MASTER THESIS
UNIVERSITY OF TWENTE
FACULTY OF ELECTRICAL ENGINEERING, MATHEMATICS AND COMPUTER SCIENCE

Multi-domain Cyber-attack Detection in Industrial Control Systems

Jan-Paul Konijn

September 2022

Supervisors:

Prof.dr.ir. R.M. (Roland) van Rijswijk-Deij (UTwente)
Prof.dr. P.J.M. (Paul) Havinga (UTwente)
Dr.ir. H.B. (Erik) Meeuwissen (TNO)
Ir. I. (Irina) Chiscop (TNO)

TNO innovation for life

Abstract

Industrial control systems are everywhere and form the backbone of many critical infrastructures, directly affecting many aspects of life. Advanced technical requirements and increased connectivity have further converged the IT and the OT domains, thereby opening up new attack areas and creating new cyber risks. Given the importance of such systems, it is essential to ensure their security. Hence, this study proposes a method of integrating alert data of the physical domain and the network domain found in multi-domain industrial control system architectures to extract the strategy of an adversary performing a multi-stage attack, an approach that has only been done to a limited extent in the literature. First, we conduct a dataset survey of publicly available datasets and a digital twin for ICS. We find that the *SWaT A6 2019* dataset is the most suitable since the dataset contains both raw network and physical sensor data under a series of attack steps. Then, three novel industrial control system detector types are introduced at different positions in the Purdue Model. The results on the *SWaT A6 2019* dataset demonstrate that the detectors are able to identify all the attack events, with the largest false positive rate being 0.063. Finally, we use the alert output of our novel ICS attack detectors and the mock-up physical sensor detector as input for our proposed merging detector alerts method. This integration allows us to extract the attack strategy applied by a multi-stage attacker in an ICS environment. The results demonstrate the potential of this method to reduce the number of false positives by 81% on the *SWaT A6 2019* dataset while still providing insight into the methodology used by the multi-stage attacker.

Acknowledgements

This thesis has been an intensive effort of more than half a year, where I developed myself further to become an expert in the field of cyber-security, specialising in Operational Technology and the Internet of Things. It was a phase in which I not only learned to set up valid and reliable scientific research but also to critically assess, improve, evaluate, and, most importantly, appreciate my work. This development and progress could not have succeeded without help, critical feedback, and the support of many. Therefore, I would like to thank the individuals who provided great wisdom, help and support during my research. First, I would like to express a special thanks to my TNO supervisors: Dr.ir. H.B. (Erik) Meeuwissen and Ir. I. (Irina) Chiscop for supporting and facilitating this research. You have been of great value to this research by giving critical, well-grounded and thorough feedback. Second, a special thanks to Prof.dr.ir. R.M. (Roland) van Rijswijk-Deij for providing the opportunity to perform my research at TNO, as well as for the constructive feedback sessions, meetings and being a great mentor. Third, I would like to thank Prof.dr. P.J.M. (Paul) Havinga for taking part as a critical reviewer. Next, a special thanks to all my friends and family for their feedback, trust, and great support. Finally, I would like to thank the iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design, for providing the *SWaT* dataset used in this thesis. Their dataset and their responses to questions have been of significant contribution to this research.

Jan-Paul Konijn
Alkmaar, September 2022

Glossary

ICS: *Industrial Control Systems*

IT: *Information Technology*

OT: *Operational Technology*

IDS *Intrustion Detection System*

CPS *Cyber Physical Systems*

SCADA: *Supervisory Control and Data Acquisition*

DCS: *Distributed Control System*

HMI: *Human-Machine Interface*

PLC: *Programmable Logic Controller*

MITM: *Man-in-the-middle*

DPI: *Deep-packet Inspection*

PCAP: *Packet Capture*

SWaT: *Secure Water Treatment*

ML: *Machine Learning*

ARIMA: *Autoregressive integrated moving average*

DHALSIM: *Digital HydrAuLic SIMulator*

ENIP: *Ethernet industrial protocol*

CIP: *Common Industrial Protocol*

SARIMA: *Seasonal Arima*

ECOD: *Empirical-Cumulative-distribution-based Outlier Detection*

Contents

1 Introduction

Industrial Control Systems (ICS) are the backbone of critical infrastructure in various domains such as energy, water, and manufacturing. Sophisticated cyberattacks increasingly target these critical infrastructures due to their importance to society. A successful cyberattack on ICS can significantly impact organisations, national safety, and stability. A typical ICS architecture encompasses many different types of systems, such as supervisory control and data acquisition (SCADA) systems and distributed control systems (DCSs)[1, 2]. By comparing ICS architectures such as the abstracted Purdue model representing a multi-layered ICS architecture depicted in Fig. 1 with conventional IT systems, it can be noted that ICS systems are more closely tied to the physical world and, therefore, can also impact physical processes. The broadly used Purdue model presents a reference model for an ICS architecture, where we generally can identify two sub-domains or macro-areas, Information Technology (IT) or traditional information systems, and Operational Technology (OT), which is commonly applied in cyber-physical systems (CPS). OT is used to manage physical manufacturing processes, process events, assets, and actuation through monitoring, sensing, and the control of physical devices, while conventional IT is more focused on business logistics and other IT systems[3, 4]. In the context of IT/OT in ICS, the goal between the two domains is also inherently different. IT is primarily focusing on *Confidentiality, Integrity, and Availability* and in the OT domain *Safety, Reliability, Availability, and Maintainability* are of paramount importance [5, 6].

The number of cyberattacks against ICS infrastructure has increased worldwide in recent years. The advanced ransomware attack on the Colonial Pipeline [7], the Triton Malware [8], targeting Safe Instrument Systems in industrial control architectures, and the 2015 attack on the Ukrainian power grid rendering 225,000 people without power [9, 10]. Finally, the famous Stuxnet attack on an Iranian enrichment facility [11] that will be discussed later in this thesis.

IT systems and ICS systems used to be different as ICS were often isolated or air-gapped, running on dedicated and often proprietary hard- and software. The further adoption of IT solutions in ICS to promote corporate connectivity and remote access capabilities leads to a convergence of the two domains, often driven by the demand for new technological needs such as artificial intelligence, remote operations, reporting, and automation. This convergence and interconnectivity of modern complex systems such as CPS also give raise to an increased attack surface [3, 12, 6]. Patching vulnerabilities and the early detection of intruders enables organisations to increase their security and mitigate such concerns. Nevertheless, patching in environments with operational constraints, legacy hard- and software, and proprietary communication protocols are often not achievable, raising the question of whether we can improve the defensive measures applied in critical ICS infrastructures [2].

The next successful attack is not a concern of ”if” but rather ”when”. This means there is a dire need to develop novel security approaches to prevent, detect, understand, or mitigate cyber-physical attacks. The challenge associated with implementing detection mechanisms for ICS is the testing hereof, which may interfere with real physical processes and disrupt the normal operations of the system at hand [4]. Monti et al. [4] also indicate that public ICS datasets are scarce, and organisations are reluctant to share datasets of their ICS architectures as this can reveal a source of intelligence for potential adversaries or can reveal proprietary business information. A security approach to increase the defensive capabilities in ICS can be the adoption of Intrusion Detection systems or attack detection methods. Intrusion Detection Systems in the ICS domain offer an effective way to detect malicious behaviour or behaviour that compromises security policies. Although intrusion detection is already well established in the IT domain, intrusion detection and alert correlation research in OT environments are, in comparison, relatively new research areas.

This research aims to improve the effectiveness of multi-stage attack detection in ICS architectures by integrating physical sensor data and network data. Individual physical sensor-based detection introduced by many in the literature provides only limited attack insight for ICS architectures. For example, physical domain alerts do not necessarily indicate a cyberattack, e.g., a sensor can be faulty, or it may be affected by environmental influences rendering it hard to identify the source of the cyberattacks. The integration of multi-domain data can improve the detection process by allowing for tracing the root of a multi-stage attack or reducing the false positives [13, 14]. However, this only has been done to a limited extent before. As such, this works looks at: 1. digital Twin for

ICS data acquisition, 2. suitable datasets for developing ICS attack detection systems, 3. proposes two novel network-based attack detection mechanisms, 4. introduces an ICS alert correlation method for attack strategy extraction and false positive reduction using alert data.

1.1 Research Questions

We have formulated one main research question given below. We aim to answer the main research question through a set of sub-questions.

How to combine network data with physical sensor data to achieve full attack visibility of an advanced cyber-attack against ICS architectures?

Where we define ***full attack visibility*** as all the anomalous behaviour caused by an ICS cyber-attack and ***full visibility*** as being informed of all the anomalous behaviour in an ICS architecture.

To answer the main research questions, we have defined the following sub-questions:

- **i** *What is the current state-of-art on cyber-attack detection in ICS systems?*
- **ii** *What public datasets are available for the development of a network and physical sensor-based attack detection method for ICS architectures?*
- **iii** *How to achieve full visibility in an ICS architecture?*
  - (a) *At what position in the Purdue model should each (sub)-detector be placed?*
  - (b) *What (sub)-detectors are required to obtain full visibility?*
  - (c) *What features are required for each (sub)-detector?*
- **iv** *How to achieve full attack visibility characterised by all the observed anomalous behaviour?*

The research is structured as follows: in Section 2 we offer the reader general background information required to introduce the topics discussed in the Related works in Section 3. We give in Section 4 an overview of the intended methodology of this thesis. Furthermore, in this section a more detail overview of of the section layout can be found with the help of a visual overview. Next, in Section 5 we perform a survey of relevant ICS datasets. Section 6 presents a detailed analysis of the selected datasets and the most relevant features explored. In the Sections 7, 8 we introduce the physical and network based detectors to evaluate in Section. 8. This is followed by Section 10 where we integrate the output of the physical and network-based detector. The merging method is evaluated in Section 11. Finally, in Section 12 we discuss our findings, present limitations and share some thoughts for the future work.

2 Background

In this chapter, we will introduce the reader to background notions on various topics and concepts required to comprehend the details found in this thesis. Each topic will be briefly introduced to the level required. First, we discuss a common reference architecture model for ICS systems and describe the different components/devices herein. Then, we introduce the Mitre ATT&CK (ICS) framework, which categorises ICS cyber attacks. Alternatively, we introduce the Spiral Attack Model, describing ICS cyber attacks in a different way. Afterwards, we would like to give a brief overview of past ICS cyber attacks based on the literature. Then, since the intention is to implement a new ICS attack detection mechanism, we will discuss the definition of Intrusion Detection Systems. In the last three sub-sections, we discuss a novel method for data acquisition and different network traffic characterisation formats.

2.1 Purdue Model

Fig. 1 indicates the Purdue Enterprise model used in this thesis for the reference of an enterprise ICS architecture separating the network into different levels and zones [15]. Furthermore, the Purdue model allows for describing the scope of data, positions in an architecture and gives an intuitive overview. The ICS representation introduces the concept of zones to split the network into smaller networks where consistent security controls can be installed or, in other words, a zone is a segmented networking environment with a well-defined perimeter [4, 15]. The Purdue model used throughout this report provides an abstraction for an ICS architecture, which is useful when considering the broad diversity of ICSs [16]. For instance, an ICS deployment encountered in a power grid might be distinct from ICS systems found in water distribution systems. The Purdue model provides a generalisation with the lower parts depicting the OT section and the upper portion showing the IT area of the infrastructure. IT is a broad term encompassing services, equipment, or interconnected system(s) used for automatic acquisition, storage, manipulation, management, control, display, and transmission of data or information. In contrast, OT consists of a broad range of programmable systems or devices that interact with the physical environment or manage devices that interact with the physical domain. These devices and systems can control, monitor, or interact with physical processes and can cause or detect change [17]. The paper will focus on level 0 to level 3 depicted in Fig. 1. The different levels of the manufacturing zone in the model are as following [18, 19, 20]:

- Level 3 - Operations Management: This layer includes manufacturing operating systems control plant operations to produce the desired end product.
- Level 2 - Supervisory control equipment: This layer includes control room workstations and Human Machine Interfaces (HMI).
- Level 1 - Control Equipment: Includes intelligent devices such as programmable logic controllers (PLC) and Remote Terminal Units (RTU)
- Level 0 - Equipment under control: The physical sensors, actuators, and other hardware that directly interacts with the physical process.

The different zones can be defined as safety zones that comprise components used to ensure safe operation: the manufacturing zone includes components used to monitor, control, and automate physical processes. The architecture is divided into separate logical segments with different requirements.

2.2 Industrial Control Systems

Industrial Control Systems (ICS) are systems that control industrial processes such as manufacturing, product handling, production, and distribution. ICS is a general term that encompasses several types of control systems, including supervisory control and data acquisition (SCADA) systems used to control geographically dispersed assets, as well as distributed control systems (DCS) and smaller control systems using PLCs to control localised processes [21, 6]. ICSs consist of interconnected systems, sub-systems, interdependent computing devices, mechanical hardware, sensors, and other electrical devices to mainly perform three primary operations: acquisition, control, and supervision depending on the field of application [1, 19]. Industrial control systems have inherently

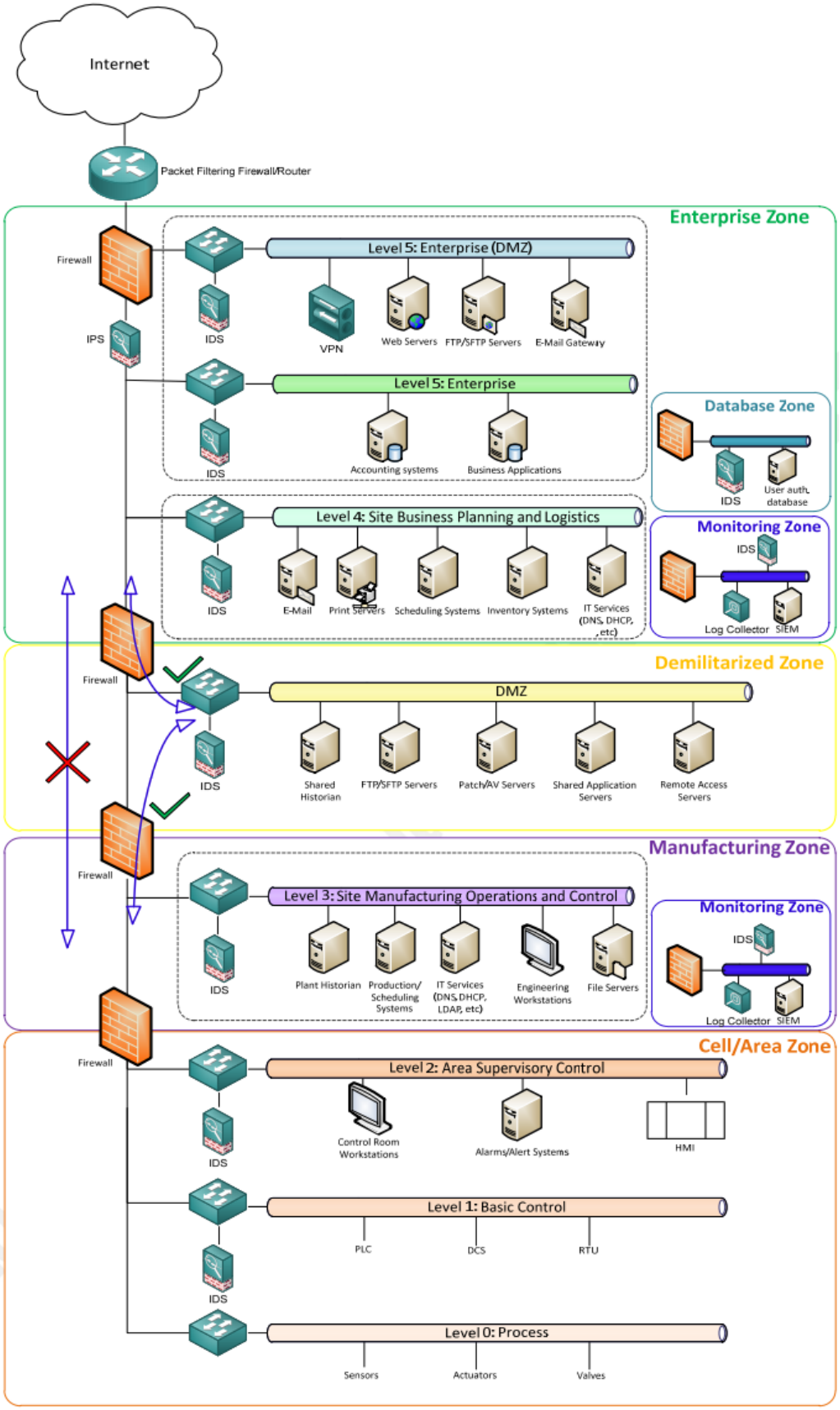


Figure 1: Purdue model for ICS architectures - model for control hierarchy [15].

different security requirements compared to traditional IT systems. Hu et al. [2] give a detailed summary of security requirements for ICS:

- **i** *Limited Computational Resources*: ICS often consists of a wide variety of devices, including sensors and actuators with limited computational power. Therefore, running or integrating additional security software, programs or measures can be challenging or even impossible.
- **ii** *Legacy Systems:* ICS hardware is often costly to replace, and replacement will disrupt the continuity of physical processes. Therefore, ICS systems just run for many years without soft- or hardware updates, allowing for the easy hijacking of systems.
- **iii** *Poor security of industrial protocols:* ICS systems were initially designed to operate air-gapped. As a consequence, many of the currently used ICS protocols do not include advanced security features.
- **iv** *Hard updating and restarting ICS hardware:* Due to the point discussed in ***ii***, updating ICS hardware is difficult. Moreover, in ICS, where the continuity of the system is important, deployments cannot easily be halted for an update restart or bug fix.
- **v** *Real-time:* ICS operations often perform real-time and time-critical operations where deviations can lead to damage or failure if delays and jitter exceed acceptable levels.
- **vi** *Fixed business logic:* ICS should follow specific business logic where failure could lead to business incidents.

From a high-level approach, ICS can be sub-divided into the following systems/components [18]:

2.2.1 Safety Instrumented Systems (SIS) and Protection Systems

This is the next safety layer after the basic process and control system. An SIS is an autonomous control layer ensuring that operational process parameters stay within boundaries or do not exceed levels that lead to an unstable system by taking a process back to a safe state, protecting personnel, the environment, and equipment [22, 18]. Unfortunately, failure of these systems can remain hidden and often does not have a visible effect, thus leaving the option that these systems cannot react when certain failing conditions are presented, compromising safety. One example of malware targeting the SIS is the Triton malware changing the firmware with a malicious version to change the control logic [23].

2.2.2 Basic Process Control Systems

This high-level layer is responsible for taking sensor inputs and process instruments to provide output-based control functionality according to the intended control design [18]. The functioning of this system includes alarm/event logging and control processes according to defined conditions. According to Alexander et al. [18], failures of the Basic Process Control system are ”self-revealing”, can happen promptly, and are noticeable in the underlying process.

2.2.3 Engineering and Maintenance Systems

These systems are used to maintain, diagnose, and configure the systems discussed previously [18]. Often these are vendor-supplied tools and software for maintenance and engineering, which requires console access or network/direct connection to the equipment.

2.2.4 ICS Components

ICS architectures are often diverse and can exist out of many heterogeneous devices and protocols. Therefore, based on the literature, we identify and address the most important and common ICS components and protocols.

SCADA Systems: SCADA systems can be seen as a higher level of the ICS architecture responsible for controlling, data gathering, and managing distributed assets [6, 19, 4]. The SCADA server connects to SCADA clients

managing communication between different types of devices, collecting and transferring information to a centralised computer to display and visualise operational information. This allows facility operators to monitor and control remote or local operations in near real-time. SCADA systems function in a supervisory fashion, rather than performing control [24]. Communication can be accomplished through various types of channels, including wireless or radio, telephone line, cable, or satellite communication controlling Remote Terminal Units or PLCs. The primary purpose of this communication is data acquisition and presentation through Human-Machine Interface (HMI) [24].

DCS Systems: Are comparable to SCADA but have a few subtle differences. Like SCADA, it is a software package that communicates with control hardware, presenting information to a centralised HMI [24]. Furthermore, DCSs are often used to control production systems within the same geographic location. DCSs are integrated as a control architecture containing a supervisory or high-level controller managing multiple integrated subsystems to controller a localised process. A DCS applies a centralised supervisory control loop to mediate a group of localised controllers that share the overall tasks of carrying out an entire production process [6].

The motivation behind this architecture is to modularise the process reducing a single point of failure [19, 6]. The primary difference between SCADA and DCS is that DCS is process-driven rather than event-driven [24]. For this reason, DCS can present a steady stream of process information. Furthermore, in DCS, a much higher level of interconnection between the control hardware and software layer is present, resulting in more reliable communication between devices removing the necessity to focus on the loss of data during transfer, unlike SCADA, where the software needs to account for unreliable communication media. SCADAs and DCSs have similar architectures at a higher level and have similar technologies. Likewise, DCSs are also implemented using computers that communicate with the plant equipment either through a communication bus or directly. Finally, DCSs are similarly affected by changes in the IT landscape and have similar security requirements to SCADA [24].

Programmable Logic Controller: is a small industrial computer to perform logical function execution using electrical hardware. PLCs have the capability to control complex processes and can be widely found in both DCS and SCADA systems [6]. Additionally, PLCs can be found in all industrial processes solving complex logic process function and communications [19]. PLCs can be viewed as the boundary between the OT network and the physical processes [4, 25].

Furthermore, ICS hardware, including PLCs, is designed for 10-15 years of continuous operation, contrary to conventional IT hardware, which often has a shorter lifespan. According to Monti et al. [4] modern PLCs can also use UNIX-derived micro-kernel and present a built-in Web interface increasing the IT/OT convergence discussed earlier and introducing novel attack surfaces [25].

Human Machine Interface: is dedicated hard- or software allowing operators to control monitor and modify operational settings and processes. HMI can interface directly with SCADA servers or the control network allowing it to monitor a multitude of devices, networks, and processes [25].

ICS Field devices: are all the devices that directly interact with the physical processes, including actuators, transducers, sensors, and various types of machinery [25]. For example, sensors allow for measuring multiple parameters such as sound, pressure, vibration, voltage, and current. Controllers such as the PLCs mentioned earlier can utilise ICS Field devices to obtain information or control actuators to interact with the physical process [4]. Interfacing between the field device and the controller can be done digitally or via an analogue I/O module.

Remote Terminal Unit: The last component of ICS that we want to address for the scope of this research is the remote terminal unit (RTU). RTUs can be found at remote sites interacting with field devices [25]. Similarly to PLCs, RTUs are at the boundary between the real world and the cyber domain. RTUs can also control and obtain information digitally or via an analogue I/O module like PLCs. However, RTUs generally use different communication mediums utilising various types of Wide Area Network technologies such as cellular, satellite, or modern short-range technologies equivalent to WirelessHART, Zigbee, or IEEE 802.15.4 [26].

2.3 Mitre ATT&CK ICS Framework

The Mitre ATT&CK framework for ICS provides an aggregated database of cyber adversarial tactics, techniques, and procedures in the ICS domain [18]. The Mitre ATT&CK ICS framework aims to bridge the gap between cybersecurity and operational engineers, increasing the understanding from both perspectives supporting better security decision-making [18]. Therefore, describing ICS attacks according to this framework gives a certain level of standardisation or abstraction and is used throughout this thesis. The Mitre ATT&CK ICS applies the ATT&CK methodology to the ICS domain, covering levels 0-2 of the Purdue model. In some cases, the technology domains spill over to level 3 of the Purdue model if level 3 provides aid to the lower levels. For instance, there is a certain level of overlap when Linux or Windows devices serve support for control hardware in level 1. For all the tactics, techniques, and procedures, we refer to the overview[1].

2.4 Spiral Attack Model in ICS

An alternative approach to the Mitre ATTCK ICS framework is the Spiral Attack Model proposed by Hassanzadeh and Burkett [27], which considers the totality of a converged IT and OT architecture as opposed to domain-specific attack models as seen in the separate Mitre ATT&CK ICS and Mitre ATT&CK framework. The spiral model implements the previously seen Purdue model and considers multi-step, multi-domain attacks with a recurring set of attack phases to better visualise and classify ICS attacks.

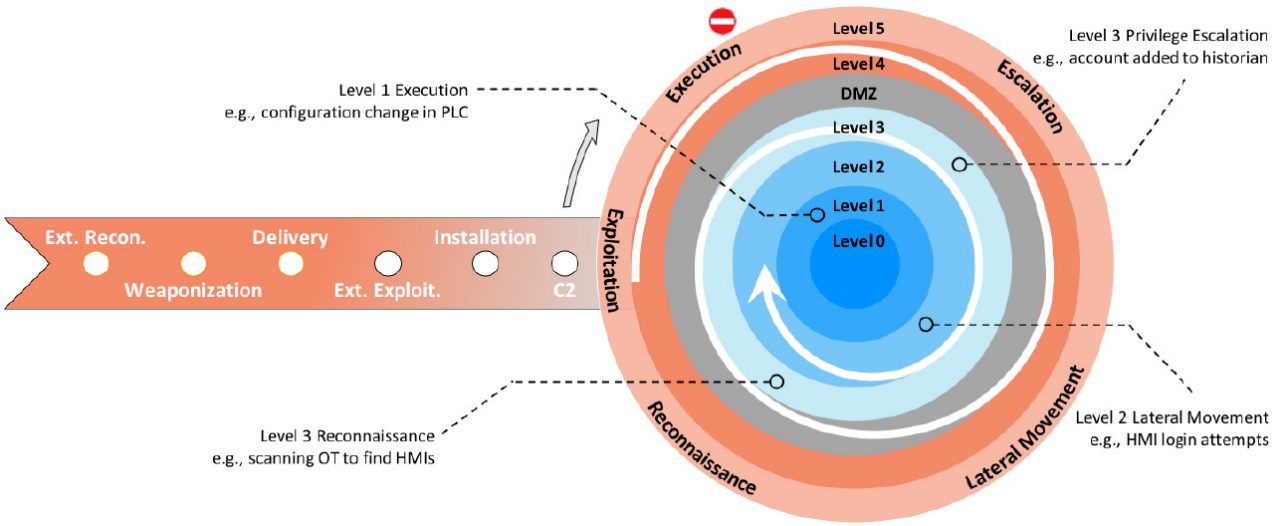


Figure 2: Spiral attack model by Hassanzadeh and Burkett [27].

Fig. 2 depicts the model introduced by Hassanzadeh and Burkett [27], which consists of two parts. The linear part proves the steps necessary to achieve the initial exploration somewhere in the Purdue model, assuming this can be in an IT zone or lower in the OT zone, similar to the case of Stuxnet, which we will address later. The spiral part of Fig. 2 represents steps after an initial ICS exploration where an attacker moves laterally through a Purdue level or moves vertically. The white arrow indicates how an attacker can move between the different levels of an ICS architecture.

