681228519.762912       10.10.0.240     57065   10.10.42.252    2404    OFF     -
1681228519.785780       10.10.0.240     57065   10.10.42.252    2404    OFF     -
1681228551.133613       10.10.0.100     50952   10.10.42.252    2404    -       ON
#close  2023-04-12-13-26-25
```

Figure 16. *I104.log*

when analyzing the same three packets using Wireshark, see Figure 17. Note that the highlighted packet seen in Figure 17 is not seen in the *I104.log* file because the M_DP_TB_-1 information object is not implemented yet. The Zeek script file that the prototype uses is configured to recognize single and double command information objects (C_DC_NA_1 and C_SC_NA_1) in the ASDU part of the IEC 104 packet.

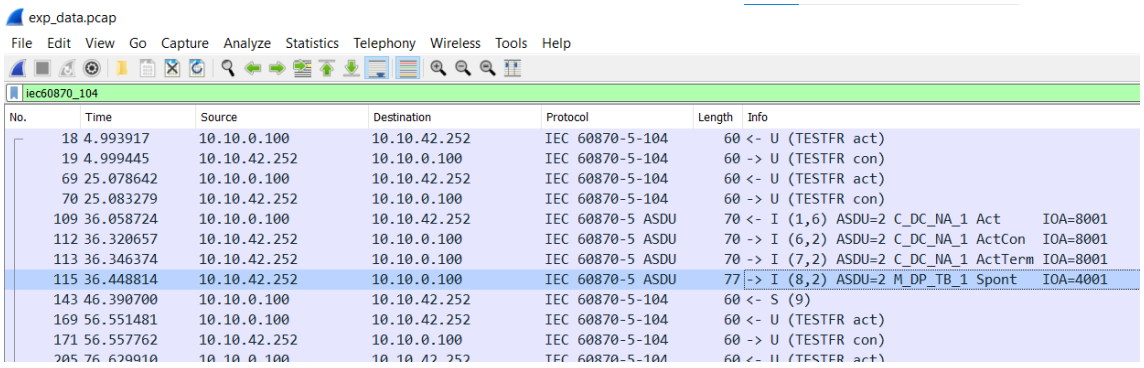

Figure 17. *OFF commands in Wireshark*

Most importantly, one can see the change in source IP address seen in Figure 16 in the Appendix, which illustrates the prototype's capabilities to see connections originating from different IPs. This can be further leveraged when creating IP whitelists and developing another Industrial Control System (ICS) Operational Technology (OT) network protection layer.

Although not explicitly tested on network traffic generated in a production environment,

the rules created and tested during the prototype validation stage, in author's opinion, should be able to detect rogue IP addresses without any false positives.

When looking at the outputs produced by the solution from Chromnik et al. [25] (see *typeid_count_output.txt* and *lvl_isu_output.txt* in the Appendix) one can see some statistics such as counters for different ASDU object and IEC 104 packet types. This information could be leveraged in some way to aid an NIDS system but using it as-is provides no useful information from a security perspective. For example, without further interpretation and changes in the code, it is impossible to tell that a foreign IP has made a potentially malicious command. See Figure 18 for a better overview of the difference of features.

Recognition	Prototype	Chromnik et al. solution [25]
IP addresses	yes	no
Ports	yes	no
IEC 104 protocol	yes	yes
IEC 104 packet type (I/U/S)	partial (only looks at I)	yes
Single command	yes	yes
Double command	yes	yes
other ASDU information object types	partial (not implemented in the rule-set)	yes
contextual logging	yes (Fig. 14)	no
statistical overview of traffic	no	yes (*typeid_count_output.txt* and *lvl_isu_output.txt* in Appendix)
Possibility to integrate into a NIDS	yes	partial (substantial amount of modifications needed)

Figure 18. *Results produced from analyzing traffic generated in the Experiment*

The current prototype can be seen as more of a proof-of-concept work than actual NIDS suitable for a production environment. As touched on Chapter 3.3, the prototype needs to be further developed to include more information object types such as the C_IC_NA_1 - interrogation command as well as not only recognize and log the potentially malicious traffic, but also act upon seeing it. This could be accomplished using Zeek plugins, such as the NetControl framework. Although the prototype is not yet ready to be placed into a production environment, majority of the foundational work has been completed, providing a solid basis upon which suitable plugins can be implemented on.