2.5 Cyberattacks on ICS

In the last decade, we can observe an increase in the complexity of cyberattacks targeting ICS architectures. Similarly, we can observe that there is not only an increase in complexity but also a significant shift in the types of threats (see Fig. 3). It used to be that attacks were primarily being performed by insiders, employees, or individual hackers. However, in recent years, attacks have been increasingly being executed by well-funded and organised threat actors, such as Nation-states or organised cyber criminals [11].

[1] https://collaborate.mitre.org/attackics/index.php/Main_page

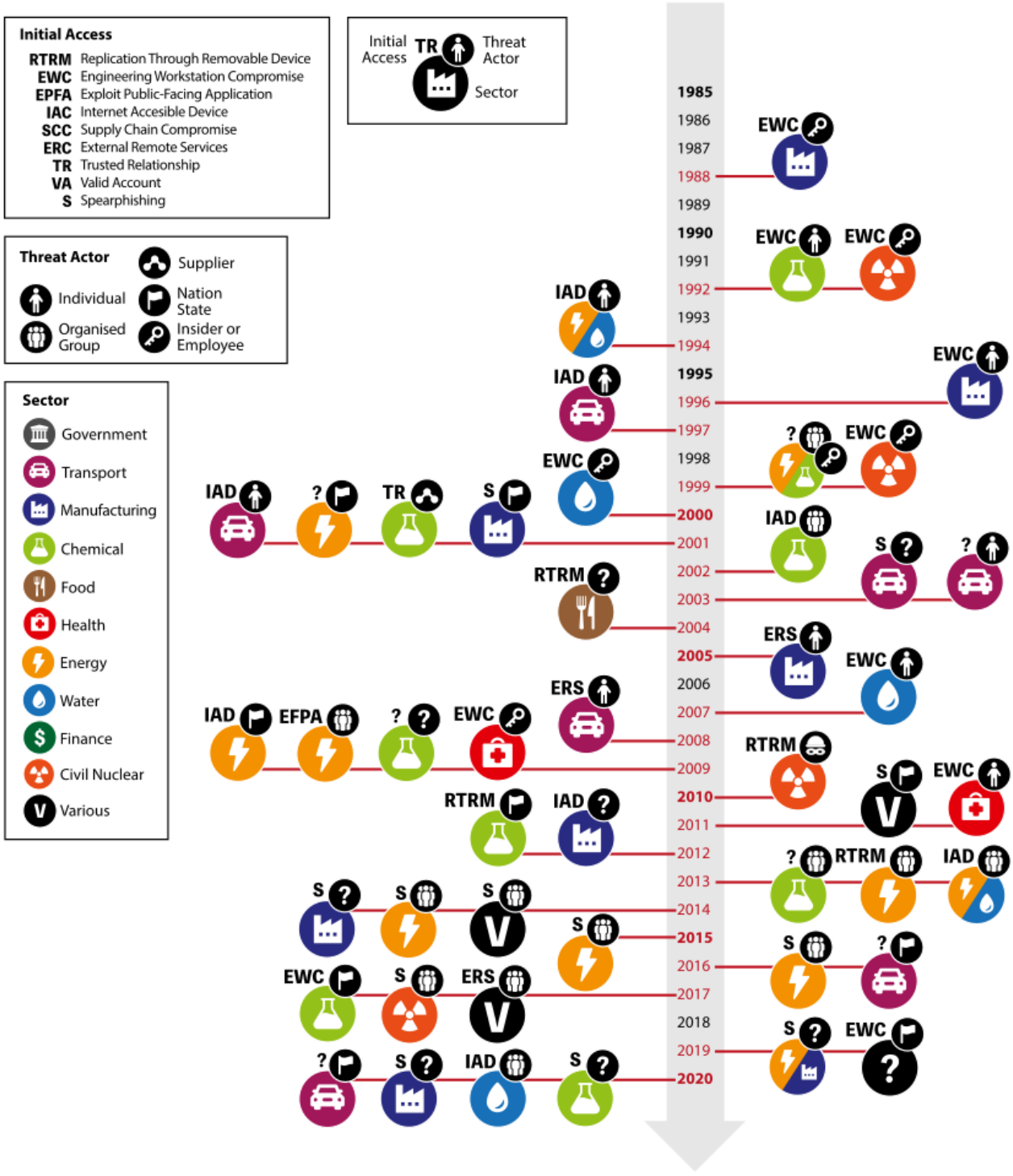


Figure 3: Timeline of Attacks against ICS [11].

One attack suspected to have been carried out by a nation-state or organised crime group is the recent attack on the Ukrainian power grid rendering a large part of critical power infrastructure affected [11]. The attack heavily relied on the BlackEnergy Malware, with the first version of the malware being identified in 2007. Due to the sophistication of the attack, three energy distribution companies were affected, resulting in a power outage of six hours over a large geographical area [11]. According to a report by the US Department of Home Land Security CISA, the attack was synchronised and coordinated following an extensive reconnaissance phase [28]. The attack occurred within 30 minutes targeting regional and central power facilities. The attackers opened the circuit breaker, after which they utilised the KillDisk Malware. The malware rendered the affected system unusable by removing system files and corrupting the master boot sequence, resulting in significant recovery time. Furthermore, it was reported that the adversaries corrupted Serial-to-Ethernet devices at substations and overwrote human-machine interfaces to make a recovery from the incident challenging.

The Triton malware was first detected in 2017 and still remains a threat to this day [22, 11, 29, 23]. The malware specifically targets the Safety Instrumented System (discussed in Section 2.2), changing the firmware of the control element, allowing to manipulate SIS controller into a failed safe state, a state that can automatically shutdown industrial processes [11]. The interruption of the SIS can lead to hazardous events as it is the system's last line of defence when failing parameters can exceed the bounds of the system required for safe or stable operation.

Stuxnet is probably the most widely known attack targeting ICS [11]. The malware exploited different vulnerabilities of multiple systems that can be found in ICS architectures. The malware of high complexity first targeted conventional IT systems to traverse to lower levels of the Purdue model. The primary goal, according to experts, was to cause physical damage to the Iranian nuclear program by changing sensor readings [11].

The aforementioned attacks are only a subset of the total attacks on critical infrastructures for a more elaborated overview, we refer to the paper by Makrakis et al. [30]. Moreover, due to the further expected integration of the Internet of Things and other connected systems, it is anticipated further to open critical infrastructure to remote cyber threats such as the earlier observed Triton malware and BlackEnergy malware [31].

2.6 Intrusion Detection Systems

Intrusion detection is the process of monitoring events in a network or computer system and analysing them for signs of possible incidents, which are violations or imminent threats of computer security policies, acceptable use policies, or standard security practices [2, 32]. Generally, two attack approaches apply to cyber-physical systems like industrial control systems. First, the attack model encompasses long- and short-duration attacks, long-duration where adversaries might take an established period to spread through networks. In contrast, short-time adversaries might aim to cause quick damage [33]. Below, we discuss a classification of intrusion detection systems (IDS) and give an overview of different performance evaluation metrics often used in the literature.

2.6.1 Detection Techniques

Fig. 4 provides an general overview of the classification of intrusion detection systems in ICS [33]. Below we discuss the different intrusion detection abstractions.

Knowledge-Based Intrusion Detection: Attack or intrusion detection based on features that match a specific pattern. The authors suggest this category has a low false-positive rate as it will only detect previously known attacks or intrusions [33]. However, the disadvantage is that attacks are required to be known. Therefore, to be effective against new attacks, it needs to be updated with the latest attack information.

Behaviour-based Intrusion Detection: Behaviour-based intrusion detection is a variant of intrusion detection that searches for deviations in the run-time features, often based on historical information [33]. Detection methods can include clustering based on unsupervised learning or Bayesian classifiers, eliminating the requirement to know attack behaviour or keep an attack information database [33].

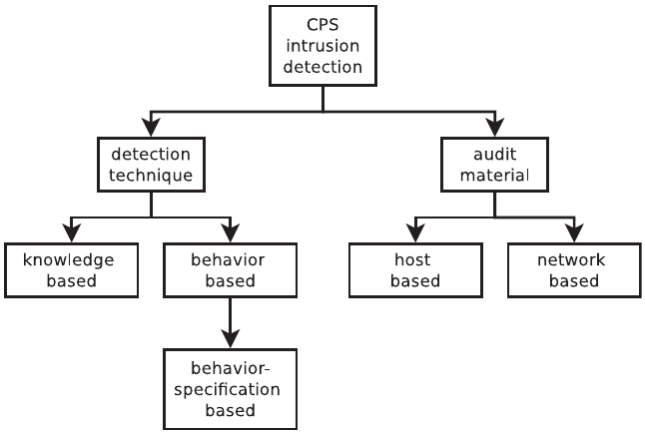


Figure 4: Classification for Intrusion detection Techniques for CPSs [33].

Behaviour-Specification–Based Intrusion Detection: The last type of detection technique addressed by Mitchell and Chen [33] is Behaviour-specification-based intrusion detection, an approach that relies on providing a formal definition of the normal system behaviour and identifying devices deviating from it. When a bad node disrupts the formal specification of the system, an intrusion can be detected. In other words, the IDS measures a node's deviating misbehaviour from the formal specification, allowing for lightweight intrusion detection, according to the authors.

Mitchell and Chen [33] continue to argue that this method has one significant advantage being the fact that this gives a relatively low false-negative rate. Furthermore, Behaviour-specification-based intrusion detection systems have the property that they are immediately effective as no training phase is required. On the other hand, a disadvantage of this method is the requirement that one needs to specify a formal definition of the system describing behaviour. Examples of such intrusion detection systems can be physics-based attack detection which creates a model of the regular operation of the control systems to detect deviations between the model and the physical sensor values[34].

2.6.2 Audit Sources

Based on the literature [35, 36, 33, 20] we can generally identify two types of information sources or audit sources.

Host-Based Intrusion detection: Host-based intrusion detection systems are commonly used in ICT systems and less in ICS as not all ICS hardware components are suitable for host-based intrusion detection due to computational constraints. In ICT, host-based intrusion can be based on information such as login times, system reboot events, hardware access records, systems logs, or password failures [36]. However, this information is less obvious in ICS since not every device allows for logging and monitoring. Alternatively, an ICS approach can be to install a host-based IDS on a SCADA server that monitors the physical values of underlying processes.

Network-Based Intrusion detection: Network-based auditing uses network information to determine compromise. In a network, there can be several points where data can be collected from, including switches and routers [33, 20]. When considering network-based intrusion detection in ICS, we have to assume that the architecture protocols use a TCP/IP stack for communication. The advantage of applying network-based auditing is the scope of visibility. However, this can also pose a disadvantage as it is hard to get inter- and intracel visibility of activity [33].

2.6.3 Detection Performance Evaluation

Mitchell and Chen [33] outline three core metrics being used in IDS evaluation which can be calculated using Table 1 presenting a confusion matrix for prediction evaluation.

		Predicted: Attack	Predicted: Normal
Actual	Attack	True Positive (TP)	False Negative (FN)
	Normal	False Positive (FP)	True Negative (TN)

Table 1: Confusion matrix.

- False-Positive Rate (FPR), e.g., False-Alarm, should remain under controllable bounds as much as possible.

$$FPR = \frac{FP}{FP + TN} \tag{1}$$

- True-Positive Rate (TPR)

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

- False-Negative Rate (FNR)

$$FNR = \frac{FN}{TP + FN} \tag{3}$$

However, the above mentioned rates might not provide a sufficiently thorough performance evaluation [2, 37]. For example, one important point the authors address is the lack of detection latency used in research as a critical means to categorise IDS performance [33]. The time between an adversary exploiting the architecture and the detection of the adversary can be critical, especially when we consider critical industrial control systems such as water distribution systems.

Monti et al. [4] also consider the F_1 score and *accuracy* as measures to compare performance where the F_1 score (Eq. 4) represents the harmonic means between *precision* and *recall* where the precision and recall are given by Eq. 5 and 6.

$$F_1 \ score = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

2.7 DHALSIM Digital Twin

To develop the attack detection approach (elaborated in Section 4) we will build upon previous research by F. Murillo et al. [38], which introduced Digitial Hydraulic Simulator (DHALSIM), a novel Digital Twin for water distribution systems. The proposed virtual CPS can simulate the infrastructure that carries potable water. For example, the tool can simulate a large-scale virtual water grid in a city. DHALSIM co-simulates an industrial network emulator MiniCPS proposed by Antonioli & Tippenhauer [39] and a hydraulic simulator Water Network Tool for Resilience (WNTR) proposed by Murray and Haxton [40] to simulate the totality of a water system including a virtual industrial control system. The simulators provide high-fidelity models to accurately co-simulate network and physical processes, enabling cybersecurity research. The DHALSIM tool makes it possible to configure network topologies as well as realistic water distribution networks for modelling operational conditions affecting an industrial network. DHALSIM emulates physical parameters, information, and data regarding SCADA operations and network traffic in the form of a *.pcap* file.

Out-of-the-box DHALSIM supports various configurations for simulating different attack scenarios, including direct attacks targeting PLC devices or man-in-the-middle attack strategies. When we consider the current build-in attacks in DHALSIM, we have the following options:

- **Device Attacks:** these attacks emulate a situation where an attack is performed on the PLC level. For instance, an attack where an adversary would have direct physical access allowing to change values directly.
- **Network Attacks:** are attacks where a new node is added to the virtual Mininet network topology. This node represents an "attacker" and can perform two types of attacks on the network:
  - *Man-in-the-middle Attack:* where the attacker will sit in between a PLC and its connected switch. The attacker will then host a CPPPO server and respond to the CIP requests for the PLC.
  - *Naive Man-in-the-middle Attack:* are attacks where the attack will sit in between a PLC and its connected switch. The attacker will then route all the TCP packets that are destined for the PLC through itself and can, for example, modify the response to the other PLCs.

Furthermore, as can be seen in Fig. 5, the tool enables network type topology specification where we find two settings: *Simple network topology* and *Complex network topology*. Besides built-in attack strategies, one can also specify other simulation parameters such as attack triggers based on demand levels or sensor values, the number of iterations for which a simulation is executed, the water distribution topology in the form of a *.inp* file, and the overlaying SCADA architecture. For all the functionality of the tool and the various options, we refer to the documentation of DHALSIM on Github[2]. The architecture of DHALSIM is depicted in Fig. 6 and includes two primary components, MiniCPS and WNTR, which we briefly describe below.

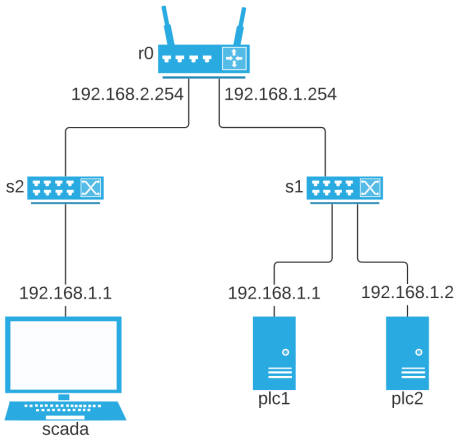
2.7.1 MiniCPS

MiniCPS is an industrial network emulation framework for simulating real-time cyber-physical processes. The framework utilises the MiniNet [41] framework as its basis but extends functionality to include PLCs and SCADA nodes. Mininet emulates the full network stack, allowing it to capture all network data types. Furthermore, Mininet allows configuring other network parameters, e.g., bandwidth, packet loss ratios, and delays.

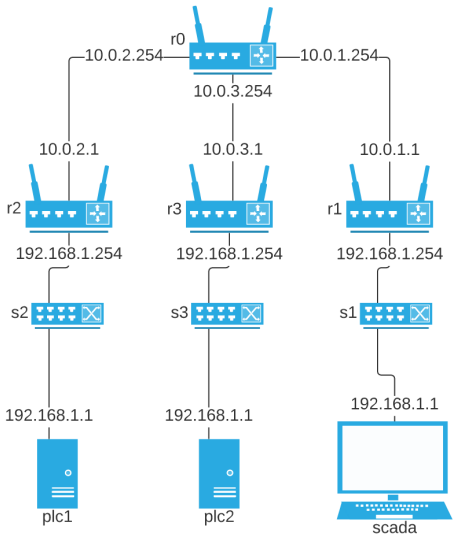
2.7.2 Water Network Tool for Resilience (WNTR)

As discussed earlier, one of the core components of DHALSIM is the WNTR [40], a Python-based framework for simulating water distribution networks and water quality. The framework is a wrapper for EPANET [42], an underlying application for Modeling Drinking Water Distribution Systems. The tool supports a wide array of components, including water valves, pumps, tanks, and reservoirs.

[2]https://github.com/afmurillo/DHALSIM



(a) DHALSIM Simple Network Topology.



(b) DHALSIM Complex Network Topology.

Figure 5: DHALSIM network topology types.

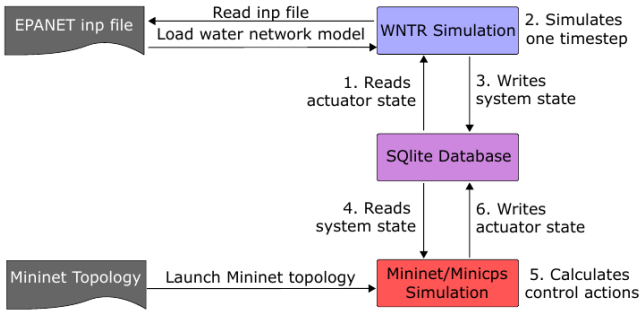


Figure 6: DHALSIM's co-simulator architecture [38].

When both simulation topologies are launched, state communication work through the SQLite database depicted in Fig. 6. The DHALSIM simulation framework provides the system variables (referred to as the "ground truth") for each step in the form of a *.csv* file containing the received and sent data for each industrial component in the architecture. In addition, the tool will generate a *.pcap* file for each node with at least one network interface.

The last point we want to address in this paper is the limitation of DHALSIM. Murillo et al. [38] indicate DHALSIM inherits some shortcomings of MiniCPS and WNTR. One such shortcoming is the limited capability to perform water quality analysis. This property implies that cyber-physical attacks on DHALSIM will not impact the water quality. Furthermore, the authors indicate that co-simulation presents several challenges concerning the synchronisation of events in the network and physical layers.

During a contact session with the main author, F. Murillo, we were informed that changes and updates are planned for the tools following a new publication. Therefore, we wish to stress the current version 0.3.0 of the tool which was used for this thesis [43].

2.8 Flow-based Traffic Characterisation

One option to perform network monitoring is to describe network communication into flow information. This form of information can provide security experts with a data source for detecting attacks. To better understand the features or information Flow-based traffic characterisation can provide, we take an initial dive into the inner-working of this passive network monitoring method and familiarise the reader with the core functionality and concepts. Later in the paper, we will consider this method as an input pipeline for the detection of malicious

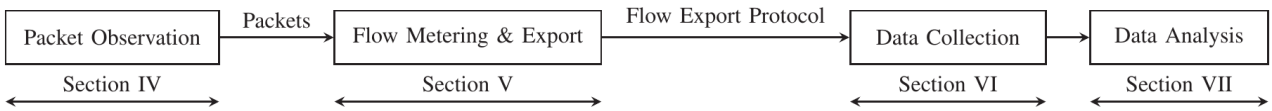


Figure 7: Flow-monitoring Architecture [44].

behaviour in network traffic. A *flow* can be defined as ”a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular Flow have a set of common properties. [44]”

As suggested by Hofstede et al. [44], flow-based monitoring tends to be more scalable in high-speed networks when we compare it to deep-packet inspection (DPI). According to Hofstede et al. [44], flow-based monitoring and deep-packet inspection pose major differences. Firstly, flows are an aggregation of often multiple packets avoiding the need to store each individual packet and thus having a lower storage ”footprint” - a storage reduction of 1/2000 of the original traffic volume can be achieved. The second main difference suggested in the literature is the fact that flow-based monitoring is less privacy-sensitive as traditionally, only the packet headers are considered. At the same time, with DPI, the payload can be inspected at the packet level. However, the authors indicate that this property is fading due to further integration of application-level information.

Fig. 7 depicts a typical flow monitoring setup consisting of several stages. The initial stage shows a packet observation point responsible for capturing packets from the line and pre-processing for further usage. Often, this functionality is supported by forwarding devices or network interfaces. The second stage in the architecture is given by a Metring and Export process responsible for transforming packets into flow aggregates. A data collector receives the export flow data from the previous stage. This stage allows for flow operations, including filtering, compression of data, and aggregation. In the last stage, data analysis is performed. Typical analysis strategies include network traffic classification through an observation point, traffic profiling, and anomaly- and intrusion detection.

For flow monitoring, one can identify several versions, such as *Netflow V5, Netflow V9, and IPFIX*, with Netflow V5 being the most popular among the three formats. Netflow V5 is a fixed format, while Netflow V9 is template-based, offering more flexibility regarding data fields that can be selected [45]. Furthermore, Netflow V5 is restricted to IPV4, whereas V9 supports the more recent IPV6 format. Comparing both Netflow V5 and V9, we observe that the two standards were originally developed by Cisco [46] whereas IPFIX emerged as an open standard. Netflow V9 served as a basis for IPFIX, and there are commonalities between the two standards except for additional fields found in IPFIX [46].

2.9 Zeek (Bro)

Zeek, formerly known as Bro, is an open-source passive network traffic analyser that produces high-fidelity ”transaction” logs of networking events a user can specify [47]. The tool emphasises network security, and over the years, the software has been further developed to support new features for network investigations and investigations into malicious or suspicious activity with the aim to exceed the capabilities of signature-based intrusion detection systems. The framework is fully customisable and allows for extending the core functionality with plugins using a custom scripting language to handle networking events. Although signature-based approaches are supported, the architecture's extensibility facilitates a broader spectrum of detection approaches, such as network-based anomaly detection using machine learning.

”Out-of-the-box” Zeek comes with extensive logging functionality describing networking events in the form of logs. Default logs include: *ssl.log(s)*, *conn.log(s)*, *http.log(s)*, *dns.log(s)* and more. Due to its open and non-proprietary nature, Zeek enables a solution that runs on commodity hardware and software, making it a versatile platform. The software accommodates a balance between high-fidelity networking information and storage or scalability.

Fig. 8 depicts Zeek's internal architecture which is consists out of two high-level components: *The Event Engine* transforming incoming packet streams into so-called *events*. The event describes what a vantage point or Zeek has observed in neutral terms. This means events are logged without stating the significance level of why an event is raised. The event engine consists of several sub-components for processing packets, sessions, and file analysis. The last component based on the events is captured in Zeek's Policy Scrip Interpreter component and is responsible for expressing security policy semantics to the events from the previous component.

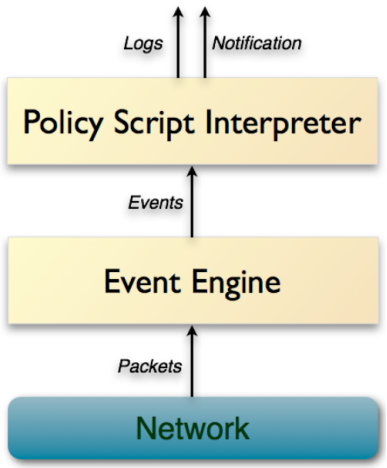


Figure 8: Zeek architecture.

3 Related Work

Although attack detection in cyber-physical systems is a novel research area, several works in the scientific literature lay a good foundation for ICS attack detection. This section addresses literature based on attack detection in industrial control systems. The different publications are discussed chronologically.

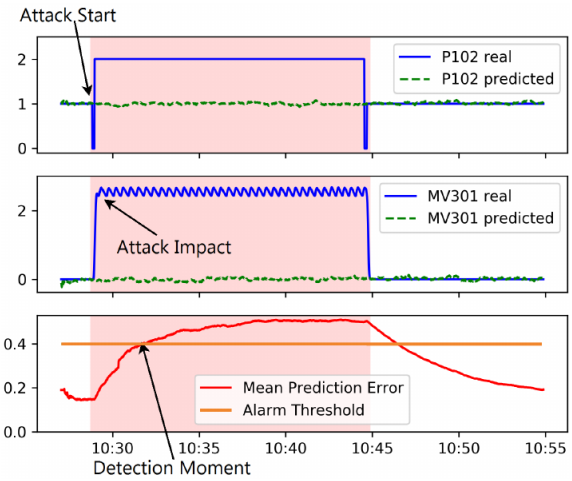
One of the early detection approaches was proposed by Jones et al. [48] Their method is based on constructing a Signal Temporal Logic (STL) formula to describe the ordinary behaviour of the CPS. STL expresses systems properties according to time bounds and bounds on physical systems parameters. In other words, STL is a predicate temporal logic defined over signals and can express system properties that include time bounds and bounds on the physical parameters—allowing for comparing individual system states with a pre-defined global property or model that the systems require to satisfy. The research aims to find an STL formula with a high correct detection rate and low false alarm rate. Using an unsupervised learning algorithm, the authors propose a method to learn the STL formula automatically and use the output for ICS attack detection, proving a suitable solution for classifying attack behaviour.

Rather than looking at the physical features for attack detection, Terai et al. [49] present a support vector machine-based architecture using features of ICS communication to detect attacks. The discriminant model allows for distinguishing between standard packets and attack packets on ICS communication. The downside of the considered method is that the authors train the model in a *Supervised* fashion requiring a labelled dataset. The results indicate that support vector-based detection can successfully spot attacks based on simple features such as packet interval and packet length. However, the authors argue that the technique does not improve earlier IDS performance. It allows for attack packet detection targeting individual machines and looks at communication features rather than features from the physical processes, such as sensor readings and actuation values.

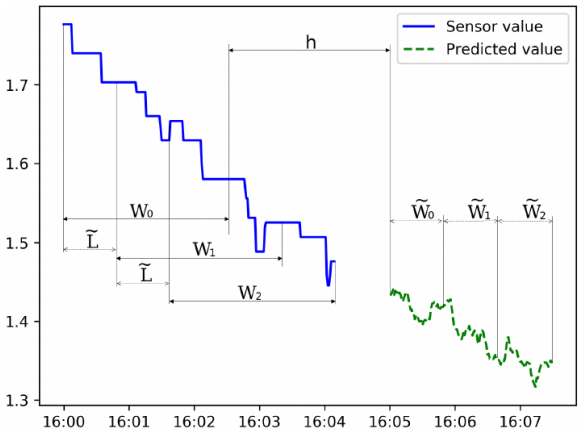
The research by Goh et al. [50] is a machine learning-based attack detection. They propose using Long Short Term Memory Recurrent Neural Network (LSTM-RNN) combined with the Cumulative Sum method to detect anomalies in a long sequence of data. This machine learning model allows for learning sequences containing patterns of unknown length without labelled data. Based on the prediction output, the LSTM-RNN model, the difference between the actual sensor value and the output is calculated to detect deviations. If the value exceeds a set threshold, the authors determine that an anomaly has occurred. The authors used the publicly available Secure Water Treatment dataset (SWaT) [51] based on a small-scale physical water treatment plant with a six-stage filtration process to mimic a large modern water treatment facility. In the proof-of-concept, the authors use the sensor data of six sensors that are involved in the normal behaviour of water filtering in the SWaT dataset. In total, 496.800 samples were used for training the LSTM-RNN according to the system's normal behaviour. To evaluate their approach, the authors consider various attacks against the water system PLCs, noting that the attacks are available in the dataset. All the attacks target the PCLs by, for example, spoofing the values. Different attack scenarios can be considered: Single Stage Singe Point (SSSP) focusing on a single attack point within the same state. The intent can be to attack a sensor to cause an overflow of the water storage tank. Alternatively, causing the system to think water is present in a tank while the tank is empty, causing the pump to continuously pump out water, potentially damaging it. Next, the author considers Single Stage Multi-Point (SSMP) attacks focusing on multiple attack points, for example, targeting water pumps. Multi-Stage Single Point (MSSP) attacks target a single point in the system with multiple stages. The last attack scenario the authors evaluate their detection system is against a Multi-Stage Multi-Point (MSMP) scenario, as the name suggests targeting multiple points with multiple stages. In total, the authors claim that nine out of the ten attacks could be detected with a total of three false positives. The limitation of this proposed approach is that the detection method can only be applied to systems with an available trained model, and due to the fast amount of data and complexity of industrial control systems, this might take time or can prove difficult.

Instead of evaluating one Machine Learning (ML) architecture, Shalyga et al. [52] introduced an approach based on a genetic algorithm (GA) to find the best NN architecture for a given dataset. They perform their research also on the SWaT dataset [51] as we have observed in the research by Goh et al. [50]. Similarly to the paper published by Kravchik and Shabtai [53] they forecast sensor values based on previous monitored values, as depicted in Fig. 9b or given by Eq. 7. The authors propose consecutive to various prediction algorithms and

different threshold calculation mechanisms. Generally, the anomaly will be detected if the mean prediction error exceeds the threshold. The mechanism is also depicted in Fig. 9a.



(a) Attack Detection Method.



(b) Prediction Window for Sensor Values.

Figure 9: Physical based attack detection approach by Shalyga et al. [52].

Furthermore, Shalyga et al. [52] use a genetic algorithm for automatic neural network generation via evolutionary computation, which can automatically perform architecture optimisation. Their approach considers three architectural templates: multi-layer perception, convolutional networks, and recurrent neural networks for time-series analysis. According to the researchers, the findings suggest that multi-layer perceptron performed best on the SWaT [51] dataset with a F_1 score: 0.812 and an average delay of 11% of the anomaly's length.

$$\hat{y}_i = \hat{f}_N(y_{i-1-l}, ..., y_{i-1}) \tag{7}$$

Next, the authors propose using the Kolmogorov-Smirnov test for feature selection. The methods were evaluated on three public datasets, SWaT [51], BATADAL [54], and the WADI dataset [55]. An overview of the proposed method can be found in Fig. 10. For each model, 1D CNN, PCA, UAE, and VEA hyperparameter tuning is performed using genetic algorithms and grid search.

The research findings suggest that for the BATADAL dataset and the SWaT dataset, AEs provided the best detection results, even exceeding the state-of-art performance on three public datasets while maintaining simplicity and generality. However, PCA also performs well universally for all datasets and is fairly simple to apply, proving a solution that can be sufficient in many real-world setups. Therefore, the PCA-based approach can serve as a good baseline attack detector before resorting to more complex approaches.

Lastly, the authors state that frequency domain analysis can contribute to attack detection, but it is suggested to be less effective in a system with unstable periodicity and poses imprecise detection of short cyber-physical attacks.

When comparing the earlier results by Shalyga et al. [52] with the results of Kravchik and Shabtai [53] on the SWaT [51] dataset, we observe that Kravchik and Shabtai achieve a higher F_1 score of 0.885 compared to the achieved F_1 score of 0.812 by Shalyga et al. [52]. However, nothing seems to be mentioned on the additional attack delay or attack detection latency.

Xu et al. [56] present ATTAIN, a novel anomaly detection approach that builds a Timed Automaton as a digital representation of a CPS. Furthermore, the approach utilises Generative Adversarial Network (GAN) to implement digital twin capabilities, which, according to the authors, can be trained based on both historical unlabelled and real-time unlabelled data. Similar to Kravchik and Shabtai [53] the authors evaluate the proposed method against the SWaT [51], WADI [55], and BATADAL [54] datasets. The digital twin element of the design is split into

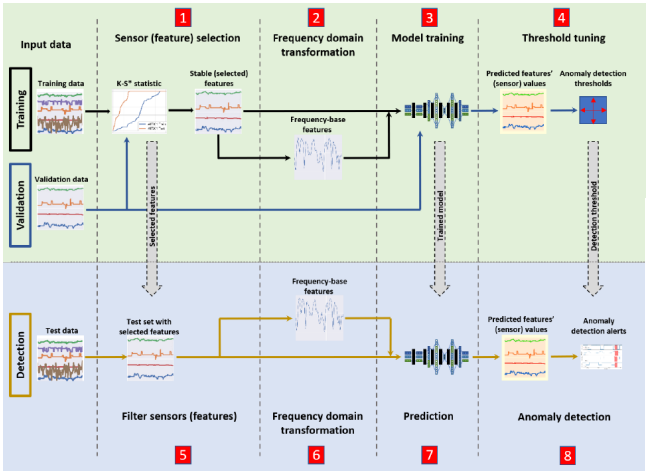


Figure 10: Overview of proposed method by Kravchik and Shabtai [53].

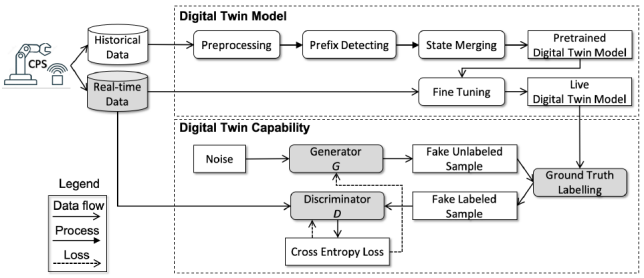


Figure 11: Overview of proposed method by Xu et al. [56].

two segments: *Digital Twin capabilities*, which entails the anomaly detection capability, detecting anomalies under operation. This segment is trained on real-time data only, and a GAN is used for the backbone of the design depicted in Fig. 11. We will omit the underlying technical details for the scope of this report. The second part of the design is the proposed *Digital Twin Model* referring to the digital representation of the CPS. The *digital Twin Model* provides the ground truth for the *Digital Twin Capability* segment as can be seen in the architectural design of the the proposed method, see Fig. 11. The result of the approach indicates that on the SWaT dataset, ATTAIN achieved an F_1 score of 0.97585, achieving the highest score so far identified in the literature. However, for the WADI dataset and the BATADAL dataset, a lower F_1 score is obtained. Also, this proposed method should allow for real-time detection, a feature that has not yet been proposed in the previously discussed literature. Additionally, we observed so far that the papers mentioned earlier primary goal was only to improve the detection evaluation metrics while not aiming to develop an approach that could be utilised in real-time, something that Xu et al. [56] try to address in their proposes approach.

A more recent paper by Jadidi et al. [57] presents a multi-layer anomaly detector referred to as ”AFAD”. The system consists of two primary components, a lightweight network-based anomaly detector based around Netflow data applying Hierarchical Cluster Analysis and a physical anomaly detector based on ARIMA/GARCH predictor for time-series forecasting of sensor values. The paper's authors indicate that this work presents the first implementation that utilises Netflow-based analysis in ICS architectures, allowing for scalable anomaly detection. The first layer, or the Netflow-based layer, detects abnormal behaviour using unsupervised histogram clustering for detecting flooding attacks, while, as mentioned earlier, the second layer uses the unsupervised forecasting models Autoregressive integrated moving average (ARIMA) in combination with the Generalised Autoregressive Conditional Heteroskedasticity (GARCH) model for detecting deviations from the expected model behaviour. When either of the two parallel detectors triggers, an alert is raised. The detection system was tested against *S7comm factory* datasets and earlier found Water distribution dataset: SWaT. However, since the SWaT dataset does not contain any flooding attacks, the first layer could not be used for attack detection reverting the detection to the log-based detector around the ARIMA/GARCH predictor-based detector.

During this part of the research, we found many other related works that were not completely aligned with the scope of our research. For the interested reader, an overview of various detection approaches is given in [58, 59, 60, 61, 62, 63, 64, 65].

Reference	Year	Content	Features	Model Type	Evaluation
[48]	2014	Applies machine learning and provenance analysis techniques to automatically detect the presence of APT infections within hosts in the network	Physical features	Signal Temporal Logic specification describing behaviour the system should adhere to under normal operation.	Performance Unclear
[49]	2015	Proposes detection architecture based on the ICS communication features. Allowing for distinguishing between benign ICS communication behaviour and attack behaviour.	ICS communication features such as, packet interval, packet length.	Based around supervised Support Vector Machine	Evaluation based on testbed by Matta et al. [66] F_1 **score: 0.95**
[50]	2017	Proposes long short-term memory using a recurrent neural network for a time-series predictor. Using Cumulative Sum method attacks are detected.	Physical features: Sensor values (present in the SWaT dataset)	Research considers: LSTM-RNN using Cumulative Sum for anomaly detection.	Against SWaT dataset[51] different attack scenarios are considered. Limited performance metrics are reported.
[52]	2018	Proposes method for finding optimised Neural Network architecture. Use of genetic algorithm for automatic for evolutionary neural network training. Best model found is based on multilayer perceptron	Physical features: Sensor values (present in the SWaT dataset)	Research considers: multi-layer perceptron, convolutional neural networks, recurrent neural networks.	Evaluation also performed against the SWaT dataset. Furthermore, used **NAB** (see paper) score for evaluation. **Detection delay: 11%** of anomaly length. F_1 **Score 0.812**
[53]	2021	Examines different lightweight machine learning models for ICS attack detection. The models predict time-series information. Optimisation based on grid-search and genetic algorithms.	Physical features: Sensor values (present in the different datasets, see evaluation column)	Research considers: 1D convolutional neural networks, shallow under complete autoencoders, variational autoencoders, and PCA	Using three publicly ICS datasets: SWaT[51], BATADAL[54], and WADI[55] dataset. Best performance metric for SWaT dataset: F_1 **Score 0.885**
[56]	2021	Presents ATTAIN an approach using a digital twin. The authors propose the use of Generative Adversarial Network for Method allows for real-time detection.	Physical features: Sensor values (present in the different datasets, see evaluation column)	Digital Twin based model utilising Generative Adversarial Network.	Evaluation considers SWaT [51], BATADAL [54], and WADI [55] dataset. Performance for SWaT dataset: F_1 **score: 0.9785** .
[57]	2021	Introduces AFAD the first method considering a multi-input method. The authors apply physical-based anomaly detector and network information-based flood detector.	Physical features Sensor values and Network traffic data (*.Pcap*) (present in the different datasets, see evaluation column)	ARIMA (physical-based detection) Hierarchical Cluster Analysis (HCA) for network detection.	Evaluation considers SWaT [51], S7comm factory dataset [67]. Performance for SWaT dataset: F_1 **score: 0.91** .

Table 2: An overview of the state-of-art regarding attack detection in ICS cyber-security research. Including evaluation metrics, type of model approach, and used features.

3.1 Lessons Learned and Open Knowledge Gaps

We observe that there is already well-established research on attack detection on single layers of the Purdue model (layer 0-2, in Fig. 1). State-of-the-art detection methods often leverage physical parameters, i.e., sensor readings or system value logs, that are bound to the physical process. This is also shown in the work by Kravchik and Shabtai [53], which presents the promising idea of forecasting future sensor values based on historical data for anomaly detection. Furthermore, some approaches, including the work by Terai et al. [49] use ICS communication information such as packet data for attack detection. However, their approach used a supervised method, something that is difficult to implement because there is a lack of labelled data, and labelling data requires enormous human effort.

Based on the findings within the state-of-the-art, we have identified a research gap that we want to address in this thesis. To the best of our knowledge, research into multi-domain attack detection or combining various detectors of different layers is only limitedly done before being compared to detecting attacks on specific single levels. Furthermore, multi-stage attack detection in ICS architectures where an adversary traverses from higher levels of the Purdue model to lower levels remains a relatively new research area. Another point we want to address is that limited attack expandability is present with the use of public datasets. As a result, researchers are often bound by the attacks contained in the datasets, hampering detection research with novel attack strategies. To remedy this issue, researchers can inject adversarial data into the dataset. However, this approach is limited in capabilities as it often does not allow for simulation or understanding of the physical (cascading) effects on ICS architectures. Moreover, the attacks in the public datasets are only partially mapped to the Mitre ATT&CK ICS framework or the Spiral Model, resulting in a lack of standardisation across the proposed methods and literature. Fourth, we found during the state-of-the-art survey presented in Section 3 that researchers utilised different metrics to evaluate the proposed detection methods, making it difficult to compare the various methods. Therefore, it is essential to identify good evaluation metrics for this thesis. We have only identified one paper that proposes a multi-input anomaly detector for ICS architectures [57]. However, their network-based detector focuses only on one type of attack strategy, something that arguably can be improved, as we propose in the next section. Furthermore, their multi-input method is only limitedly tested as not all of the datasets considered

contain the attacks for which the multi-input detector is designed. Lastly, many of the introduced ”physical” anomaly detection research rather focus on purely optimising machine learning models for the best metrics than taking a security standpoint and develop from a defensive need where attack strategies are carefully considered.

Below is a non-exhaustive summary of gaps and research recommendations in the literature where we differentiate between general knowledge gaps and specific knowledge gaps. The general knowledge gaps are too extensive to solve in one study, which is why we focus on the more specific knowledge gaps in this study. From these knowledge gaps, we based the aforementioned research questions in Section 1.1.

General knowledge gaps in the literature:

- Limited available ICS datasets.
- Datasets often used in the literature are based around small testbeds making it hard to evaluate proposed systems against large-scale ICS deployments.
- Lack of ability to test against different and novel attack scenarios.
- It is sometimes unknown what version of a dataset is being used. Different versions of datasets are referred to by the same name but may contain different experiments and attacks.
- Researchers evaluate based on different performance metrics hindering comparison between papers.

Specific knowledge gaps addressed in this research:

1. Limited available public datasets can be used to develop multi-domain ICS attack detection methods.
2. Attacks used in the literature for attack detection performance evaluation are only partially mapped to the Mitre ATT&CK ICS framework or the Spiral model shown in Fig. 2 leading to a lack of comparability and standardisation.
3. To our knowledge, no paper addresses DHALSIM in the context of detection development or cyber-security research.
4. Limited literature that introduces network-based attack detectors for ICS architectures could be found. Among the consulted papers, the majority tends to focus on the physical sensor or process-aware detection.
5. Current research frequently uses data from single layers, leaving the option open for combining data from various layers of the Purdue model for ICS attack detection. For example, only one approach among the consulted papers combines sensor data from layer 0 with network data from layer 2. However, the proposed approach still leaves options open for improvement. Simply put, little research has been performed on multi-input OT attack detection as opposed to single-input ICS attack detection,

3.2 Contributions

Given the specific knowledge gaps and the lessons learned in the previous section, we are seeking to improve the current state-of-the-art with the following contributions. Each contribution is annotated with the knowledge gap number that is addressed in the contribution.

- Present an overview and assessment of publicly available ICS datasets that allow multi-domain ICS detection research. *Knowledge gap: 1*
- Take an in-depth look at the *SWaT A6 2019* dataset and introduce a previously unseen interpretation of the dataset, mapping the dataset to the Spiral model introduced earlier and the MITRE ATT&CK ICS framework. *Knowledge gap: 2*
- Examine a novel means of data acquisition through a digital twin and generate a publicly available custom dataset using the digital twin (Dataset: [68]). *Knowledge gap: 1,3*
- Present strategic detection positions in the Purdue Model. *Knowledge gap: 5*
- Introduce novel ICS attack detectors predicting the logging behaviour of ICS components using Seasonal Arima and stateless differencing to detect attacks. *Knowledge gap: 4,5*
- Introduce novel ICS attack detectors monitoring for outlying connections using Empirical Cumulative Distribution Functions for outlier detection. *Knowledge gap: 4,5*
- Present a method for integration of physical alert data with network data to reduce false positives and extract the attack strategy of a multi-stage ICS attack. *Knowledge gap: 5*

4 Methodology

In this research, we aim to lay the foundation for a multi-domain ICS attack detection architecture that integrates network-traffic data and physical process information. This chapter describes the outline of our research methodology to find answers to the research questions formulated in Section 1.1 adopting a practical approach and propose a detection pipeline as described in Fig. 12. The core idea is to implement network- and physical sensor-based detectors and combine their output in a meaningful way. The resulting prototype is validated with the aid of assessed public ICS datasets.

As our focus is on detecting advanced persistent threats or a multi-stage adversary in ICS architectures, we aim to look beyond traditional signature-based methods because they remain an insufficient option. In most cases, signature-based fingerprinting methods are inadequate, as advanced threat actors often rely on zero-day vulnerabilities and adopt refined attack strategies, which a pure signature-based approach can miss. Instead, we consider an approach that learns the system behaviour corresponding to normal operation modes using past data in an unsupervised manner. This results in the need for high-quality and suitable data. Consequently, various data sources will be considered in the initial phase of the research.

From a high level, the methodology of this thesis can be best explained by Fig. 12. Initially, we will perform data collection and analysis in Sections 5, 6 where we distinguish between physical sensor data and network data. Then, we will consider separate detector types, network-based detectors, and a physical-based detector for each domain. In the end, we integrate the alerts of the various anomaly detectors presented earlier and test them against a publicly available ICS dataset.

As an alternative means of data acquisition, we also look at a publicly available digital twin of an ICS water distribution system. To our knowledge, we present the first work that uses data from the DHALSIM public digital twin put forward by Murillo et al. [38]. Second, we consider publicly available datasets used throughout the literature on ICS attack detection methods. Datasets in the lower layer of Fig. 12 should cover both network traffic data and physical sensor data, as we aim to integrate anomaly detectors for both types. However, there are several challenges associated with gathering suitable datasets. Firstly, cyber-physical datasets are only of limited availability due to the involvement of intellectual property, for instance, companies are reluctant to share their process data, impeding cyber-security research for CPS or ICS. Secondly, as we have encountered in the Related Work in Section 3, that anomaly detection on network traffic in ICS data is still in its early phases, often due to the public dataset limiting to only physical-based sensor data.

After the data identification phase, our project has two main detection components, or rather two domains: a network-based attack detection component and a physical-based detection component. As the name suggests, here we distinguish between network-based detection (Section 8), which performs anomaly detection based on network-traffic characterisations, we argue that there are already numerous papers with open-source code bases that propose interesting approaches for physical-based anomaly detection, such as DEAMON [69], MAD-GAN [60], FID-GAN [70], Sequence-to-Sequence [62], and RANSynCoders [71]. Due to the time constraints of this research project and the many works already addressing physical process-aware detection in literature, we adopt in Section 7 the method of using a mock-up physical-based attack detector, mimicking an effective anomaly detector.

In the merger part of Fig. 12 (Section 10) we want to explore the idea of integrating alerts originating from network-based detectors and the alerts raised by the physical-based detector in a meaningful way. We suspect that integrating physical and network alerts can reduce false positives, increase detection performance for cyber-physical attack detection or allow for identification of the attack strategy during a multi-stage attack. To illustrate, detecting an anomaly in a physical-based sensor does not automatically imply a cyber-attack: a sensor might provide faulty readings due to environmental influences, a sensor can be broken, or send duplicated faulty data [72]. For that reason, we suspect the integration of network-based information can increase alert understanding, provide attack insight, and reduce false positives. For instance, suppose an alert has been raised through physical-based detection indicating an anomaly in the physical process, and before this alert, anomalous behaviour in the network traffic has been seen. We can state with more statistical significance or certainty that a

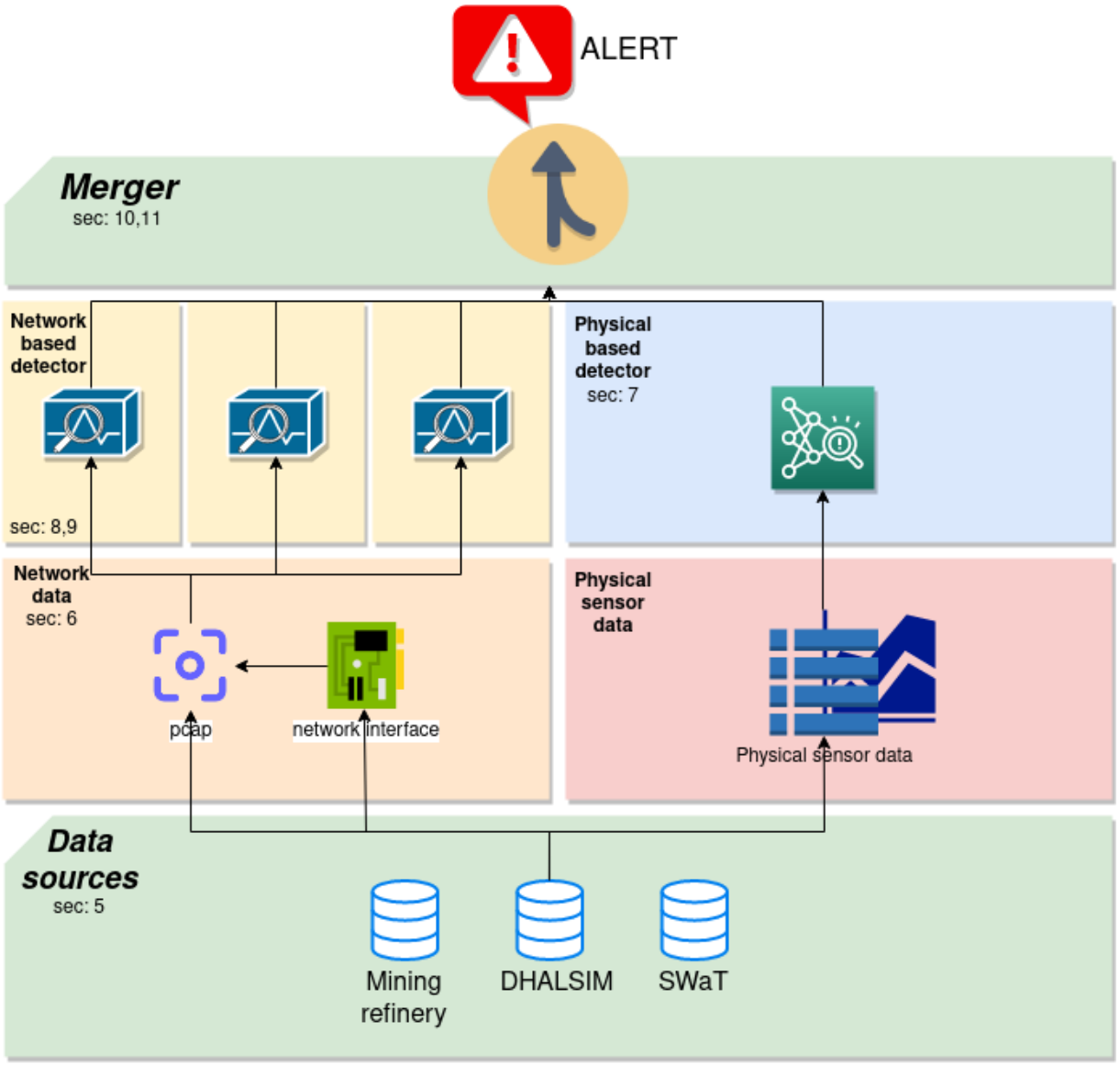


Figure 12: Overview of approach with section numbers.

cyber-attack has occurred and possibly determine the attack path an attack has taken through the Purdue model. Besides, we suspect that the proposed solution, in turn, might even be able to detect attacks even before reaching the layers where an attacker achieves the ability to interfere with the physical process, for instance, reaching level 0 of the Purdue model shown in Fig. 1 resulting in a water tank overflow or other cascading effects. The alert integration of the two different domains is only minimally addressed in the identified current state-of-the-art on attack detection in ICS. Therefore, our research can contribute to effective ICS cyber-attack detection based on network traffic characterisation and physical-based detection. Increasing the defensive capabilities in ICS architectures.

5 ICS Dataset Survey

For the development of an effective multi-domain ICS detection system, it is essential to have an understanding of what datasets are available. For our approach, we require datasets that contain both network and physical-process data, include sufficient normal system behaviour for both of these domains, i.e., are complete, present a worthy representation of a real ICS architecture, and are accurate and valid. When scouting the appropriate literature, we did not find many eligible datasets. One significant challenge in developing ICS attack detection systems is the absence of quality that fulfils the points above. First, industrial control systems are often highly specific towards particular systems, with organisations being reluctant to share their ICS datasets. There exists a fear among operators that this will lead to a loss of proprietary process information, exposure of confidential information, release of data on the network architecture, or disclosure of protocol information. On top of that, proper data sanitation can be a challenge which further fuels this fear. Secondly, consulted ICS datasets lack the required detail and representation of real-world ICS deployment, hindering the development of systems that can be applied to the existing real-world ICS architectures.

In this section, we survey different public datasets that can be used in high-quality attack detection research. We present two approaches — first, the use of existing public datasets, and second, a novel approach for data acquisition, focusing on the DHALSIM digital Twin for simulating ICS water distribution architectures. One earlier identified paper that we specifically discuss the work of Monti et al. [4], which compares publicly available ICS datasets and provides indexing according to the available attacks contained in the dataset and other features, such as data type and format. Based on the paper presented by Monti et al. [4] and earlier literature investigation, we establish a selection of public datasets that we analyse in detail for the scope of this research.

As our stated approach is the first to meaningfully integrate network traffic data with physical sensors and actuator data, the available datasets should fulfil these requirements. Not all datasets stated below contain the required physical sensor data and network traffic information. However, we suspect some datasets will still allow us to develop a feature set and test our detection approach. Furthermore, some of these dataset representations can also be simulated in the DHALSIM digital twin and, therefore, are worth mentioning.