6. Summary

The work done during this research centered around the IEC 60870-5-104 (IEC 104) protocol and aimed to further open-source Network Intrusion Detection System (NIDS) development by analyzing the IEC 104 traffic and existing NIDS solutions with an ultimate goal of developing and testing a working NIDS prototype capable of dissecting IEC 104 traffic to filter Operational Technology (OT) network security related events. The work was successful at creating a NIDS prototype based on the IEC 104 protocol and its source code is published on GitHub [40]. The prototype can be adapted and used at Industrial and Automation Control Systems (IACS) environments where the IEC 104 protocol is prominent.

6.1 Evaluating the hypothesis

The hypothesis put forward in Chapter 1.2 was the following:

The experimental analysis of malicious IEC-60870-5-104 traffic, conducted using a Network Intrusion Detection System prototype based on IEC-60870-5-104 Single and Double command detection, will be able to detect potentially harmful traffic.

The hypothesis proved to be true since the developed prototype was able to detect potentially harmful OT network traffic by looking at Single and Double command injections. This was proven by experimentation utilizing a NIDS prototype.

6.2 Answering Research questions put forward in Chapter 1.3

1. *What solutions, insights, and suggestions have been explored regarding IACS NIDS that look at the IEC 104 protocol?*
   After conducting a review of relevant literature, it was apparent, that there are some

solutions that concentrate on the IEC 104 protocol that could potentially be used as NIDS but they would require substantial modifications and would not be suitable for production environments. Taking that into account, this research concentrated on developing the prototype using a versatile, industry-standard tool - Zeek. Sub-questions:

(a) *What IEC 104 NIDS and rule sets are available for Industrial Control System (ICS) systems?*

While there was a substantial quantity of research done on NIDS that are based on the IEC 104 protocol with seemingly good results, there was rarely any openly available code that could be tested or verified. The research available for testing/validation was limited to work done by Chromnik et al. [25], which was an advancement of previous work done by Udd et al. [24]. Chromnik et al. had made a parser for IEC 104 but did not create a sufficient rule-set that could be used in an NIDS.

(b) *How well do the current open-source NIDS solutions support the IEC 104 protocol and what could be done to improve their capabilities?*

The most prominent open-source NIDS solutions include Snort [46], Suricata [47], and Zeek [23]. Zeek is the only solution of the three to have some support for ICS protocols since it is designed for deep-packet inspection and, more importantly, its analysis. Zeek does not have built in support for the IEC 104 protocol but it has a plugin for protocol parser development - Spicy, which was used by Chromnik et al. [25] and in this research to allow for IEC 104 dissection so that Zeek can understand its network traffic. See Chapter 3.1 for more details regarding the inner workings of Zeek and Spicy.

See Chapter 2.1 for further background information and Chapter 3.1 for a more in-dept illustration of Zeek, Spicy, and their ability to analyze network traffic.

2. *What aspects of the IEC 104 protocol and its traffic need to be examined more closely to support open-source NIDS solutions?*

During the prototype development and its testing it became apparent that the answer to this question lies in the Application Service Data Unit (ASDU) part of the IEC 104 packet. More specifically, parts of the ASDU that were responsible for issuing commands, such as C_SC_NA_1 and C_DC_NA_1 Information Object codes. Com-

bined with the right connection context (IP addresses, timings) they provide valuable information for the NIDS. This proved to be true whilst experimenting with known and public exploits that were based on the IEC 104 protocol. See Chapter 4 and 5 for further details on experimentation and results.

Sub-questions:

(a) *What would be the most effective approach for detecting anomalies in traffic generated by the IEC 104 protocol?*

For this research it was determined that monitoring for unrecognized IP addresses that are issuing IEC 104 commands would be the best way to detect anomalies.

(b) *How can a deeper understanding of the IEC 104 protocol contribute to the development of more effective rules for open-source NIDS solutions?*

A better understanding of the IEC 104 protocol gave important context for rule-creation for the NIDS. Without this knowledge it would be impossible to create any meaningful NIDS rules.

(c) *Do the newly developed rules effectively detect known attacks in the context of the IEC 104?*

As discussed in Chapter 5, the developed prototype successfully sees and logs potentially malicious traffic generated by the IEC 104 protocol.

6.3 Future work

It is important to note that the development of the prototype is an ongoing process that will continue beyond the conclusion of this research. The findings and insights gained from this study serve as a foundation for future refinements and enhancements to the prototype. Future work includes the following aspects:

- Completing the ASDU types that are recognized by the prototype's dissector to include all 127 available types.
- Conducting research to identify and address specific requirements for implementing the NIDS in a production environment. This includes discussing the prototype with appropriate stakeholders, such as plant engineers, IACS architects, and others as

well as testing the findings in the CERT.LV IACS laboratory.

- Comparing the developed rule-based NIDS prototype with other methods of intrusion detection, such as NIDS that are based on anomalies, machine-learning, behaviour, and hybrid.

There are several directions for future research that could build upon the findings and insights gained from this work. Most importantly, researchers and developers can rely on the open-source GitHub code [40] and tailor it to their NIDS needs. The work referenced in [40] is created and released by me, Rudolfs Kelle.

References

[1] *CERT.LV - the Information Technology Security Incident Response Institution of the Republic of Latvia*. URL: https://www.cert.lv/en/about-us.

[2] *Protecting ICS Environments*. URL: https://resources.infosecinstitute.com/topic/intrusion-detection-and-prevention-for-ics-scada-environments/.

[3] Gerhard P. Hancke Brendan Galloway. "Introduction to Industrial Control Networks". In: *860 IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 15, NO. 2, SECOND QUARTER 2013*. 2013.

[4] Péter György and Tamás Holczer. In: *Attacking IEC 60870-5-104 Protocol* (Nov. 2020).

[5] Emmanouil Vasilomanolakis et al. "Multi-stage attack detection and signature generation with ICS Honeypots". In: *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium* (2016). DOI: 10.1109/noms.2016.7502992.

[6] Seonggwan Ahn, Thummin Lee, and Keecheon Kim. "A study on improving security of ICS through honeypot and ARP spoofing". In: *2019 International Conference on Information and Communication Technology Convergence (ICTC)* (2019). DOI: 10.1109/ictc46691.2019.8939925.

[7] Qais Saif Qassim et al. "Simulating command injection attacks on IEC 60870-5-104 protocol in SCADA system". In: *International Journal of Engineering & Technology* 7.2.14 (2018), p. 153. DOI: 10.14419/ijet.v7i2.14.12816.

[8] Mustafa Altaha et al. "An autoencoder-based network intrusion detection system for the SCADA system". In: *Journal of Communications* (2021), pp. 210–216. DOI: 10.12720/jcm.16.6.210-216.

[9] Petr Matoušek et al. "Flow based monitoring of ICS Communication in the smart grid". In: *Journal of Information Security and Applications* 54 (2020), p. 102535. DOI: 10.1016/j.jisa.2020.102535.

[10] Heng Chuan Tan, Carmen Cheh, and Binbin Chen. “Cotoru: Automatic generation of network intrusion detection rules from code”. In: *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications* (2022). DOI: 10.1109/infocom48880.2022.9796697.

[11] Kevin Wong et al. “Enhancing suricata intrusion detection system for cyber security in SCADA Networks”. In: *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)* (2017). DOI: 10.1109/ccece.2017.7946818.

[12] Petr Matoušek. “Description and analysis of IEC 104 Protocol”. In: (2017), p. 38. URL: https://www.fit.vut.cz/research/publication/11570.

[13] Dimitrios Pliatsios et al. “A survey on SCADA systems: Secure protocols, incidents, threats and Tactics”. In: *IEEE Communications Surveys & Tutorials* 22.3 (2020), pp. 1942–1976. DOI: 10.1109/comst.2020.2987688.