5.1 ICS Dataset Overview

Below we give a selection of publicly available datasets used for evaluating ICS attack detection architectures by other identified papers. The dataset(s) of choice should represent a realistic ICS architecture while containing *attack data*, *normal operation data* and preferably also contain benign background traffic. Furthermore, as stated earlier, this research intends to develop an attack detection method to detect a large set of attacks. The dataset should preferably include a wide variety of attack scenarios emulating an advanced persistent threat. With all these points in mind, the following datasets were identified. For some datasets, multiple experiments or sub-datasets are provided with different configuration settings, attacks, or setups which can be found in the Appendix A

- **The Secure Water Treatment (SWaT) Dataset** - was first presented by Mathur and Tippenhauer [73] and is part of the Singapore University of Technology (Itrust research group). The system represents a physical water filtration testbed for cyber-security research and training. The dataset contains a multitude of different experimental data captures.
- **The Water Distribution (WADI) Dataset** - is an extension of the SWaT testbed and was first presented by Ahmed et al. [74]. The architecture represents a scaled-down version of a large-scale water distribution architecture often seen in cities. Since the testbed architecture is physically connected to SWaT, discussed later in this section, it is automatically part of the Singapore University of Technology (Itrust research group).
- **The Batadal Dataset** - presented by Taormina et al. [75] is a synthetic dataset of C-Town, a digital representation of a large-scale water distribution network. The datasets contain various simulations with variable simulation lengths and are partially labelled.

- **S7comm factory Dataset** - is in comparison to the above datasets, the S7comm factory dataset represents a mining refinery plant and is first introduced by Rodofile et al. [76]. The dataset provides a labelled dataset containing various cyber-physical attacks on a physical testbed.
- **DHALSIM - custom generated dataset** - this dataset is custom generated concerning the scope of this research using the DHALSIM digital Twin [38]. To our knowledge, this is the first paper addressing the usage of DHALSIM-generated data for attack detection development.

In Appendix A we give a detailed overview of the various datasets, sub-datasets, or experiments present for each surveyed dataset and describe whether or not the experiments contain network data, physical sensor data, or both. We find that the *SWaT A6 2019* dataset and the *S7comm factory* dataset contain both network and physical sensor data. However, the *S7comm factory* dataset is limited in size with only four PLCs. Instead, the *SWaT A6 2019* dataset presents a more elaborate testbed with six main processes tracked by a variety of control hardware. Next, we find that the *SWaT 2015* dataset contains data for an extended period. In contrast, the *SWaT A6 2019* dataset contains only a few hours of data. However, the network data is captured using a custom logging format and is unavailable in raw captures. Next, in Appendix A we present the size of the various datasets to get an idea of the order and representation of the dataset. As well as discussing the options for the *labelled* column: *not* labelled, *Partially* labelled, indicating that the dataset does contain some information, often in the form of external documentation allowing for inferring events in the dataset. For *Partially* labelled data, we argue that the dataset itself is not labelled directly, and finally *labelled* yes, indicating that we can apply the dataset for supervised learning methods.

5.2 The SWaT Dataset

The SWaT dataset contains measurements of a real and modern ICS testbed allowing researchers and practitioners to develop and assess attack detection algorithms supporting cyber-security research. The architecture represents a scaled-down version of a modern water treatment facility, and consists of a six-stage [73] water filtration process. The physical installation of the system is roughly 90 square meters and can be found in Fig. 13. Each stage of the SWaT architecture has two PLCs assigned: one PLC (primary) is responsible for interacting with local sensors and control actuators like valves and pumps. The second PLC functions as a backup in case failure occurs in the primary PLC, proving to be a Safety Instrumented System. The six stages of the physical process can be given by: the (P1) stage of raw water intake followed by a chemical process (P2). Next, the water is filtered through an Ultra-filtration system (P3). Then, a pump feeds the water through an ultra-violet de-chlorination component controlled by a PLC responsible for P4, sending the processed water to a Reverse Osmosis system in P5. The last stage (P6) supervises and regulates the cleaning of the membranes in the ultra-filtration unit by controlling the backwash pumps. Fig. 14 depicts the six-stage SWaT process: the solid arrows indicate either the water flow or the chemical dosing flowing in the system, while the arrows with a cross indicate possible attack points in the system.

The local communication between the PLC and the connected sensors or actuators uses Allen-Bradley's Device Level Ring (DLR) protocol, ensuring that failure of a single link does not lead to failure of the control functionality or data loss. The six-stage process controlled by PLCs communicate over a separate network. This network can be seen as a standard Ethernet star-based topology, with an industrial switch connecting all the six stages mentioned earlier [73]. Network communication by the various system components such as PLCs, actuators, and sensors in the architecture uses the Common Industrial Protocol (CIP) over EtherNet/IP (ENIP) stack, which can be wireless or wired. Fig. 15 depicts the network architecture between the PLCs of the six stages and the various architectural components such as engineering workstation, historian servers, HMI touch panel, etc.

The public repository provided by the Itrust research groups provides the data of various operational experiments on the SWaT testbeds. It is important to note that several updates have been performed on the SWaT testbed over the years. In Appendix A.1, Table 13 can be found giving the various datasets or experimental runs for the SWaT dataset. It is worth mentioning that not all presented sub-datasets in Table 13 contain network-traffic information.

Figure 13: Itrust SWaT testbed architecture [77],

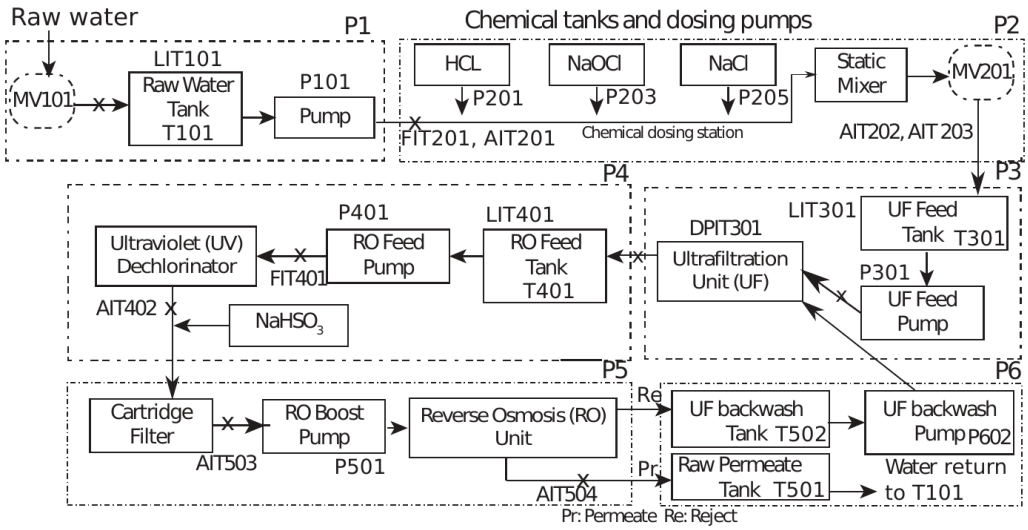


Figure 14: SWaT six-stage process, LIT: Level Indicator and Transmitter, Pxxx Pump, AITxxx: property indicator and Transmitter; DPIT; Differential Pressure Indicator and Transmitter [73].

5.2.1 SWaT.A6 Dec 2019

In this sub-section, we shortly want to address the *SWaT A6 2019* dataset specifically since this dataset contains both network traffic and physical data during 15 minutes of regular operation under attack operation of what can be considered an advanced persistent threat actor with a series of attack steps. The combination of network information and physical sensor data and the deployment of six independent attack strategies are beneficial for the scope of this research. Although six attack steps are mentioned in the documentation, we spot five steps. Furthermore, no paper referring to this specific sub-dataset could be identified in the previously observed literature. At the same time, we argue it is the first realistic dataset where adversaries traverse from higher- to lower levels of the Purdue model using several attack steps.

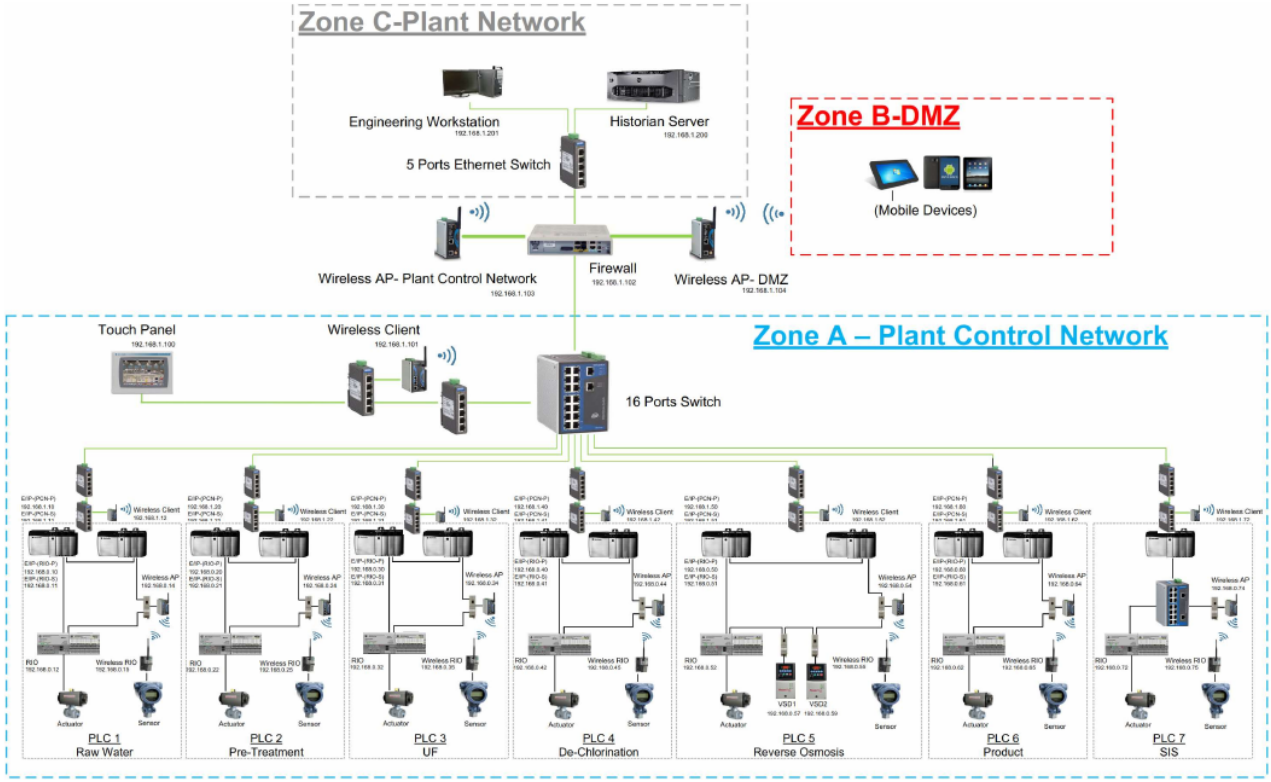


Figure 15: SWaT network Architecture [73].

Table 3: *SwaT A6 2016* - Information.

Experiment Time Range	06/12/2019 10:05:00 - 06/12/2019 13:45:00
Experiment Duration	0 days 03:40:00
Size - Physical	13201 rows x 81 columns (excluding timestamp column)
Sampling Time - Physical	1 Second
Format - Physical	Column based - *.xlsx*
Size - Network traffic	41.5 Gigabytes
Format - Network traffic	Packet Capture - *.pcap*
Number of packet - Network traffic	321086771 packets
Number of Attacks	6 (We observe 5)

The experimental run contains, according to the documentation, six attacks with a measurement duration of more than 3 hours. In Table 3 we provide some detailed information regarding the *SWaT A6 2019* dataset. The dataset can be considered *partially* labelled as it does not contains any annotation or labelling on whether a measurement can be considered malicious, as we observe in the *SWaT.A1 & A2_Dec 2015* dataset. However, when we assess the dataset, we find a measurement event timeline in Table 4 that describes the dataset's different events. Based on Table 4 we observe that an initial malware infection on the workstation leads to an entity in the network completing a data exfiltration of the historian data. After several data exfiltration events at regular intervals, the attacker continues exploiting the workstation with a second malware that is downloaded from the command-and-control servers, which is undocumented.

Following Table 4 we find that in the last step, the second malware starts to disrupt sensors and actuators in the control network resulting in physical impact.

Event Timeline			
Time	**Event** ⇓	**Time**	**Event**
1005	15 mins of normal operation		⇓
1020	First malware infection with USB thumb drive	1230	Disrupt Sensor and Actuator
1030	Exfiltrate Historian Data	1233	Sleep
1035	Sleep	1243	Disrupt Sensor and Actuator
1045	Exfiltrate Historian Data	1246	Sleep
1050	Sleep	1256	Disrupt Sensor and Actuator
1100	Exfiltrate Historian Data	1259	Sleep
1105	Sleep	1309	Disrupt Sensor and Actuator
1115	Exfiltrate Historian Data	1312	Sleep
1120	Sleep	1322	Disrupt Sensor and Actuator
1130	Rest 60 Min	1325	Sleep 5 Mins
1230	Infiltrate SCADA workstation with second malware download C&C server	1330	Post attack capture
	⇓	1345	End

Table 4: Event Timeline for SWaT A6 2019 dataset.

Finally, it is important to note that when using the *SWaT A6 2019* dataset, an offset of 8 hours between the physical data timestamps and the packets timestamps is present in the raw packets capture. This inconsistency needs to be accounted for during the pre-progressing phase.

5.3 The WADI Dataset

The WADI testbed is a physical extension of the previously discussed SWaT testbed to understand, execute, and analyse cascading effects of cyber-physical attacks on larger-scale water distribution systems [74]. The testbed is physically connected to the SWaT testbed to receive filtered water from the latter. Compared to the SWaT testbed explained in the above section, WADI consists of three process stages. The primary grid (P1) comprises two raw water tanks of 2500 litres with tank-level sensors to supervise water levels in the two tanks. The water tanks can acquire water from the SwaT system, a public water utility board inlet, or from WADI return water. The secondary process (P2) features six consumer tanks and two elevated reservoir tanks. The consumer tanks take water from the elevated reservoirs on a pre-determined demand pattern. The two elevated tanks are filled by the two raw tanks from process P2. When the consumer tanks have reached their full capacity, water is returned to the water return grid. Fig. 16 shows the three-stage process for the WADI testbed.

Furthermore, the difference between the SWaT and WADI testbed is the usage of the Modbus protocol in the latter over the Ethernet/IP stack found in the former for the control and acquisition of sensors and actuators. For the full technical specification of this testbed, we refer to documentation [78] and Appendix A.2 presenting a summary description of the identified datasets.

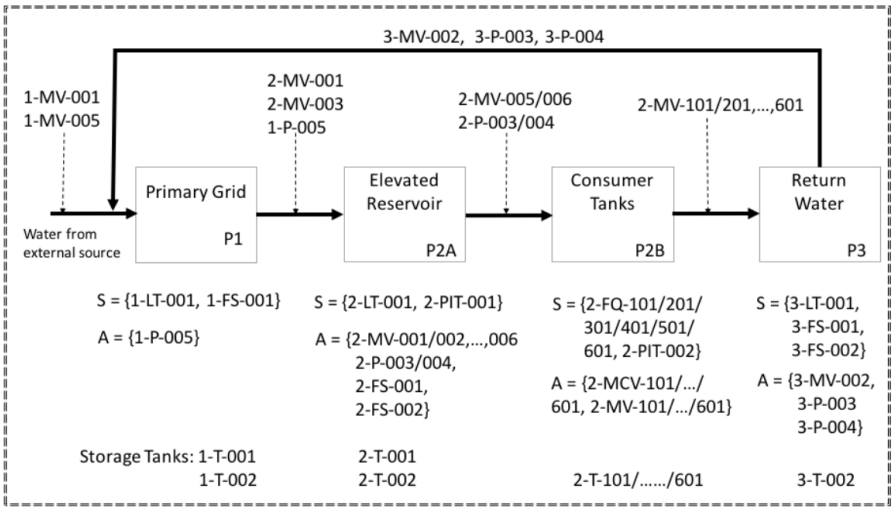


Figure 16: WADI three-stage process, S and A indicate set of sensors and actuators respectively, e.g., 1-LT-001; level sensor for stage one [74, 78].

5.4 The Batadal Dataset

The BATtle of the Attack Detection ALgorithms (BATADAL) proposed by Taormina et al. [75] fulfils the motivation to provide a dataset on which objective attack detection algorithms can be performed. The dataset is distinguished by the fact that, in contrast to previously seen datasets, it is a synthetic dataset, meaning that the dataset was obtained through simulation. Batadal is based on a digital network representing a medium-sized water distribution network called C-town depicted in Fig. 17. This virtual water distribution system is supervised by 11 PLCs that monitor the water distribution network and use a virtual SCADA system to collect system data such as PLC readings. According to the authors, the dataset is simulated using the MATLAB toolbox epanetCPA with an *.inp* emulation file available for performing simulation in the DHALSIM Digital Twin. The simulator provides a method to simulate the hydraulic responses of the water distribution and a wide variety of attacks. In Appendix A.3 we can find an overview of the available data (sub-)sets for the Batadal ICS dataset. Finally, although it is stated that the dataset simulates an ICS architecture containing a PLC and a SCADA network, no other network data could be found for this surveyed ICS dataset.

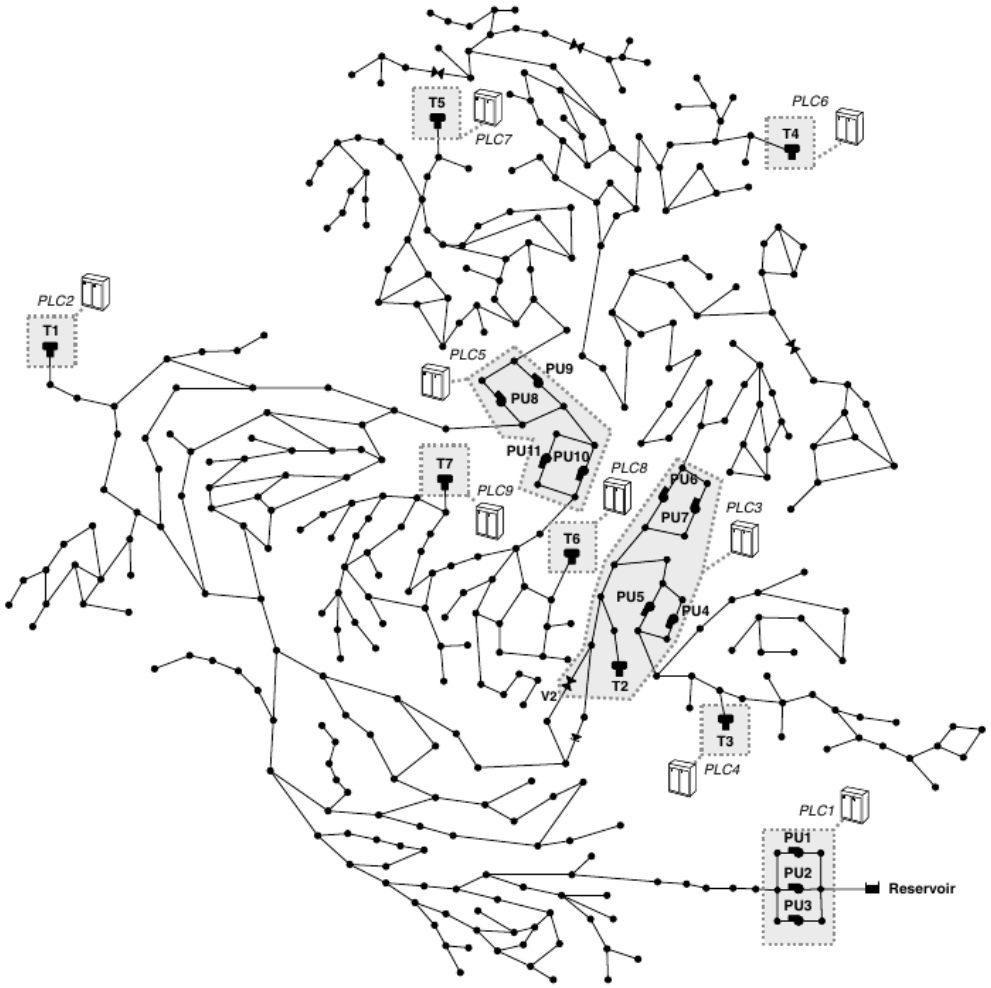


Figure 17: C-Town architecture representing a large-scale water distribution architecture.

5.5 S7comm Factory Dataset

This labelled factory dataset represents a small-scale mining refinery plant and is first presented by Rodofile et al. [76]. The dataset contains raw network traffic captures and SCADA process logs for the various testbed processes. The authors of the paper indicate that the dataset embodies three time-based sub-processes executed in the following order: Conveyor, Wash Tank, and Pipeline Reactor, each undergoing various attacks. Fig. 18 depicts the network structure overview where the system consists of three ”auxiliary” PLC controllers, each responsible for a sub-process. The attack strategy targets sub-processes via the Master PLC responsible for storing operational parameters in registers and only violates control processes via TCP/IP connections. The dataset contains raw packet captures and process logs for each sub-process for approximately 9 hours. A series of HMI logs from the four PLC devices are depicted for the physical process data in Fig. 18.

We identified two different versions of the dataset where the differences and the respective descriptions can be found in Table 16 in Appendix A.4. For convenience we refer to two versions: *V1* made available by Rodofile et al. [76] and *V2* by Myers et al. [67].

5.6 DHALSIM Dataset

Earlier in Section 2.7, we introduced the high-level outline of the DHALSIM water distribution digital twin. This section will discuss an overview of the configuration settings and simulation datasets to explore the capabilities. Since DHALSIM version *0.3.0* comes with various network topologies ”out-of-the-box”, we will omit some of the water distribution network topologies for the scope of this research. However, three topologies are of interest

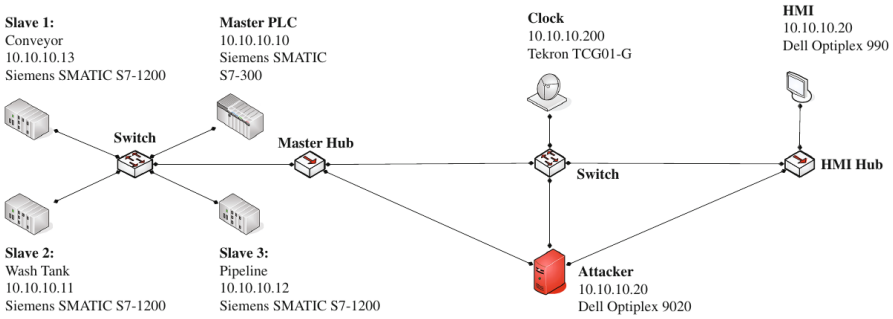


Figure 18: ICS network Test-bed *S7comm factory* dataset [76].

for our research as some of the presented configurations pose a virtual representation of a physical testbed, such as the one found in the WADI testbed or the Batadal virtual representation, and can be used for comparison later on in the research if required.

For the initial data exploration in the next section (Section 6) we generated different datasets for the *WADI_topology* using a complex SCADA network topology (see Fig. 5b) in DHALSIM, assuming (for now) that this topology is truthful to the real testbed generated data. In Table 17 in Appendix A.5 we present the various experimentation runs using the *WADI_topology* and different attack strategies to generate different exploratory datasets. Furthermore, for each run, we provide the required simulation time to get an idea of the computational requirements of the simulator. The DHALSIM digital twin simulates data for each configured element in the SCADA system and emulates the traffic over the virtual SCADA system. For each SCADA component configured, a separate raw packet capture of the traffic is provided, while for the physical sensor data, two separate files are generated: a *ground_truth.csv* indicating the real-values of the underlying physical process and a *scada_values.csv* listing the values seen by the virtual SCADA systems. This can also be seen in Fig. 19 where the parameters (configured in DHALSIM) of the underlying physical model and the values exchanged over the virtual SCADA system are shown. When we plot the *attack-mitm-medium* attack scenario in Fig. 20 simulating a scenario where an adversary performs a man-in-the-middle (ID T0830 in Mitre ATT&CK ICS). We can notice a disruption in the normal pattern highlighted by the red part in Fig. 20 showing a difference between the actual values of the physical process and the exchanged values over the SCADA system. In this case, an orchestrated attack would lead to an overflow of a tank as the sensor value is being spoofed, resulting in the system ”thinking” the tank is almost empty while it is not.

Coming back to the point of whether the data generated by DHALSIM using the *WADI_topology* accurately represents data of the ”real” WADI testbed, we compare the physical sensor data with the simulated physical data. We quickly find that for the WADI dataset, the column names deviate from the simulated *ground_truth* column names resulting in the fact that we cannot perform a one-to-one mapping between the simulated data and the real WADI dataset identified earlier. Secondly, when looking at the values simulated over miniCPS in the DHALSIM simulator, we note that the SCADA system is only partially simulated in the default DHALSIM *WADI* configuration. Alternatively, the network configuration of DHALSIM based on *WADI_topology* remains limited and does not yet fully capture the real testbed network architecture. This makes us suspect that (as of now) we cannot use the simulated results by DHALSIM as a representation of the real *WADI* testbed and use the dataset interchangeably. Lastly, during a contact session, it was indicated by the researchers of DHALSIM that a comparison between the simulator results and the physical-testbed results is still ongoing, and up to now, nothing can be stated on the similarity or truthfulness between the simulated dataset and the real dataset. The configuration of the generated datasets and the simulation results can be found on the following repository: [68].