[14] Panagiotis Radoglou Grammatikis et al. “An anomaly detection mechanism for IEC 60870-5-104”. In: *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)* (2020). DOI: 10.1109/mocast49295.2020.9200285.

[15] Peter Maynard, Kieran McLaughlin, and Berthold Haberler. “Towards understanding man-in-the-middle attacks on IEC 60870-5-104 Scada Networks”. In: *Electronic Workshops in Computing* (2014). DOI: 10.14236/ewic/icscsr2014.5.

[16] Dinghua Wang and Dongqin Feng. “Intrusion detection model of SCADA using graphical features”. In: *2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)* (2018). DOI: 10.1109/iaeac.2018.8577543.

[17] *MATLAB*. URL: https://www.mathworks.com/products/matlab.html.

[18] Y. Yang et al. “Intrusion detection system for IEC 60870-5-104 based SCADA Networks”. In: *2013 IEEE Power & Energy Society General Meeting* (2013). DOI: 10.1109/pesmg.2013.6672100.

[19] John Hurley, Antonio Munoz, and Sakir Sezer. “ITACA: Flexible, scalable network analysis”. In: *2012 IEEE International Conference on Communications (ICC)* (2012). DOI: 10.1109/icc.2012.6363995.

[20] Y. Yang et al. “Stateful intrusion detection for IEC 60870-5-104 scada security”. In: *2014 IEEE PES General Meeting | Conference & Exposition* (2014). DOI: 10.1109/pesgm.2014.6939218.

[21] Rohith Raj S et al. “Scapy- A powerful interactive packet manipulation program”. In: *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)* (2018). DOI: 10.1109/icnews.2018.8903954.

[22] *Spicy - Robust Parsers for Protocols & File Formats*. URL: https://docs.zeek.org/projects/spicy/en/latest/index.html.

[23] Vern Paxson. “Bro: A system for detecting network intruders in real-time”. In: *Computer Networks* 31.23-24 (1999), pp. 2435–2463. DOI: 10.1016/s1389-1286(99)00112-7.

[24] Robert Udd et al. “Exploiting bro for intrusion detection in a SCADA system”. In: *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security* (2016). DOI: 10.1145/2899015.2899028.

[25] Justyna Chromik et al. “A parser for deep packet inspection of IEC-104: A practical solution for industrial applications”. In: *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks – Industry Track* (2019). DOI: 10.1109/dsn-industry.2019.00008.

[26] *Chromnik et al. open-source IEC 104 dissector*. URL: https://github.com/jjchromik/hilti-104.

[27] *C++ vs Python*. URL: https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-vs-.

[28] László Erdődi et al. “Attacking Power Grid Substations: An experiment demonstrating how to attack the SCADA protocol IEC 60870-5-104”. In: *Proceedings of the 17th International Conference on Availability, Reliability and Security* (2022). DOI: 10.1145/3538969.3544475.

[29] *IEC-104 rogue command injector*. URL: https://github.com/lockout/iec104inj.

[30] Tamás Holczer Péter György. "Attacking IEC 60870-5-104 Protocol". In: *Conference on Information Technology and Data Science* (2020).

[31] Panagiotis Radoglou-Grammatikis et al. "Attacking IEC-60870-5-104 scada systems". In: *2019 IEEE World Congress on Services (SERVICES)* (2019). DOI: 10.1109/services.2019.00022.

[32] *Cyberattack on Critical Infrastructure: Russia and the Ukrainian Power Grid Attacks*. URL: single%20transmission%20substation%20in%20northern%20Kiev%20lost%20power.

[33] *Industrial Cybersecurity for OT Environments*. URL: https://www.dragos.com.

[34] *ESET - Protecting digital progress with award-winning security*. URL: https://www.eset.com/int/.

[35] *CRASHOVERRIDE - Analysis of the Threat to Electric Grid Operations*. URL: https://www.dragos.com/wp-content/uploads/CrashOverride-01.pdf.

[36] *WIN32/INDUSTROYER - A new threat for industrial control systems*. URL: https://www.eset.com/fileadmin/ESET/INT/Landing/2017/black-hat/WIN32_Industroyer-USLetter-WEB.pdf.