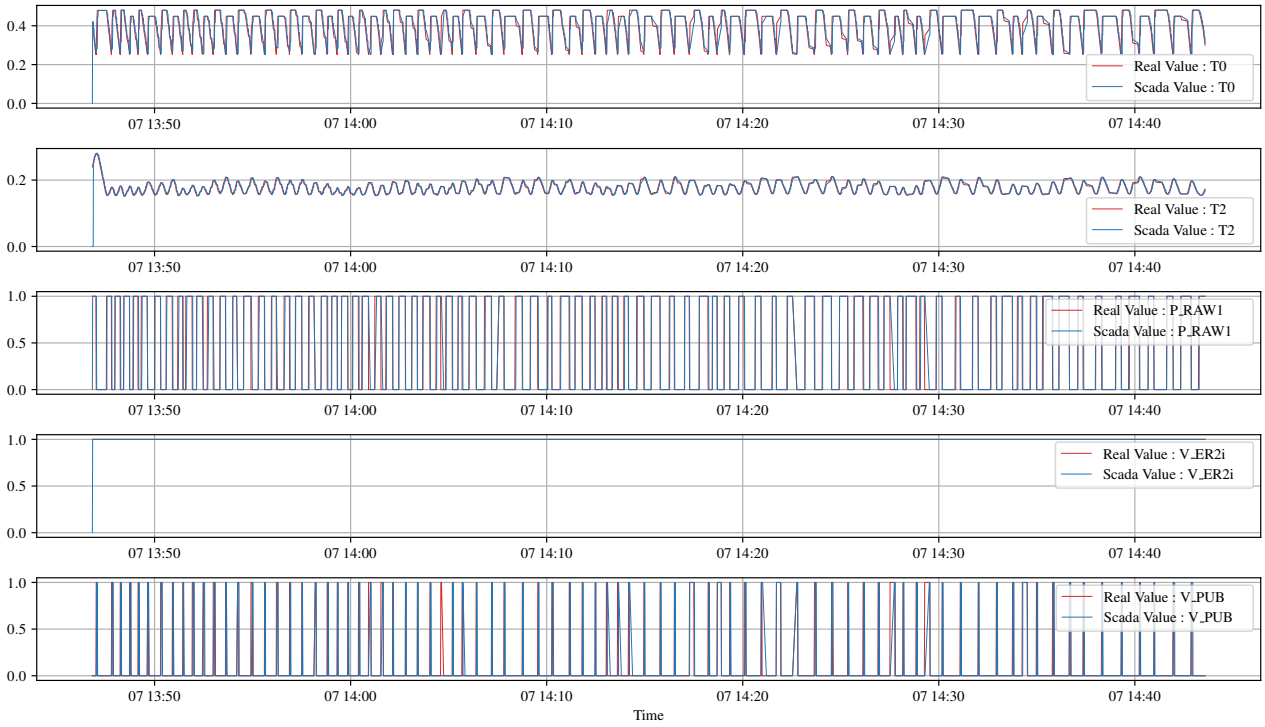


Figure 19: *Normal-operation-no-noise* sensor and actuator value plot.

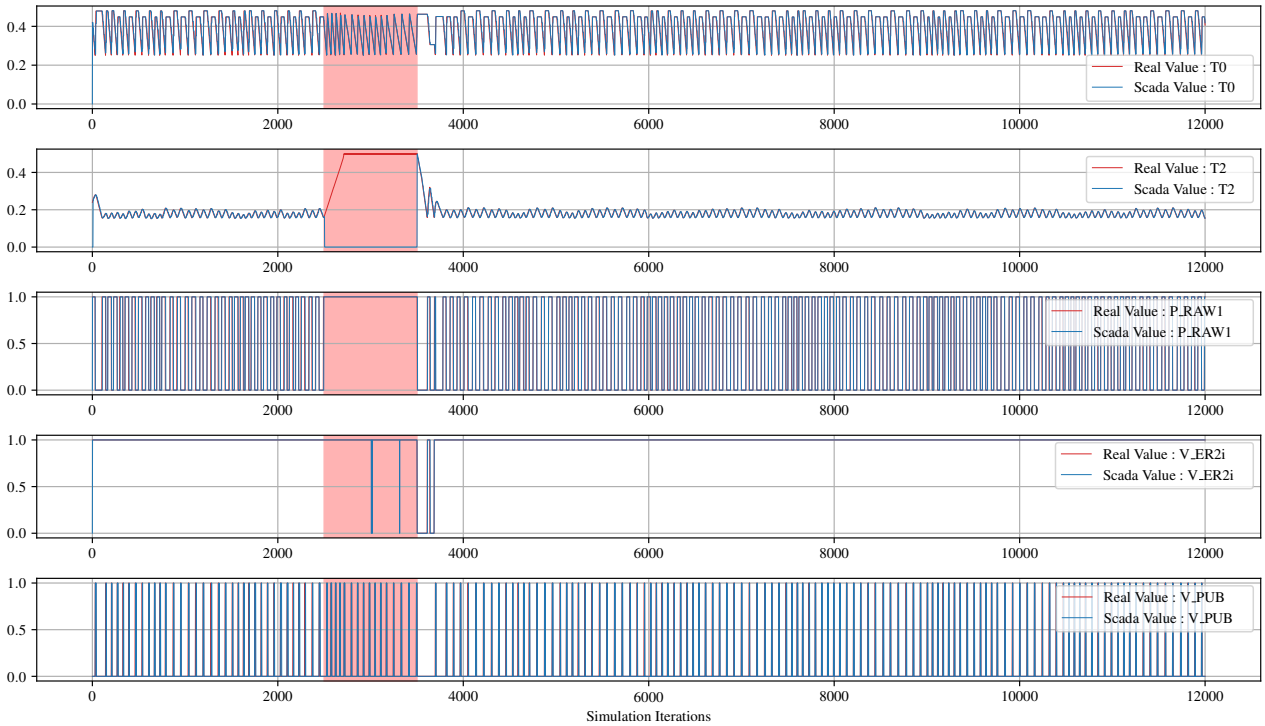


Figure 20: *Attack-mitm-medium* PLC attack causing a overflow in Tank *T2*.

5.7 Lessons Learned

In this section, we scrutinised different ICS datasets. At this point, we found that the dataset *SWaT A6 2019* and the *S7comm factory* dataset contain both physical sensor data and network data during a number of attacks. By comparing both testbed architectures, we found that the SWaT architecture exhibits a better representation of large-scale ICS infrastructure. In comparison, the *S7comm factory* architecture consists of only 4 PLCs, an HMI and a clock, while the SWaT architecture features six different processes, each controlled by two PLCs. In addition, the architecture contains higher layers of Purdue components, such as a SCADA engineering workstation and a historian server. Subsequently, mapping the Purdue model to the *S7comm factory* dataset proved more complicated. After all, we only found two datasets contain the required physical sensor data and network data. As a result, we looked at a third alternative method for data acquisition using the DHALSIM digital twin. We performed several simulations on the WADI topology under different attack scenarios. However, we discovered that the simulated and real datasets do not yet match. This point might be addressed in future versions of DHALSIM as the tool is still in development.

6 Data and Feature-set Exploration

After establishing an overview of the different datasets in the previous Section 5, we can explore the different datasets and test various network-based features and put forward different advantages, considerations for each dataset and feature-set options. For the continuation of this paper, we will refer to two different data sub-domains, physical and network data. According to the earlier proposed system architecture given in Fig. 12 it became apparent from the discussion in Sections 5.2 and 5.6 that only the two sub-datasets of SWaT and the DHALSIM simulated datasets contain raw packet captures and physical sensor information for a water distribution ICS architecture. Similarly, in Section 5.5 we have found that the *S7comm factory* dataset also contains raw packet captures and physical sensor data during normal and attack operations for an ICS architecture.

Furthermore, having various datasets allows us to test features under normal and attack situations across different ICS architectures preventing overfitting on one particular dataset. The goal for the dataset and feature-set exploration is to demonstrate or develop a set of unified features available across different datasets or ICS architectures that can be adopted in our network-based detectors and to get a deeper understanding of the data. As a starting point, we will discuss network-based features introduced by three different approaches to demonstrate what feature sets are useful for our network-based detectors. As a result, each approach was not tested against every dataset since the procedure was rather explorative than linear. To put it in another way, we found that some methods might not offer an "all-fit" solution and are therefore discarded against other datasets or not every method is tested against all datasets. Finally, in the feature-set exploration, we explain the different pre-processing methods required to convert the *RAW* data in accordance with the feature method.

6.1 Data Pre-processing

We consider three different feature classes for the feature-set exploration discussed in the next sub-sections. Some approaches require pre-processing of the collected data to be converted into the feature class representation explored for the method. Therefore, we give an overview of how the raw data pre-processing should be performed for the format following the intended feature class.

6.1.1 Deep-packet Inspection Pre-processing

For deep-packet analysis, we use Python in combination with the *Scapy* library [79], which allows for parsing individual packets. For network traffic in the form of raw-packet capture no, further pre-processing methods are required except for the *SWaT A6 2019* dataset, which needs to account for the timing offset of packets. This applies to all the feature set options given below.

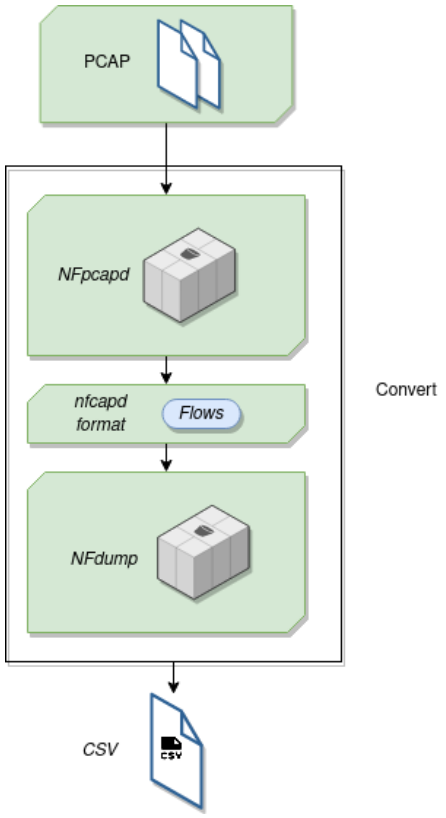
6.1.2 Netflow Pre-processing

Since the raw packet captures for the selection of the datasets are available in *.Pcap* format, pre-processing is required to convert the packet captures into Netflow V5 format. Fig. 21a depicts our current conversion method. In a real-world deployment, components can be interchanged to provide Netflow V5 format in real-time. Alternatively, hardware that supports this export format natively is available so that no conversion strategy would be required.

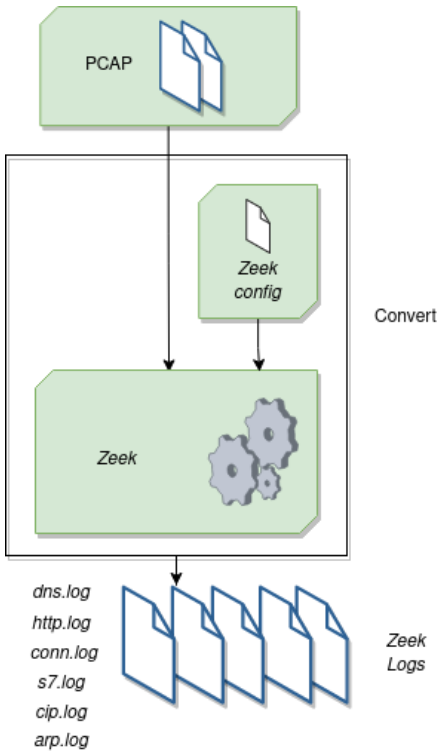
For the conversion we developed a custom conversion script that uses an optional binary called *NFpcapd* which is part of the *NFdump* toolkit [80] for converting raw packet captures from *.pcap* files to Netflow V5 format (*.csv format*). The conversion tool can be found on Github [81].

6.1.3 Zeek Pre-processing

Zeek is fully-customisable, allowing for extending the interpretation and event capabilities for common ICS control protocols and network activity. When Zeek is used in its default configurations, a variety of logs are generated depending on the networking event. These logs include *http.log* information, *conn.log* information and others. See the documentation for default configuration settings [47]. Due to the customisable nature of Zeek,



(a) Converting Pcaps to Netflow format.



(b) Converting Pcaps to Zeel logs format.

Figure 21: Pre-processing methods.

custom scripts can be used to extend the core functionality of the framework. To interpret ICS protocol-specific events, we extend the capabilities of the default configuration with a custom *ARP* event parsing script, a publicly available *CIP/ENIP* interpretation script [82] and finally, a parsing script for the interpretation of *S7* [83] communication which can also be encountered among the earlier identified datasets. Fig. 21b pictures our Zeek conversion strategy.

For the analysis of the data and log ingestion, the Zeek analysis tool (ZAT) [84] is used, allowing for the conversion of logs into an understandable Pandas format for data analysis in Python. The conversion tool can be found on Github [81].

6.1.4 General Pre-processing

Among the datasets, it can be observed that both network data and physical sensor information data can sometimes be ”noisy” for analysis. To make certain time series more ”stable”, we consider the simple rolling mean method where a one-dimensional time-series data frame can be considered as:

$$x[n] = [x_1, x_2, x_i .., x_n] \tag{8}$$

And the sequence of data with the rolling mean method is indicated by the following expression.

$$\bar{x}[w] = [\bar{x}_1, \bar{x}_2, \bar{x}_j, .., \bar{x}_w] \tag{9}$$

The rolling mean method is given by:

$$\bar{x}_j = \frac{w}{n} \sum_{i=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} x_i \tag{10}$$

The method is used to resample data into smaller series or to make the signal less noisy. In the figures, resampled data is labelled by $RM : \{resample time | resample factor\}$. *Resample time* indicating the interval over which the mean of a sub-set is taken and the *Resample factor* to indicate the number of data points over which a signal is averaged to make it more stable.

6.2 Raw Packet-based Features

Using the custom-generated DHALSIM dataset, we initially compared normal-operation data with the attack-operation data for different attack scenarios listed in the earlier Table 17 and explored a variety of different features that proved the ability to distinguish between normal and an attack situation.

Fig. 22 shows an overview of the features that allow us to differentiate between normal and attack measurements based on network-traffic information. The left column of sub-figures indicated by the Fig. numbers 22a, 22c, 22e respectively are showing behaviour on the *normal-operation-no-noise-medium* generated dataset using the *Wadi Topology* configuration. On the right side of the Fig. 22 the same selection of features are depicted for the *Attack-mitm-medium* dataset. Note that both the settings are exactly the same except for the attacks set in the experiment, which can be found in Table 17.

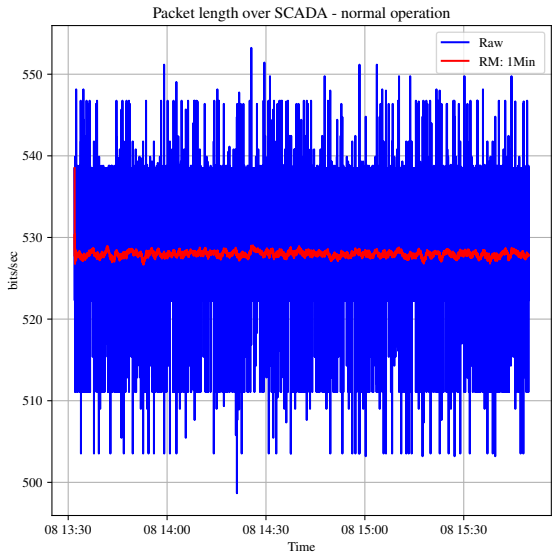
Fig. 22a and 22b demonstrates the traffic in $bits/sec$ that is being sent over the emulated SCADA control network. Since DHALSIM's default *Wadi Topology* SCADA control network remains limited in size, traffic deviation when a Man-in-the-middle is performed targeting Tank-level sensor *T2* are clearly visible, where one can observe significant differences between Fig. 22a and Fig. 22b. When inspecting with greater attention Fig. 20 indicating the effects on physical values and Fig. 22d depicting SCADA traffic, one can see that the Man-in-the-middle attack available in the DHALSIM simulator leads to both a physical effect and a network effect.

Next, we look at what entities in the virtual network are communicating with each other. Since DHALSIM takes a capture for each network entity that is being simulated, a separate plot per *.Pcap* is given in Fig. 22c which provides a network graph of the conversations between the various simulated PLCs and other virtual network entities. So in the case of *PLC2-eth0* shown in Fig. 22c, it can be seen that data exchange takes place with *IP:10.0.1.1* and *IP:10.0.2.1*. However, when comparing the capture strategy on the same entity during a MITM attack situation, one can see in Fig. 22d where an unidentified entity in the network with *IP:192.168.1.4* is introduced, likely indicating the presence of an adversary in the network. Fig. 22e and 22f depict for each capture taken on a PLC and the overlay SCADA the packets for the identified protocols during *normal operation* and *attack operation* for the network architecture presented in Fig. 5b. It can be easily seen that there is a significant increase in the number of ARP packets in the *attack* situation. We suspect that the built-in MITM attack option in DHALSIM performs an ARP poisoning attack to launch the MITM, causing an increase in the observed ARP packets.

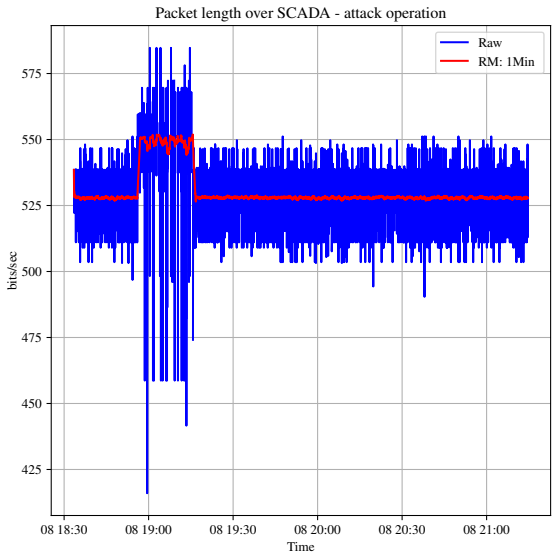
Continuing our feature set and dataset exploration, we intend to design and find a universal feature set that can be used across multiple ICS architectures for network-based attack detection. Yet, the presented DHALSIM features indicate that attack options in the tool available remain relatively easy to spot for the current *Wadi Topology*. Furthermore, for our own generated datasets, the scale of the simulated ICS control network remains rather limited in size. Using the default *Wadi* topology (DHALSIM) only two PLCs and a SCADA entity are simulated as shown in Fig. 5b. Instead, a suitable option would be to extend the network and simulate other PLCs to get a more representative real-world ICS architecture. However, when assessing and developing our feature set using the DHALSIM dataset, we recognise that the current version of the DHALSIM digital twin remains rather limited in attack capabilities, "Out-of-the-box" only two attack types are supported in the DHALSIM data simulation environment.

6.3 Netflow-based Features

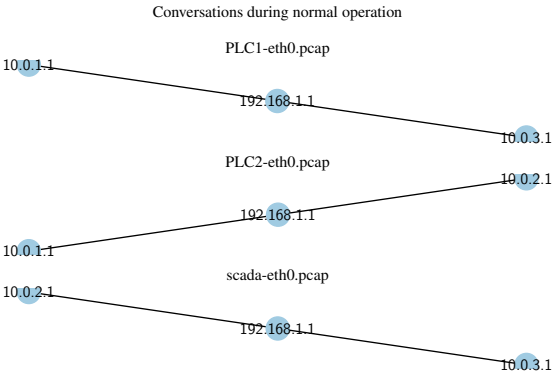
As the previous section showed that the current version of DHALSIM provides limited attack options, we utilise data from the SWaT testbed instead. Based on the dataset survey in Section 5 and Table 4, we find that the *SWaT A6 2019* includes more attack steps compared to the own generated DHALSIM datasets. Moreover, similar to



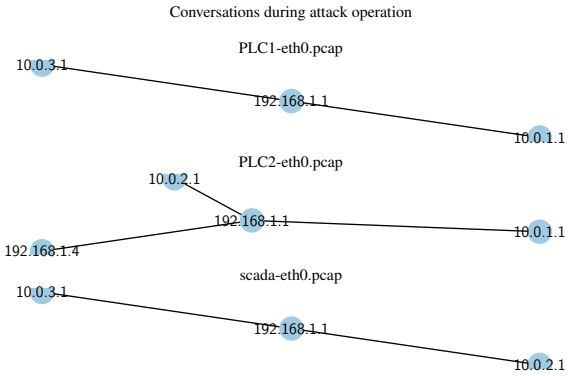
(a) Traffic sent over SCADA - normal.



(b) Traffic sent over SCADA - attack.



(c) Control communication patterns between different devices - normal.



(d) Control communication patterns between different devices - attack.

packet overview - normal operation						
capture name	**CIP**	**ENIP**	**ICMPv6**	**MDNS**	**ARP**	**Total**
Scada-eth0	33840	13536	47	20	2	*47445*
PLC2-eth0	36038	29270	46	20	2	*65376*
PLC1-eth0	42806	29270	47	20	2	*72145*

(e) Protocol packet count - normal.

packet overview - attack operation						
capture name	**CIP**	**ENIP**	**ICMPv6**	**MDNS**	**ARP**	**Total**
Scada-eth0	39360	15744	48	20	2	*55174*
PLC2-eth0	40654	31738	65	20	19	*72496*
PLC1-eth0	49588	33844	47	20	2	*72496*

(f) Protocol packet count - attack.

Figure 22: DHALSIM feature exploration: normal operation vs attack operation using WADI configuration.

the DHALSIM generated data *SWAT A6 2019* dataset includes both physical sensor and network data allowing for the development of both physical sensor-based detection and network-based detection.

Consequently, when attempting to perform the deep-packet inspection approach against the *SWaT A6 2019* which proved to be a suitable method for the custom-generated DHALSIM dataset analysis of the *SWaT A6 2019* dataset, confirms that the scale of the dataset is too large for efficient data analysis using deep-packet inspection (utilising an Intel i7-10750H @2.60 GHz Linux-based machine with 16GB RAM). Due to the vast number of control messages expected to be found in large-scale ICS architectures and the *SWaT A6 2019* dataset, network traffic captures can generate an extensive collection of data. Therefore, DPI should only be considered where an ICS architecture is limited in size or when computational constraints are not posing a problem.

As a result, we first attempted to perform a feature set exploration using Netflow V5 on DHALSIM to see if attacks could still be noticed using primarily Netflow features. In Appendix B.2 and Fig. 41, 42 one can find the DHALSIM feature set exploration on network traffic information considering Netflow V5 features. One downside we encountered when using Netflow V5 is that there is some loss of granularity. For example, visibility of the information on *ARP* packets is lost, and from the previous section in Fig. 22f, it could be seen that this might serve as a good attack identifier.

Considering the *SWaT A6 2019* dataset we obtain the following overview for the *Flows/sec*. It can be easily observed from Fig. 4 that the data exfiltration event, when mapped to the event timeline, leads to an increase in the number of flows measured over the entire SWaT architecture. For the *SWaT A6 2019* dataset, given the event timeline in Table 13 we consider the first 15 minutes of the dataset *normal*, as no attack is taking place on the testbed. However, we recognise that this is for the current version of the *SWaT A6 2019* dataset limited in size when considering the entire experimental dataset. As such, for the continuation of feature set exploration for the *SWaT A6 2019* dataset, we will consider the entire timeline to map each attack event to observations in the data.

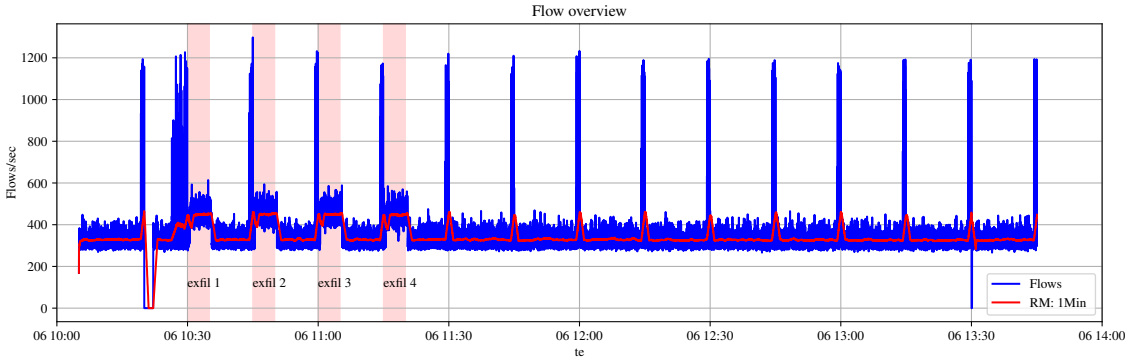


Figure 23: Flow/sec overview *SWaT A6 Dec 2019* dataset.

Even though the Netflow V5 method proved to be an effective method for identifying specific attack steps in the *SWaT A6 2019* dataset, other steps such as ”Second malware from C&C server” and disruption of the ”Sensor and Actuator” demonstrated to be more challenging lacking enough features for identification. As a result, we explored a third feature domain presented in the next section (Section 6.4).

6.4 Zeek-based Features

Ghaeini et al. [85] propose HAMIDS, a solution that integrates Zeek onto the physical SWaT testbed (not the previously identified SWaT datasets) to allow for distributed attack detection in a hierarchical SCADA system. We explore the capabilities offered by Zeek, as it has proven to be an effective method in the actual testbed in Singapore, on which the *SWaT A6 2019* dataset is based.

For the analysis of the *SWaT A6 2019* dataset and each attack step contained in the dataset we consider the following Zeek logs: *conn.log*, *dns.log*, *cip.log*, *arp.log*, *enip.log* with all the fields presented in Table 18 in Appendix B.1. The other default logging capabilities by Zeek are disabled in the settings. These include: *dhcp.log*, *ntp.log*, *cip_io.log* etc., the full configuration of Zeek and the custom parsing scripts can be found on Github [86].

As a starting point for developing a feature exploration using the Zeek features, we again start with the first step in the previously found event timeline, ”Exfiltrate Historian data,” where in the Netflow V5 approach, we observe that the exfiltration event in the *SWaT A6 2019* dataset leads to a suspicious increase in the number of flows. Using Zeek the logs for the exfiltration event, one can observe Fig. 24 where during the attack interval, an increase in the generated HTTP logs can be observed with a connection to the undocumented server at *IP:192.168.1.219*. In Appendix B.1 we provide more extended feature and dataset exploration for the *SWaT A6 2019* dataset using Zeek. Fig. 25a shows the host URI count for each identified Host in the *SWaT A6 2019*

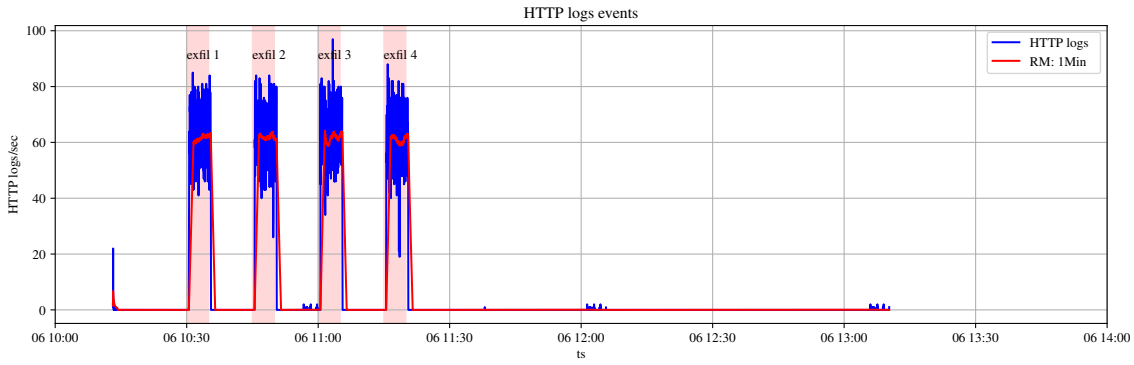


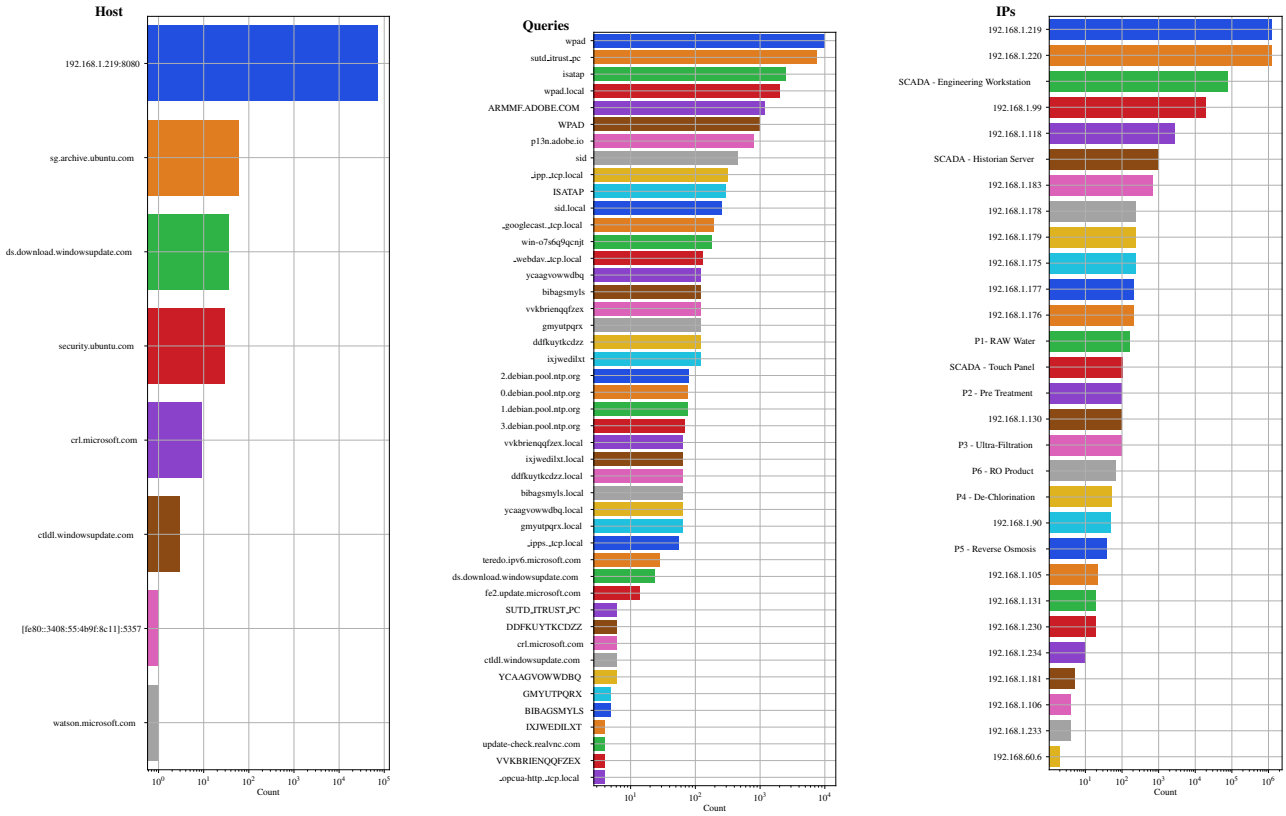
Figure 24: HTTP log events/sec overview - Zeek *SWaT A6.Dec 2019* dataset.

dataset. What is interesting is that besides the exfiltration host, we find background *HTTP* traffic contacting URIs from Microsoft.

Continuing our analysis we take a look at *dns.log* in Fig. 25b showing the *DNS* query count for all the unique *DNS* entries. In Fig. 25b we find besides benign looking *DNS* entries also ”random” looking *DNS* queries, for instance queries such as *”ycaagvowwdbq”*, *”ddfkuytkcdzz”*, the cause of this observation remains unknown, however, should be considered. Combining the observations in Fig. 25b and the previous *HTTP* URI hosts overview in Fig. 25a we observe that the *SWaT A6 2019* dataset contains next to exfiltration also other traffic including Windows updates traffic and traffic to Canonical (Ubuntu) making it arguably a representative dataset of a real-world deployment.

Since the documentation by the researchers of the SWaT testbed remains limited, we check what other *IPs* are present and are part of the testbed. Fig. 25c constitutes an *IP* count visualisation of all the *IPs* part of the *”192.168.xxx.xxx”* subnets assuming these *IPs* are an element of the measurement testbed or the SCADA control network. The naming of the known *IPs* documented in the Technical details V4.4 [77] is stated according to their component’s name. This shows that although documentation is present for the SWaT testbed and the *SWaT A6 2019* dataset, it remains a relative ”black box” since other researchers cannot obtain the role of these components in the testbed.

In the next step, we want to see which PLCs and other SCADA components are communicating with one another. Due to Zeek’s ability to specifically parse *CIP* communications used as ICS control protocol for the overlay SCADA network in the SWaT testbed, we map In Fig. 25d each CIP talking associated pair and map this to the known *IPs* mentioned in the SWaT technical details [77]. One interesting observation is the unknown *IP:192.168.1.99* also encountered in Fig. 25c which communicates with all the process PLCs of the six-stages found in Fig. 25d, but is not being mentioned in the technical details. This supports our earlier point that although we observe deviating observations in the data consecutive to the event timeline, we cannot state with 100% accuracy that the deviating patterns are, in fact, the attack steps.



(a) URI host overview based on *http.log*.

(b) DNS queries based on *dns.log*.

(c) Testbed IPs count based on *conn.log*.

(d) Control communication patterns between different devices extracted from the *cip.log*.

Figure 25: Feature and dataset exploration Zeek *SWaT A6 2019* dataset.

6.5 Understanding SWaT A6 2019 and Achieving Full Attack Visibility

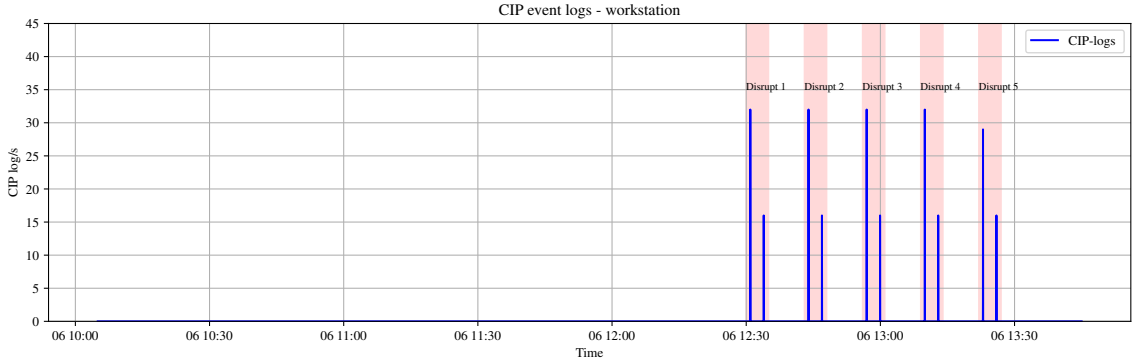
In this section, we aim to elaborate and discuss the *SWaT A6 2019* dataset and address our interpretation of this dataset using Zeek-based features to identify all the contained attack steps. Currently, only limited documentation that describes this dataset and the included attack steps is available. As a result, we identified each attack step found in the event timeline presented in Table 4 using a forensic or a security analyst approach such that we can establish full-attack visibility and provide future research with an interpretation of this dataset.

We try to map all the previously unknown attack steps in the event timeline and map this to deviating patterns that probably indicate the attack steps. We also attempt to determine the applied method. Each step is annotated with a Mitre ATT&CK ID or Mitre ATT&CK ICS ID to allow for the development of attack detectors around the *SWaT A6 2019* dataset. Note that there is an overlap between the conventional Mitre ATT&CK framework and the ICS framework.

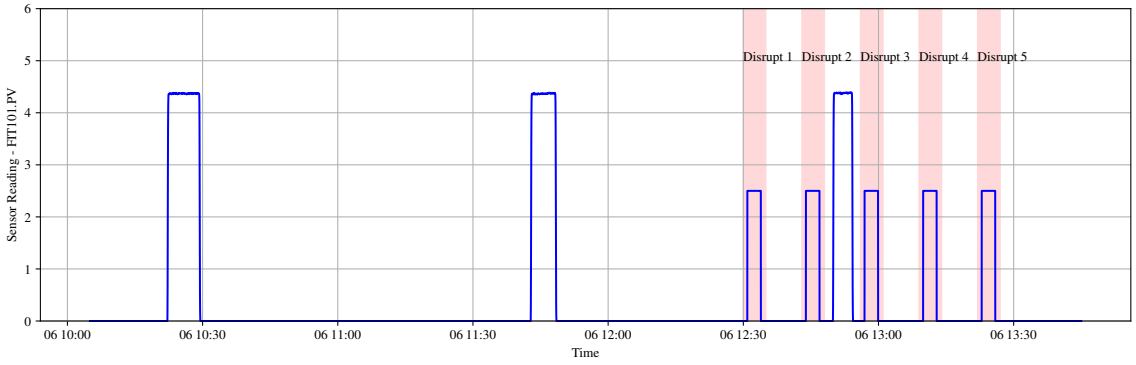
Fig. 27 depicts all the identified attack steps in the *SWaT A6 2019* dataset in a novel overview. The initial step we find in Fig. 27 is a malware infection through a USB stick. This first step can presumably not be identified in the network traffic information or the physical sensor data. However, this step in the attack chain has been pinpointed using the provided event timeline stated in Table 4. The malware placed on the workstation leads to an observed communication between the workstation and a server hosted on *IP:192.168.1.181* in the SWaT testbed immediately after the malware installation event. The method of communication between the server hosted on *IP:192.168.1.181* and the workstation remains unknown. We can only identify a *TCP* connection.

Following the initial compromise of the workstation through a malicious USB, exfiltration of the historian server is performed. The workstation starts performing *HTTP GET* request to the *Exfiltration* server depicted in step two (Fig. 27) hosted on *IP:192.168.1.219*. Different subdomains are being queried using *Python request user-agent* in the format of: */historian/current/D8CF2232/MV501*. Interestingly enough, during the exfiltration event, an increase in *HTTP GET* requests is visible while one would suspect a *POST* or *PUT* request method for data upload. We suspect that due to the workstation being in the same segmented network and on the same Purdue model level, direct access between the workstation and the data historian is possible and, therefore, is being used as a medium to perform the exfiltration. However, looking at the communication between the workstation and the historian server, we identify occasional *TCP* traffic, yet, we find no additional communication during the exfiltration event. For this reason, we are uncertain how exactly the exfiltration event between the workstation and the historian is performed.The next step we observe in the event timeline given in Table 4 is the second malware download event. During this interval, an increase in the response data of a connection with *IP:192.168.1.181* can be observed using *conn.log* information. Particularly for step 3 in Fig. 27 or the second malware download event, we observe an outlier response transfer of $10157166 Bytes$ presumably demonstrating the second malware download. This can also be noted in more detail in Fig. 45b in Appendix B.4 where the observed outlier is depicted. The connection during this increased communication is performed using a *TCP* connection with endpoint port number *6001*. It remains uncertain how this step is carried out as no documentation on the attack strategies is available. It is also worth noting that in addition to malicious traffic, the dataset also contains several benign connections, among them the previously seen Windows update traffic.

Following the second malware download, we observe right after the successful installation of the second malware an increase in the *CIP* traffic originating from the workstation targeting a PLC in the first process stage of the SWaT testbed. When these disruptions are mapped to the aforementioned event timeline in Table 4 we derive Fig. 26a presenting CIP write requests targeting the PLC with *IP:192.168.1.10* responsible for the control process of the Raw Water intake. Fig. 26b illustrates the impact of the CIP write commands on the physical sensor data for the *FIT101.PV* pressure valve sensor. Although the event timeline for the *SWaT A6 2019* dataset suggests an attack on multiple sensors or actuators in the physical process, only the influence on one sensor value could be identified in the dataset suggesting inconsistency in the current documentation or missed deviations for an actuating component.



(a) CIP log event workstation based on *cip.log*.



(b) Physical sensor value *FIT101.PV*.

Figure 26: Disruption Attack on Sensor visible in both the network data and the physical sensor values

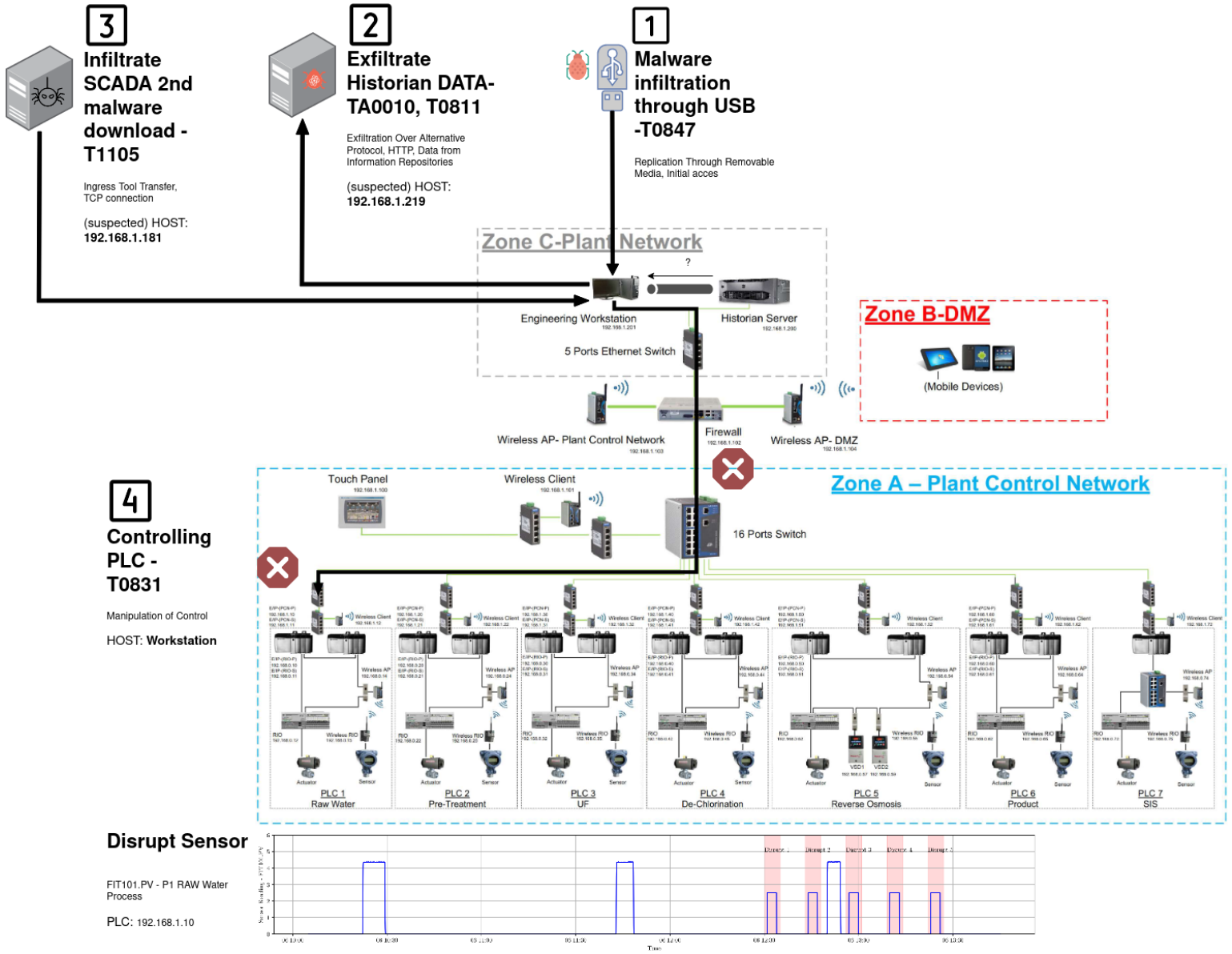


Figure 27: Attack steps in *SWaT A6 2019*.

When we map all the previous identified attack events for the *SWaT A6 2019* shown in Fig. 27 with the earlier proposed Spiral attack model by Hassanzadeh and Burket [27] we obtain Fig. 28 clearly showing that a realistic attack performed by an advanced adversary can comprise of multiple attack steps. In the case of Fig. 28 we observe that more than one attack event is detectable at a level higher than level zero, enabling early detection in contrast to purely physical sensor-based attack detection.

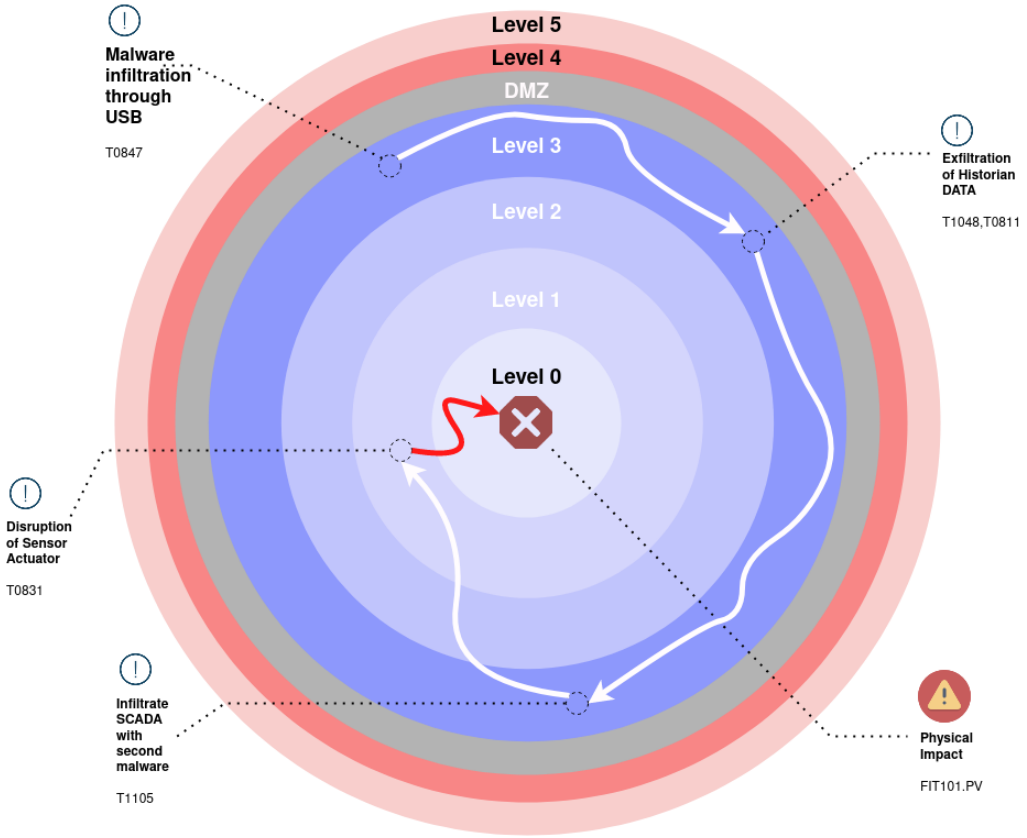


Figure 28: Mapping *SWaT A6 2019* to the Spiral Model.

6.6 Lessons Learned

Below we outline the essential findings during our dataset and feature set assessment:

– **Usage of Digital Twins**: To our knowledge, we are the first to employ data generated by DHALSIM. From this setup, we analysed both the network and physical sensor data. In this experiment, a question of fidelity arose concerning the degree to which the data from the digital twin matches real-world captured testbed data. Although the authors state that the digital twin provides high-fidelity data, no further analysis has been performed on the similarities between a real deployment and the synthetic dataset. For example, when we compared network traffic from a SWaT testbed and the digital twin, we encountered in the *SWaT A6 2019* dataset more benign background traffic such as Windows update traffic, *NTP* traffic, *DNS* queries, encrypted *TLS* traffic which is not present in the digital twin but is discernible in real-world deployments. The DHALSIM digital twin provides a cost-effective solution compared to real-world testbeds. However, other processes such as the observed *DNS* or update traffic can be challenging to integrate into a digital twin resulting in a lower fidelity compared to real-world testbeds. As such, developing detection approaches using digital twin data only goes as far as the data allows. This means that in the case of the DHALSIM digital twin, the detection methods can only be tested on available attack scenarios such as the currently available *direct attack* and the *MITM* attack scenario. One solution would be to extend the tool to include additional attack scenarios, but it will require considerable effort to add new attacks for the current version of DHALSIM. Secondly, a potential improvement would be to include simulated benign traffic such as

Windows update traffic, as we observe in the *SWaT A6 2019*, making the DHALSIM digital twin a better representation of a real-world ICS deployment. DHALSIM is thus a promising and novel method for ICS data acquisition. Still, it lacks enough built-in attack scenarios and detail to use as a primary data source for this research. Nevertheless, we would like to stress that this data acquisition method has great potential by being relatively easy to acquire ICS data under different attacks and scenarios.

– **SWaT A6 2019** is, for our research scope, the most realistic and complete dataset. The dataset contains several attack steps over different layers in the OT domain of the Purdue model. We were somewhat surprised by the limited use of this dataset in the current literature, mainly due to the completeness of the attack steps, which capture a realistic advanced persistent threat targeting an industrial control system. To the best of our knowledge, we are the first to employ both parts of this dataset for detection purposes, namely the physical data and the network information.

  Although we have established a complete overview of this dataset with all the consecutive attack steps, there remain uncertainties in the *SWaT A6 2019* dataset.

  – There are a large number of *IPs* present that we think are part of the testbed but are not mentioned in the Technical documentation. We suspect that these are part of the measurement setup to emulate attack entities to monitor testbed processes from outside (not part of the testbed itself) or take packet captures.

  – There is no information provided as to how the attacks are actually executed. For instance, a second malware download onto the workstation is performed as part of the attack steps. The documentation never mentions how the second malware is downloaded, how the control logic works, or what services or tools are being used to emulate the attack.

  – The *SWaT A6 2019* contains only limited normal operation data. An alternative solution would be to use normal operation data from a different experimental run on the SWaT testbed, such as the *SWaT A3 Jun 2017* dataset for training. However, it is unknown what updates the testbed has received over time, hampering the ability to substitute datasets.

  – In the event timeline in Table 4 one can see the Historian data exfiltration step. What remains uncertain is whether data originates primarily from the workstation or alternatively originates from Historian, which would require a connection between the Historian and the workstation. This could not be identified in our data analysis.

– **Feature-set**: Zeek provides the best trade-off between granularity and scalability [87]. Nonetheless, deep-packet inspection proved an inefficient method for large-scale ICS architecture since these architectures can generate a large amount of network traffic. As an alternative approach, we proposed using Netflow V5 features, commonly available in existing network equipment. However, this method lacks extensibility for industrial communication protocols and lacks the ability to parse *ARP* packets, something that proved to be a helpful feature in the DHALSIM data.

– **S7Comm Factory Dataset:** Even though the *S7comm factory* dataset contains both network and physical sensor data, no further attempt was made to delve deeper into this dataset, partly since the mining refinery architecture, in contrast to the SWaT testbed, represents a minimal architecture. In addition, the Swat A6 2019 dataset realistically covers a number of attack steps that attempt to descend from higher layers (level 3) in the Purdue model to lower layers (level 0). The *S7comm factory* dataset, on the other hand, contains mainly process and flooding attacks.

7 Mock-up Physical-based Detector

As previously presented in Section 4, current research on physical process-aware anomaly detection already has numerous established approaches, with the best performing approach being presented by Xu et al. [56]. Since our goal is to integrate both network-traffic ICS anomaly detection with physical-based attack detection, we assume for the scope of this research that physical-based detection is already proven effective. Therefore, this section will briefly explain how we assume this part to be effective, implying a perfect detector with an TPR (Recall) of 1.0 and FPR of 0.0. We also acknowledge that such detection performance is difficult to achieve in a real-world deployment or might only be achieved by supervised methods. However, the *SWaT A6 2019* dataset contains only one physical attack, and hence, for the scope of our research, it is reasonable and necessary to assume that this single attack event is successfully detected. Based on our previous *SWaT A6 2019* dataset analysis in Section 6.5, we have found that the attack effects on the SWaT testbed resulted in deviating patterns for the *FIT101.Pv* sensor affecting level 0 of the Purdue model. So now we assume a detector system is taking in the physical data from and raising alerts during the red-highlighted events given in Fig.26b and simply label the dataset for each disruption event as we observe in Table 4 accordingly. For the scope of this research, we implement the detector to raise an alert every 30 seconds during the labelled attack interval.

8 Development of Network-based Detectors

In the previous section, we have seen three attack steps in the event timeline that can be observed in the network data. During the data analysis of this thesis, we observed that specific attack events induce deviations in the patterns of generated logs. Assuming that the deviations indicate attack behaviour for the given network features, we propose three network detectors to detect these anomalies and effectively distinguish normal behaviour from attack behaviour for each previously identified attack event in the *SWaT A6 2019* dataset. Table 5 presents our proposed detection method for each of the events that may be detected in the network data. In addition, for the exfiltration event and sensor disruption event in Table 5, we propose a method to model the connection log generation pattern using *conn.log* information, using stateless differencing, i.e., the difference between the measured and the forecasted number of connections for a given component.

For the malware download event, we propose outlier detection using Empirical-Cumulative-distribution-based Outlier Detection (ECOD) [88] in combination with various Zeek features, as we have observed that this event leads to an outlying connection in terms of the exchanged bytes and packets.

Number	***Attack event***	***Observation***	***Proposed Detection Method***
1	Exfiltration event	Increase in observed HTTP traffic and logged connections at the given intervals for workstation. (see Table 4).	*SARIMA* with stateless difference
2	Malware download event	Large transfer of *10157166 Bytes* at the given event in Table 4 to the workstation.	*ECOD* Outlier Detection
3	Disruption Sensor Actuator event	Deviation in *ENIP* communication pattern at the given event interval in Table 4 from the workstation to PLC1 - Raw Water intake.	*SARIMA* with stateless difference

Table 5: Network detectable events in *SWaT A6 2019* dataset and proposed detection methods.