[37] *IEC-60870-5-104 Specification*. URL: https://webstore.iec.ch/publication/25035.

[38] *Zeek Analyzers (Accessed 10/4/2023)*. URL: https://docs.zeek.org/en/master/script-reference/proto-analyzers.html.

[39] *Zeek Architecture (Accessed 11/4/2023)*. URL: https://docs.zeek.org/en/master/_images/architecture.png.

[40] *Author's GitHub*. URL: https://github.com/rk-lv/iec104.

[41] *Official Spicy docker image*. URL: https://github.com/zeek/spicy/blob/main/docker/Dockerfile.ubuntu-20.

[42] *Docker image used for development and experimentation, author's repository*. URL: https://hub.docker.com/r/rudy3213/iec104.

[43] *Inčukalns Underground Gas Storage Characteristics and History*. URL: https://www.conexus.lv/information-about-storage.

[44] *Purdue model for ICS*. URL: https://www.zscaler.com/resources/security-terms-glossary/what-is-purdue-model-ics-security.

[45] *Network Based Intrusion Detection System*. URL: https://www.sciencedirect.com/topics/computer-science/network-based-intrusion-detection-system.

[46] *Snort - Open Source Intrusion Prevention System (IPS)*. URL: https://www.snort.org.

[47] *Suricata - a high performance, open source network analysis and threat detection software*. URL: https://suricata.io.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Rudolfs Kelle

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Furthering Industrial Control System Intrusion Detection Systems’ advancement by analyzing the IEC 60870-5-104 protocol & its traffic ”, supervised by Bernhards Blumbergs
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons’ intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

15.05.2023

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student’s application for restriction on access to the graduation thesis that has been signed by the school’s dean, except in case of the university’s right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2

Contents of **typeid_count_output.txt**

```
2023-04-12_09:57:57 [DEBUG] m_dp_tb_counter: 3
2023-04-12_09:57:57 [DEBUG] m_ei_na_counter: 3
2023-04-12_09:57:57 [DEBUG] m_dp_na_counter: 2
2023-04-12_09:57:57 [DEBUG] c_sc_na_counter: 3
2023-04-12_09:57:57 [DEBUG] c_dc_na_counter: 1
2023-04-12_09:57:57 [DEBUG] c_ic_na_counter: 6
2023-04-12_09:57:57 [DEBUG] m_sp_na_counter: 4
2023-04-12_09:57:57 [DEBUG] Total time: 103.0 msecs 51.0 usecs
```

Contents of **lvl_isu_output.txt**

```
2023-04-12_09:54:16 [DEBUG L1] Total time: 98.0 msecs 454.0 usecs
2023-04-12_09:54:16 [DEBUG L1] mode_u_counter: 20
2023-04-12_09:54:16 [DEBUG L1] mode_s_counter: 3
2023-04-12_09:54:16 [DEBUG L1] mode_i_counter: 22
```

Contents of **I104.log**

```
#separator \x09
#set_separator  ,
#empty_field    (empty)
#unset_field    -
#path   I104
#open   2023-04-12-13-26-25
#fields ts      id.orig_h       id.orig_p       id.resp_h
   id.resp_p       command_single  command_double
#types  time    addr    port    addr    port    string  string
1681228247.200431       10.10.0.100     50441   10.10.42.252
   2404    -       OFF
```

```
1681228247.462364         10.10.0.100     50441   10.10.42.252
    2404      -         OFF
1681228247.488081         10.10.0.100     50441   10.10.42.252
    2404      -         OFF
1681228429.168951         10.10.0.100     50441   10.10.42.252
    2404      -         ON
1681228519.568603         10.10.0.240     57065   10.10.42.252
    2404      OFF       -
1681228519.762912         10.10.0.240     57065   10.10.42.252
    2404      OFF       -
1681228519.785780         10.10.0.240     57065   10.10.42.252
    2404      OFF       -
1681228551.133613         10.10.0.100     50952   10.10.42.252
    2404      -         ON
#close   2023-04-12-13-26-25
```