8.1 Preliminaries

Two of our detectors rely on the (Seasonal) ARIMA model for time-series forecasting and ECOD for outlier detection. Therefore, this subsection will briefly introduce these two algorithms and explain the required fundamentals. The first publication: *Time Series Analysis, Forecasting and Control* by Box and Jenkins popularised the ARIMA model [89] introducing a method that uses an iterative method of identification, estimation and model checking [90]. The method allows for analysing and forecasting time-series data. The ARIMA model consists of two components an Auto-Regressive part integrated with a Moving Average component [91, 92, 93] allowing for forecasting future values based on historical data. The general terms for the ARIMA model are given by p,d and q, where p indicates the order of the autoregressive components, d gives the degree of first differencing and finally, p specifies the order of the moving average. By incorporating both components we obtain Eq. 11 describing non-seasonal ARIMA. The AR components is part: $\phi_1 y^{'}_{t-1} + ... + \phi_p y^{'}_{t-p}$ and the MA elements is given by: $\theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q}$ where $y^{'}_t$ is the differenced series or change between observations at time t, ϕ_p is the parameter or operator for the AR model. Similarly, ε_t gives white noise and θ_q gives the operator for the MA model.

$$y^{'}_t = c + \phi_1 y^{'}_{t-1} + ... + \phi_p y^{'}_{t-p} + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \tag{11}$$

One variant of the ARIMA model is the Seasonal ARIMA model allowing for forecasting seasonal data with the general term given in Eq. 12 [91]. The additional hyperparameters include seasonal autoregressive parameter

P, seasonal difference order D, the seasonal moving average parameter Q [94] and lastly, m for the number observation for a full period.

$$SARIMA \underbrace{(p,d,q)}_{\text{Non-seasonal Component}} \underbrace{(P,D,Q)}_{\text{Seasonal Component}} \underbrace{m}_{\text{Number of data points for a period}} \tag{12}$$

For the Seasonal ARIMA implementation, we apply the timeseries forecasting framework Merlion [95] proposed by Bhatnagar et al. [94]. The open-source machine learning framework features a unified option for various univariate and multivariate forecasting models. One feature useful for our application is the built-in auto machine learning capabilities for determining the previously seen non-seasonal components, seasonal components and the period for the Seasonal ARIMA model: $(p,d,q)(P,Q,D)m$. For the methodology applied to determine the $(p,d,q)(P,Q,D)m$ for SARIMA, Merlion implements various tests such as the theta method proposed by Assimakopoulos and Nikolopoulos [96] to estimate m, the seasonal decomposition test for estimating D and d presented by Cleveland et al. [97]. For the remaining hyperparameters, p,q,P,Q grid search is performed by minimising the Akaike Information Criterion (AIC). Lastly, to improve efficiency for the hyperparameter optimisation process and improve training time, the autoML option implements an additional approximation technique where a selection of good performing models are chosen after a few iterations which are then retrained until good AIC performance is achieved. For further details on the implementation of the Merlion framework, we refer to the paper by Bhatnagar et al. [94].

For the second detector type discussed later in this chapter, we use the unsupervised outlier detection method based on ECOD proposed by Li et al. [88]. According to Li et al. [88] existing unsupervised options often suffer from high-computational requirements, which can be a challenge in ICS architectures. Instead, ECOD provides an effective and scalable unsupervised outlier detection method. ECOD works around computing a univariate empirical cumulative distribution function for each dimension and computing the tail probability across all dimensions [88]. As reported by Li et al. [88] ECOD has a few advantages. First, ECOD is efficient with a complexity of $O(nd)$ where n is the number of samples, and d is the input data dimension. Second, the results of ECOD indicate good performance on 30 benchmark datasets outperforming 11 baseline popular outlier detection methods. Finally, ECOD provides easy interpretability allowing the ability to look at estimated probabilities for each data point. ECOD is implemented in the Python Outlier Detection (PyOD) framework introduced by Zhao et al. [98], and available on Github [99].

8.2 Seasonal ARIMA Attack Detector

Fig. 29 provides a high-level overview of the exfiltration and sensor disruption detection pipeline. The pipeline consists of two phases: first, a learning phase where we assume that during this phase, no attack is introduced in the model and only the system's normal behaviour is expressed. In the second phase, we can feed our detector malicious intervals containing the previously seen attack steps timeline given in Table 4. Here our core idea is to make several models for each individual testbed component in the ICS architecture (reported in Table 6) and not to monitor an aggregate of communication behaviour on the overall ICS architecture. The "filter for service" block in Fig. 29 makes it possible to select the connections based on service, e.g. *http* or *enip* thereby enabling better granularity for the communication of certain services in the detection process. In Zeek, these service parsing capabilities can be extended with additional external packages or custom scripts. Finally, in the "normalise block" we apply MinMax scaling given in Eq. 13 transforming the features to a *[0,1]* range.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{13}$$

Although the idea of using ARIMA to predict physical sensor values and other system parameters for detection is not new [57], we aim to take a different approach where we learn the systems communication pattern based on the connection event in Zeek's *conn.log(s)*. The testbed's control messages and benign traffic can be expressed

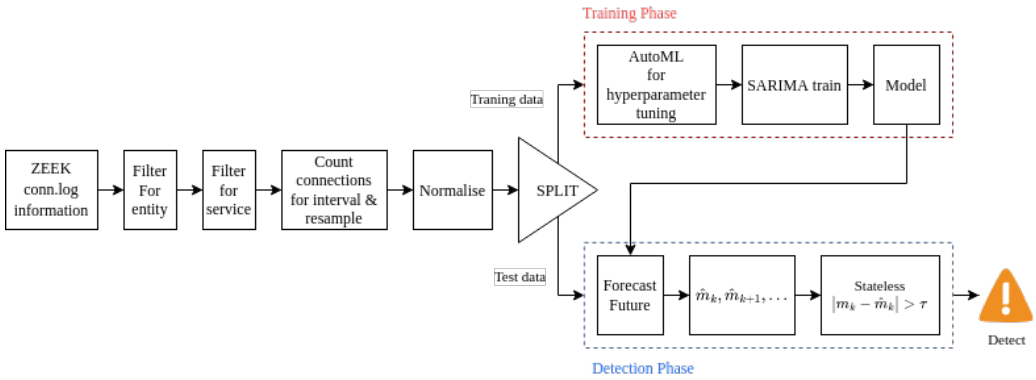


Figure 29: SARIMA based detection pipeline.

by u_k. In Zeek, each connection is logged in the *conn.log* and resembles communication behaviour m_k or the number of connection logs given at point k for an entity or ICS component. The control messages and benign traffic in the testbed can be expressed by u_k resulting in m_k connection logs such that:

$$u_k = m_k \tag{14}$$

Based on the connection events, we count the number of logs for a given entity for the specified time unit such that we can obtain, e.g., the number of unique $connections/s$ as depicted in Fig. 30. Given the previous number of connections m_k, m_{k-1} up to m_{k-n} we learn a Seasonal ARIMA model using the Auto hyper-parameter capabilities of Merlion on the historical connection count, allowing us to forecast the number of connections logs for a given entity as we observe in Eq. 15.

$$\hat{m}_k, \hat{m}_{k+1}, \hat{m}_{k+n} = SARIMA(m_k, m_{k-1}, m_{k-n})(p,d,q)(P,D,Q)m \tag{15}$$

Here Seasonal ARIMA model is applied to learn the normal communication patterns for a set of specified entities in the SWaT testbed for both connection directions: outgoing and incoming. Using stateless detection, we raise an alarm if the condition in Eq. 16 is met where τ is required to be manually tuned. In the scope of this research, threshold values are visually tuned.

$$|m_k - \hat{m}_k| > \tau \tag{16}$$

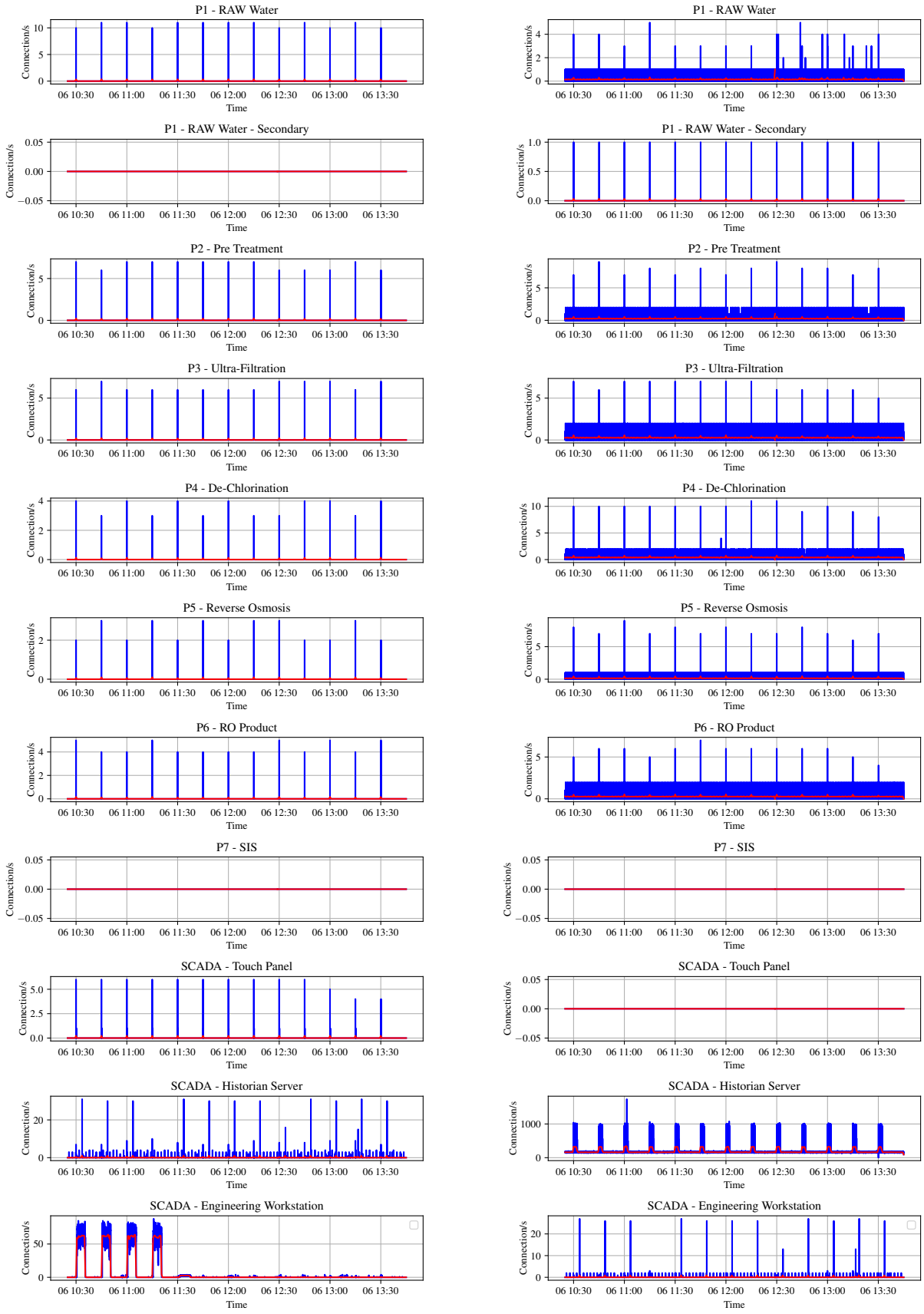
During an attack, we expect the observed communication behaviour to deviate from the learned normal patterns. If a component's predicted number of connections exceeds a certain level, we classify it as anomalous behaviour. This procedure can be adopted at different locations and for different components in an ICS architecture. For example, we monitor the behaviour of every reported (and known) testbed component. The current implementation forecasts $\hat{m}_{k+n}$ where n is equal to the length of the test data.

System Component	*IP*
PLC1 - RAW Water Intake	192.168.1.10
PLC1 - RAW Water Intake - Secondary	192.168.1.11
PLC2 - Pre Treatment	192.168.1.20
PLC3 - Ultra-Filtration	192.168.1.30
PLC4 - De-Chlorination	192.168.1.40
PLC5 - Reverse Osmosis	192.168.1.50
PLC6 - RO Product	192.168.1.60
PLC7 - SIS	192.168.1.70
SCADA - Touch Panel	192.168.1.100
SCADA - Historian Server	192.168.1.200
SCADA - Engineering Workstation	192.168.1.201

Table 6: Overview of reported testbed components present in the dataset.

8.3 Malware Download Attack Detection

To identify the second malware download, we intend to introduce a method that inspects single connections to identify malicious file transfers. The previously mentioned SARIMA method can detect an aggregate of connections and is believed to be effective in identifying deviations in the communication profile. However, it is ill-suited for identifying single-outlying connections required for identifying malicious downloads. As such, we apply ECOD for every single connection with the following features: *orig_ip_bytes*, *resp_ip_bytes*, *orig_pkts*, *resp_pkts*. Next, Fig. 31 depicts the general outline for the anomalous connection detector. To detect the anomalous connection, we select the following *conn.log* features: *resp_ip_bytes* giving the number of IP level bytes that the responder sends. The *resp_ip_bytes* corresponds to the next selected feature: *orig_ip_bytes* or the number of IP level bytes that the originator sends. The last two features we use as input are *orig_pkts* and *resp_pkts* indicating the number of packets in both directions associated with a logged connection. Similarly to the SARIMA-based detector, the pipeline consists of two phases. In the first phase, we train a model for the specified system component based on normal system behaviour for the four selected features. In the second phase, we feed each component' ECOD model malicious test data and obtain any deviating connections. If required, the returned anomalies can be sorted according to PyOD's provided anomaly score.



(a) Filtered based on *id.orig_h* or outgoing connections.

(b) Filtered based on *id.resp_h* or incoming connections.

Figure 30: Connection logs/s for each known testbed component - *conn.log*, Blue: connection logs, Red: RM 30s.

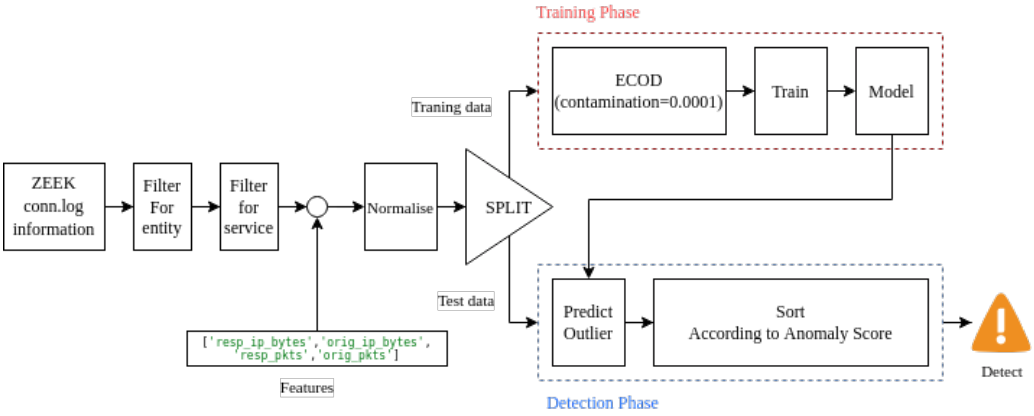


Figure 31: ECOD outlier-based detection pipeline.

9 Evaluation of Individual Detectors

This section will evaluate our proposed attack detectors against the *SWaT A6 2019* dataset. First, we discuss how we derive the labels for this dataset based on the event timeline given in Table 4. Next, we individually evaluate the two detector types for the assigned event steps such that we test the SARIMA communication detector against the *Exfiltration* event and the *Disruption Sensor & Actuator* event. Likewise, we will test our *ECOD* outlier detector designed for identifying deviating connections against the *second Malware download event*. One challenge of evaluation against the *SWaT A6 2019* dataset is the lack of "normal" behaviour. Although the researchers who published the *SWaT A6 2019* data stated that the initial 15 minutes represent normal system behaviour, we found that the initial 15 minutes do not capture enough data to represent the entire system behaviour. Therefore, to compensate for this lack, we manually select training and test data intervals, which will be reported accordingly.

The training and evaluation are performed on an Intel i7-10750H @2.60 GHz Linux-based machine with 16GB RAM.

9.1 Labelling of Dataset

A common challenge in ICS cyber security research is the lack of a labelled dataset. The previously investigated *SWaT A6 2019* dataset lacks the assignment of labels for the provided packet captures. Instead, the dataset is *partially* labelled where we are provided with a timeline of malicious event intervals. Therefore, we must perform labelling to verify our proposed attack detection system and derive quantitative metrics from getting a truthful baseline. Since no additional reporting is available for *SWaT A6 2019* dataset, we are required to assume that the previous found network traffic deviations consecutive in the event timeline in Table 4 are indeed the attack patterns. Hence, for this part, we have to assume that Fig. 27 holds true and fully captures the attacks performed in the experimental capture at the given events.

For our labelling approach, we label the attack ranges for the earlier identified physical and network deviations depicted in Fig. 27. Events labelled with 1 indicate anomalous events. If no attack event is taking place, we label it with 0, resulting in a label sequence such as, `[0,1,1,0,0,0]` for a given system component. Since the connections can be incoming or outgoing, we label them in both directions. Table 7 gives the labelling intervals for all the components mentioned in the Technical details V4.4 [77], which are present in the dataset and show malicious behaviour. So, in case of Disrupt 1 event, outgoing connections from SCADA Engineering Workstation to PLC1 - RAW water Intake between *12:30* and *12:33* are labelled as malicious. In the same way, on the incoming side for PLC1 - RAW water Intake, we label connections from the SCADA Engineering Workstation as malicious at the given attack event period. However, certain attack events, such as the Exfiltration event, are not detectable in both directions as these involve a connection to an entity outside the testbed. Therefore, for the rest of the known testbed components, we label the connections as benign.

Attack		***Labelling Start***	***Labelling End***	***Impacted***	***Connection Direction***
Exfiltration	Exfiltration 1	10:30:00	10:35:00	SCADA Engineering Workstation	outgoing
	Exfiltration 2	10:45:00	10:50:00	SCADA Engineering Workstation	outgoing
	Exfiltration 3	11:00:00	11:05:00	SCADA Engineering Workstation	outgoing
	Exfiltration 4	11:15:00	11:20:00	SCADA Engineering Workstation	outgoing
Malware Download		12:30:45	12:30:46	SCADA Engineering Workstation	outgoing
Disruption Sensor & Actuator	Disrupt 1	12:30:00	12:33:00	SCADA Engineering Workstation ↓ PLC1 - RAW water Intake	outgoing, Incoming
	Disrupt 2	12:43:00	12:46:00	SCADA Engineering Workstation ↓ PLC1 - RAW water Intake	outgoing, Incoming
	Disrupt 3	12:56:00	12:59:00	SCADA Engineering Workstation ↓ PLC1 - RAW water Intake	outgoing, Incoming
	Disrupt 4	13:09:00	13:12:00	SCADA Engineering Workstation ↓ PLC1 - RAW water Intake	outgoing, Incoming
	Disrupt 5	13:22:00	13:25:00	SCADA Engineering Workstation ↓ PLC1 - RAW water Intake	outgoing, Incoming

Table 7: Labelling strategy *SWaT A6 2019* - Dataset Range: *2019-12-06 10:05:00* to *2019-12-06 13:45:00. Data between start and end for the given component is labelled as 1.0, the remainder of the range is labelled 0.0.*

9.2 Evaluation Approach

Since the label assignment is based on intervals rather than having pre-labelled data points, our evaluation procedure will be different as we can not simply compare. To obtain evaluation metrics, we consider an anomaly to be successfully detected if at least one anomaly alert is triggered within the given event interval, provided in Table 7. If an anomaly is raised within the attack interval, we count all labels within that particular as correctly detected. This procedure is applied since the proposed method monitors on a connection basis, and during the data analysis, we have observed that in some cases, an attack can consist of only one or two control command connections within the given attack event interval. This implies that we would under-evaluate if we compare the true labels with the prediction labels naively. This approach is also depicted in Fig. 32 where we use the true labels in the blue section if at least one (red) anomaly is raised between t_1 and t_2.

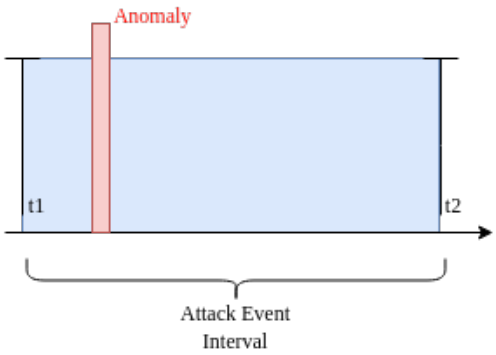


Figure 32: Evaluation approach.

9.3 Performance of Individual Detectors

First, we evaluate the individual performance of detectors against the three known network detectable event: *Exfiltration* event, *(Second) Malware Download* event, *Disruption Sensor Actuator* event. During the study, we used manual tuning of the threshold by means of optimising results visually. As such, we apply a threshold value of $\tau = 0.35$ for all SARIMA detection experiments, a value obtained by visual and manual tuning using the difference plots in Fig. 48 and Fig. 35. In Appendix C.1 an overview of the timing intervals we use for splitting the data, the training times and the hyper-parameters for the models used for the detector is given. Again, it is important to note that our detectors are tested in both ways (*Outgoing* $\longleftrightarrow$ *Incoming*) for each of the known testbed components.

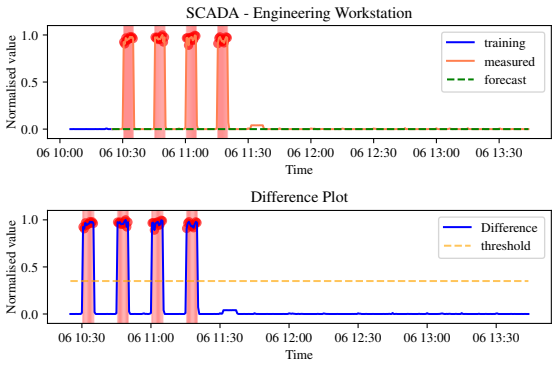
9.3.1 Evaluation on Exfiltration of Historian Data Event

To evaluate the SARIMA-based detector against the Exfiltration event, we selected the input ranges given in Tables 19a, 19b in Appendix C.1 for all the (known) system components reported in the Technical Detail V.4.4. One downside of the *SWaT A6 2019* dataset for this experiment is the limited available normal behaviour data. To ensure our detector captures and learns the correct system behaviour, manual training and test intervals have to be selected for the SCADA - Engineering Workstation. On the outgoing side, for the SCADA - Engineering Workstation, we have to select a shorter training interval than the other systems components to ensure the right system behaviour is learned, which is not optimal. Moreover, for this detector, we learn the system behaviour of all the *service* connection logs, i.e., *HTTP* connections, *ENIP* connections, *DNS* connections among others. Fig. 46 in Appendix C.2 shows the detector results, training data, measured observation and forecasting for all the testbed components.

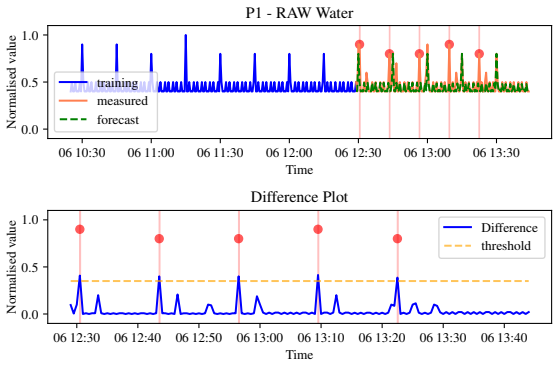
Fig. 33a shows the raised alerts for the SCADA - Engineering Workstation, where the four network deviations we observe during the exfiltrations event are all successfully detected. At the bottom of Fig. 33a we depict the difference plot indicating the *Stateless* difference between the measured and the forecasted number of connections for the test data. As can be observed, when the difference exceeds the threshold of $\tau = 0.35$ we

raise an alert. One detection, which is also picked up by the other detectors discussed later but not labelled as an attack event, is a connection around *13:36* shown in Fig 33c. However, as this connection is not labelled within an attack interval, we consider it a *False Positive*.

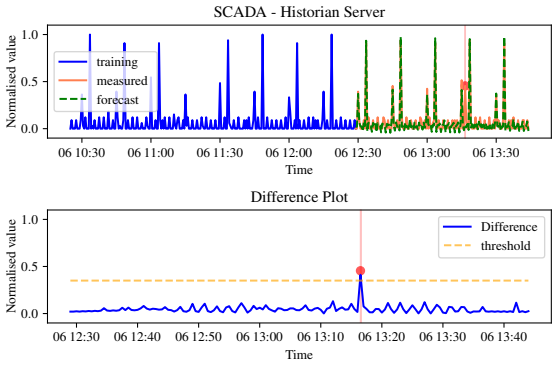
Knowing that the outgoing network connection during the observed exfiltration event is routed to an entity outside the testbed, we can assume that our detector would not pick up the exfiltration event on the incoming side of the testbed devices. However, looking at Fig. 33b, we notice several detections indicated by the red vertical lines and red points for the PLC1 - RAW Water Intake component. We find that these detections are correct as they are part of the *Disrupt Sensor & Actuator* event and match Fig. 35b, therefore, we consider these as correctly identified. At the same time, for the SCADA - Engineering Workstation, we again observe detected connections around *13:36* which we consider as *False Positives*. The time required to automatically optimise the hyperparameters and learn the underlying SARIMA model for each system component took, for the outgoing detectors, *18* minutes and 42 seconds. On the incoming side, optimisation time and training time took *9* Minutes and 52 Seconds.



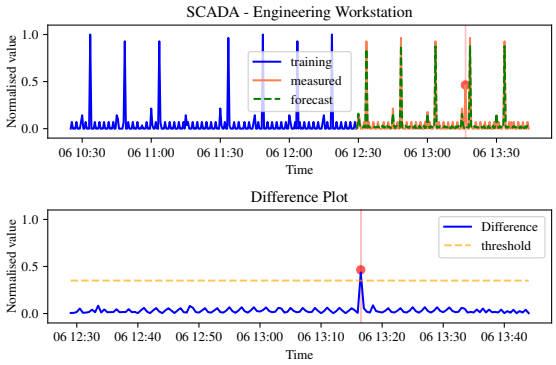
(a) Exfiltration Detection: *SCADA - Engineering Workstation id.orig_h* (Outgoing).



(b) Disruption Detection: *PLC1 - Raw Water Intake id.resp_h* (Incoming).



(c) Unknown event Detection: *SCADA - Historian Server id.orig_h* (Outgoing).



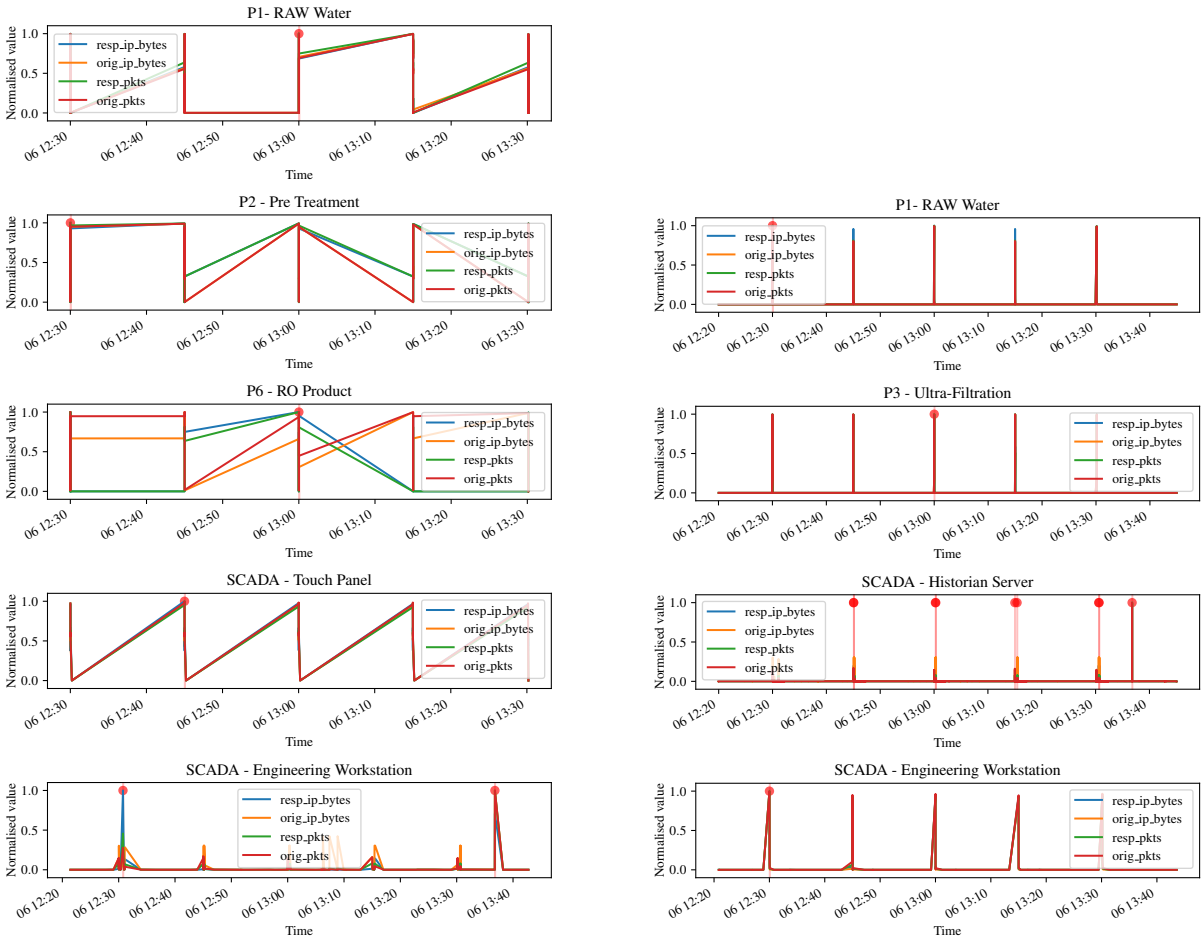
(d) Unknown event Detection: *SCADA - Engineering Workstation id.resp_h* (Incoming).

Figure 33: Raised Alerts for SARIMA Detector on *Exfiltration* event.

9.3.2 Evaluation on Second Malware Download Event

In the second evaluation experiment, we test our proposed ECOD connection outlier detector against the *Second Malware Download* event. Based on the data analysis section 6.5, we observed that the second malware download event is a connection initiated from the SCADA - Engineering Workstation and should therefore be detected on the outgoing side. Looking at Fig. 34a we observe that the *Second Malware Download* event at roughly *12:30* is successfully picked-up by our ECOD-based detector. Besides successfully detecting the attack event, we observe that the detector also picks up five benign connections, one of which is the connection(s) around the *13:36*. This means that five *False Positives* are generated beside one correctly identified connection, suggesting that

these connections indicate behaviour that deviates from the norm. Similarly to the *Exfiltration* event, the *Second Malware Download* is initiated towards an entity outside the testbed. For that reason, we suspect no detection on the incoming detector side depicted in Fig. 34b. We observe in Fig. 34b twelve *False Positives* one of which is again a detection around *13:36*, but also, incoming connection previously to the *Second Malware Download* event at *12:30*. It remains uncertain whether these connections are part of the *Second Malware Download* event. Hence we consider it a *False Positive*. Although 12 False Positives seems to be a high number, it should be noted that the detector examined 932379 individual connections. Another observation indicates that the training time of ECOD is very efficient even with a large set of samples. This can also be observed in Table 19e in Appendix C.1 presenting the training duration of ECOD for all the systems components. Fig. 47 in Appendix C.3 shows all the results for each individual system component.



(a) Detections: *id.orig_h* (Outgoing).

(b) Detections: *id.resp_h* (Incoming).

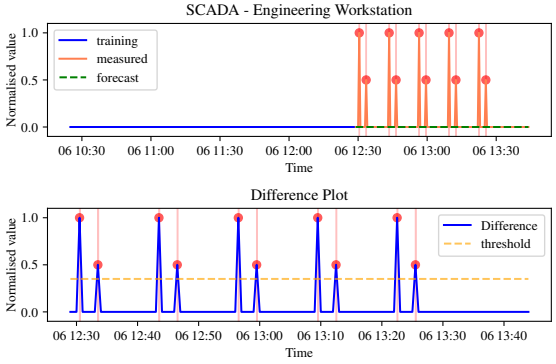
Figure 34: Raised Alerts for ECOD Detector on *Second Malware Download* event.

9.3.3 Evaluation on Disrupt Sensor Actuator Event

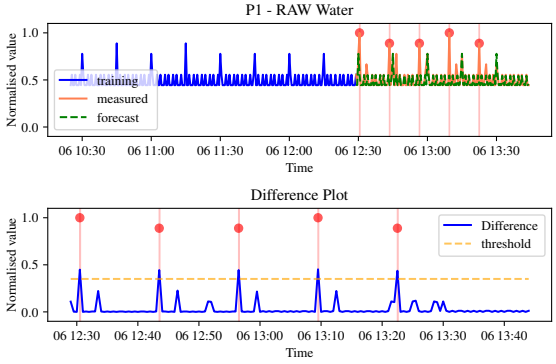
In the last individual experiment, we applied the SARIMA-based detector against the *Disrup Sensor & Actuator* event. This attack originates from the SCADA - Engineering Workstation and is launched against the PLC1 - RAW Water Intake. Therefore, this attack should be detected on the outgoing side of the *SCADA - Engineering Workstation* and on the incoming side of the PLC1 - RAW Water Intake. For the experiment, a communication model is built specifically for the *ENIP* traffic for each of the testbed's components, i.e., we filter the *conn.log* for *ENIP* communication.

Looking at Fig. 35a showing the experimental results, we can observe that the detector correctly identifies the Disruption Events instructed from the SCADA - Engineering Workstation. In the same way, Fig. 35b indicates

that the incoming disruption connections for the PLC1 - Raw Water Intake are correctly picked up by the SARIMA detector with no additional False positives. For both figures, we plot the stateless difference in relation to the set threshold ($\tau = 0.35$). In Fig. 48 in Appendix C.4 one can find the forecasting and anomaly detection for each of the testbed's component which indicate no further detection for the other components.



(a) Disruption Detection: SCADA - Engineering Workstation *id.orig_h* (outgoing).



(b) Disruption Detection: PLC1 - RAW Water Intake *id.resp_h* (Incoming).

Figure 35: Raised Alerts for ECOD Detector on *Sensor, Actuator Disruption* event.

9.4 Results Individual Detectors

Table 8 summarises the performance for the three different attack events against the two detector types and shows that all attacks are successfully detected with relatively low fall-positive rates for all detector types. Next, we observe that the overall detection delay or the time between the commencement of the attack and detection for the SARIMA detector approach is roughly equal to the resampling time used in the SARIMA detector. This suggests that in a real-time scenario, a detection delay of 30 seconds should be accounted for. However, in a post-mortem context, this would not pose a problem. Finally, we observed that a resampling time of 30*s* (calculated using the Rolling Mean method, Eq. 10 gave the best results for Merlion to automatically determine the hyperparameters of the underlying SARIMA model. A lower resampling of the data will lead to higher "noise" in data which can be harder for Merlion to automatically determine, requiring manual hyper-parameter tuning.

Next, in Table 8 we observe that the ECOD detector presents a higher number of false positives than the SARIMA-based detector. However, this result likely has to do with the low number of training samples, as shown in Table 19 in Appendix C.1, the selected features for which we want to identify the *Second Malware download* event or the fast number of connection it is evaluated against. The ECOD-based detector's false positive rate remains relatively low because the ECOD detector is evaluated on an individual connection, unlike the SARIMA detector, which operates on an aggregate of connections. Moreover, the results indicate that ECOD is scalable for larger data sets. In both cases, the maximum training time for the two directions is less than one second. In contrast, SARIMA hyper-parameter estimation and training are in the order of minutes. However, the number of training samples is still on the low side.

Name Attack	***Direction***	***TP***	***FP***	***FN***	***TN***	***Unit***	***Accuracy***	***Recall (TPR)***	***FPR***	***Detection Delay***
SCADA workstation - Exfiltration	Outgoing	52	1	0	1856	Number resampled 30s bins	0.9995	1.0	0.0005	roughly 30s (resampling time)
SCADA workstation - Exfiltration	Incoming	0	1	0	1660	Number resampled 30s bins	0.9994	-	0.0006	roughly 30s (resampling time)
SCADA workstation - Malware download	Outgoing	1	5	0	784	Individual connections	0.994	1.0	0.0063	Speed of ECOD: $< 1s$
SCADA workstation - Malware download	Incoming	0	12	0	932367	Individual connections	0.99	-	0.0000	Speed of ECOD: $< 1s$
SCADA workstation - Disruption	Outgoing	45	0	0	1616	Number resampled 30s bins	1.0	1.0	0.0	roughly 30s (resampling tim)
PCL1 - Raw Water Intake - Disruption	Incoming	45	0	0	1616	Number resampled 30s bins	1.0	1.0	0.0	roughly 30s (resampling time)

Table 8: Performance of individual detectors against Attack Event. The number of samples deviated among SARIMA-based detectors due to manual interval selection.

10 Merging of Detector Events

This section will discuss the concept of alert integration or merging alert information to achieve full-attack visibility using attack strategy reconstruction on ICS architectures. The individual detectors from Section 8 can already contribute to the detection process by integrating them at strategic positions in the Purdue Model. On the other hand, we find that some detectors still introduce false positives and observe that individual detectors alone are incapable of recognising the complete attack path shown in Fig. 28. Hence, we would like to have a method that captures the attacker's modus operandi while reducing the overall number of generated false positives.

The works of Koucham et al. [13] and the work by Wu and Moon [14] support this idea and argue that alert correlation could reduce false positives, remove redundant alerts, trace back to the root cause or reconstruct attack scenarios. The paper by Bin and Ghorbani [100] introduces a method that can extract attack strategies from a large volume of intrusion alerts. Using several similarity features, the output of a multi-layered perception and a support vector machine alerts are correlated. The correlated attacks are used to generate an attack graph providing the ability to extract the attack strategy of a multi-stage attacker automatically. However, their evaluation is limited to the classical IT domain and does not cover a multi-domain setting as one would encounter in ICS architectures. The paper by Wu and Moon [14] does cover a cyber-physical environment. They propose a method consisting of three steps to detect cyber-physical attacks. First, cyber alerts are correlated based on a set of similarity features. Then, in the same way, they perform physical-based alert correlation using physical features. Finally, in the third stage, they integrate the correlations of both domains, establishing a bridge between the cyber and the physical domain. Although the results indicate that the method can greatly reduce the number of false positives, the attack strategy of a multi-stage attacker is not considered or extracted.

In this section, we propose a step-wise approach that focuses on a method to automatically extract the attack strategy and reduce the false positives based on the previously raised alerts. Fig. 36 depicts our method divided into four phases. Phase 1 is the detection phase, where we collect all the alerts from our previously proposed detectors. In phase 2, we pre-process to enrich alerts with additional data, rendering alerts suitable for correlation. For this phase, the challenge lies in the multi-domain alert types: physical alerts and network-based alerts. After obtaining the alerts from the pre-processing phase, we recursively extract in phase 3, a tree of the associated alerts starting from the youngest obtained alert towards the oldest stored alert, i.e. building a tree descending in time. For alert correlation, we apply a similarity-based method using IP-address and *communicates with* field of the raised alerts as the primary attributes. Additionally, as the time component plays a central role in our approach, our alert correlation method can also be partially considered a sequential-based method, where alerts are linked on both a temporal basis and with the use of similarity [14].

Our proposed alert tree approach stems from the work of Kawakani et al. [101], who presented a method that correlates alerts and clusters alerts within the IT domain to identify the most typical attack strategy. The results indicate that the method can mine useful information or extract a strategy graph. Applying some of their ideas to the OT domain, we extend and build, in phase 3, an alert tree based on each unique system component in the testbed for which an alert is generated (See Table 6). In addition, we correlate different alert types from the physical and cyber domains by *timestamp, IP*, and *Communicates with* field.

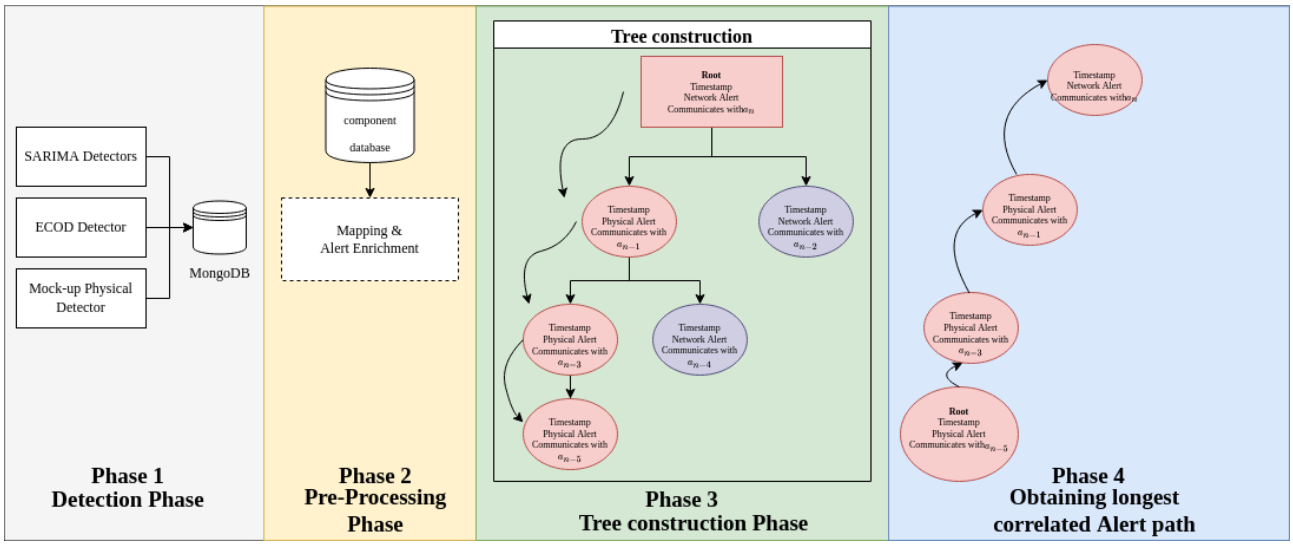


Figure 36: Going from full visibility to full attack visibility a step-wise approach.

10.1 Detection Phase

As shown in Fig. 36 we obtain two different types of alerts: a_{netw} or network-Based alerts triggered by the detectors discussed in Section 8.2 and a_{phy} physical-based alerts given by the mocked-up physical process-aware detector discussed in Section 7 such that we obtain $a_{phy}, a_{netw} \in \mathcal{A}$ where $\mathcal{A}$ represent the collection of generated alerts. We express each alert domain-bound alert differently. For the physical alerts, we introduce the attributes as given in Eq. 17, where timestamp gives the time when an alert was raised, component gives the physical component for which anomalous behaviour was detected, e.g., PLC 1 - RAW water Intake. Similarly, the next attribute gives the sensor for which anomalous behaviour is measured or triggered; a PLC can control various sensors or actuators. Finally, alert origin gives the detector for which the alert was triggered.

$$a_{phy} : [(\_id :), (timestamp :), (component :), (sensor :), (detector :)] \tag{17}$$

$$a_{netw} : [(\_id :), (timestamp :), (IP :), (communicates\,with :), (direction :), (detector :), (alert\,origin :)] \tag{18}$$

Network alerts are in the format of Eq. 18 with IP indicating the component address for which an alert is raised. *Communicates with* gives the IP address for the component it is talking to. This cannot be obtained for the proposed SARIMA, which works on an aggregate of connections. However, since we monitor for SARIMA detectors on inbound and outbound connections for each component in the testbed, we can correlate alerts such that we can deduce the *Communicates with* field for alerts triggered within the testbed. The deduction step be further addressed in Section 10.2. For ECOD-based detectors, alerts are triggered for separate connections allowing to obtain the *Communicates with* field. The alert origin is similar to the above, and direction gives the orientation of the detected alert. We monitor for the network-based detector both ways: incoming and outgoing. All the generated alerts in our implementation are collected in a MongoDB database [102], which we use in combination with the Python library Marshmallow for serialisation and deserialisation alerts into objects, each with a unique identifier. MongoDB [102] provides an easy data storage solution with support for the time-series data our project's alerts represent. However, other storage or Security information and event management solutions are suitable here as well, such as PostgreSQL or InfluxDB.

10.2 Pre-processing Phase

In the pre-processing, we map the physical-based alerts to the cyber-domains such that we have a mapping function $\prod_1$ allowing us to associate each sensor or physical component with a network address which we can

use later for correlation. Since we know the PLC for a given sensor component based on the testbed, we can enrich a_{phy} with an ip associated with the network address of a PLC as shown in Eq. 19.

$$a_{phy} \xrightarrow{\prod_1} [(unique\_id :), (timestamp :), (component :), (sensor :), (detector :), (\mathbf{IP} :)] \tag{19}$$

Since the SARIMA-based detector works based on an aggregation of logs, the communication cannot be deduced from the connection log. We can partially deduce the *communicates with* field by monitoring both ways for the same detector. Therefore, Network-based alerts raised by the SARIMA detectors can be enriched by a mapping function $\prod_2$ where we assume in our implementation the following premise: if an alert is triggered by the same type of detector at the same moment within the testbed (by means of monitoring two ways), we enrich SARIMA the *communicates with* field for both alerts. Fig. 37 gives a concrete example of the pre-processing step showing how we correlate alerts for the same detector within the test. This method only works for monitoring within the same testbed. Alerts for connections to external IPs that are not included in the monitored components can not be associated.

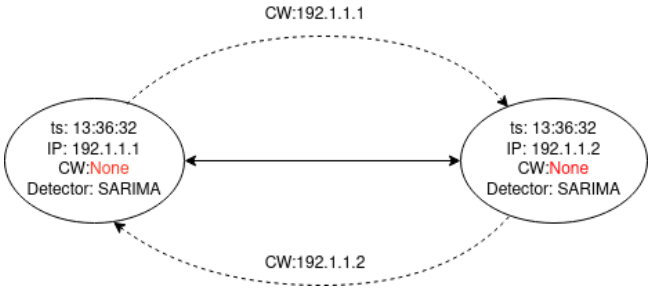


Figure 37: Enriching SARIMA alerts with communicates with.

10.3 Building Alert Tree

After the alert enrichment of the physical alerts in the previous step, we continue to build a binary alert tree recursively based on correlated alerts for each component in the testbed for which an alert is raised. Our approach correlates alerts by IP address, exploring an entire communication tree.

We start by setting the latest alert associated with a component as the root Node. The next child node is in for the alert set if the next node is the most recent and can be correlated if, for example, $a_n.communicates\_with = a_{n-1}.IP$, $a_{n-1}.communicates\_with = a_{n-2}.IP$ or $a_{n-2}.IP = a_{n-3}.IP$. The unique identifier associated with an alert is stored in a cache to avoid an explosion in the possible paths and enforce a binary tree. If an alert (ID) has already been explored in a path, the algorithm will omit this alert. The third condition to hold while building the alert tree is the requirement that the child node is older or equally as old as the parent node such that: $a.ts_2 \leq a.ts_1$. The pseudo-code of the recursive search of the alert tree is given in the Algorithm 1. We call the tree builder algorithm for each testbed component with an associated alert to obtain $b \in \mathcal{B}$, indicating a binary alert tree in a set of retrieved alert trees. Fig. 38 shows the process of constructing a binary tree for the component with *IP:192.1.1.1* according to the simple alert tables given in Tables 9a and 9b. Fig. 39 show the complete tree for all the available example alerts given in Fig. 38.

Algorithm 1: Recursive Tree Builder

Data: Set of Alerts $\mathcal{A}$

$\mathcal{A} \leftarrow$ Sort($\mathcal{A}$) based on timestamp in Descending order;
$data \leftarrow \mathcal{A}$;
set Root Node $\leftarrow$ Node;
initialise cache;
Function `TreeSearch(`*Root_Node,IP, data, parent_node, index, cache*`):`
    tmp_data $\leftarrow$ data.get_first(IP, index);
    if *(tmp_data is Empty) or (tmp_data.ID in cache)* **then**
        return;
    index $\leftarrow$ tmp_data.index;
    new_node $\leftarrow$ Node(tmp_data, parent);
    if *tmp_data.communicates_with* **then**
        $new\_found\_ip \leftarrow$ tmp_data.communicates_with;
        TreeSearch(root_Node, new_found_ip, data, new_node, $index+1$, cache);
    return TreeSearch(root_Node, IP, data, new_node, $index+1$, cache);

ts	Detector	IP	Communicates with
13:36:00	ECOD	192.1.1.1	None
13:35:00	ECOD	192.1.1.1	None
13:34:00	SARIMA	192.1.1.1	192.1.1.2
13:33:46	ECOD	192.1.1.2	192.1.1.3
13:33:33	ECOD	192.1.1.3	None
13:33:20	SARIMA	192.1.1.1	192.1.1.12
13:31:18	EOCD	192.1.1.2	None
13:29:14	SARIMA	192.1.1.2	None

(a) Selected alerts in iteration 2.

ts	Detector	IP	Communicates with
13:36:00	ECOD	192.1.1.1	None
13:35:00	ECOD	192.1.1.1	None
13:34:00	SARIMA	192.1.1.1	192.1.1.2
13:33:46	ECOD	192.1.1.2	192.1.1.3
13:33:33	ECOD	192.1.1.3	None
13:33:20	SARIMA	192.1.1.1	192.1.1.12
13:31:18	EOCD	192.1.1.2	None
13:29:14	SARIMA	192.1.1.2	None

(b) Selected alerts in iteration 6.

Table 9: Construction alert tree, selected alerts.

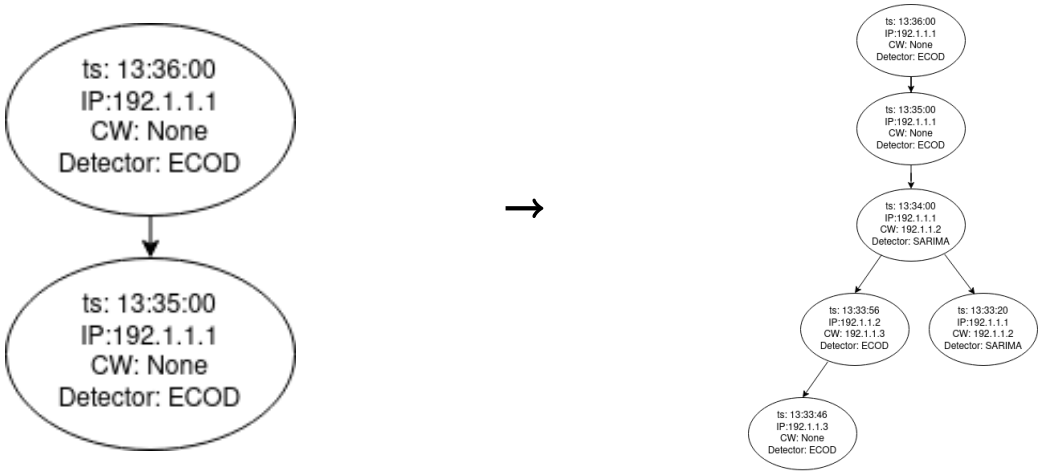


Figure 38: Constructing alert tree.

10.4 Attack Strategy Extraction

We search among the alert trees to extract the most probable attack strategy, calculate the height for $b \in B$, and sort according to the height. We assume that the the longest path represents the most indicative attack strategy of a multi-stage attacker. After obtaining the tree with the largest depth, we determine the tree's lowest node(s) and walk the tree bottom-up or ascending in time. The nodes found along the way will be added to the list. In the end, we return a list of alert nodes ascending in time representing the suspected attack strategy. Fig. 39 depicts this process given that the path annotated in red is the longest path among B or for all the systems components with a constructed tree.

Algorithm 2: Attack Path Extraction

Data: Set of Alert Trees $\mathcal{B}$

15 $\mathcal{B} \leftarrow$ Sort($\mathcal{B}$) based on tree height;

16 $b \leftarrow$ getFirst($\mathcal{B}$);

17 end_node $\leftarrow$ getEndNodeInTree(b);

18 $\mathcal{N} \leftarrow$ walkTree(root_node, end_node) ; `// Walks the Binary tree from the end Node to the root Node and returns a list of Nodes found along the way.`

19 $\mathcal{N} \leftarrow$ Sort($\mathcal{N}$) based on timestamp in Ascending order;

20 **return** $\mathcal{N}$; `// gives the extracted attack path for the raised alerts.`

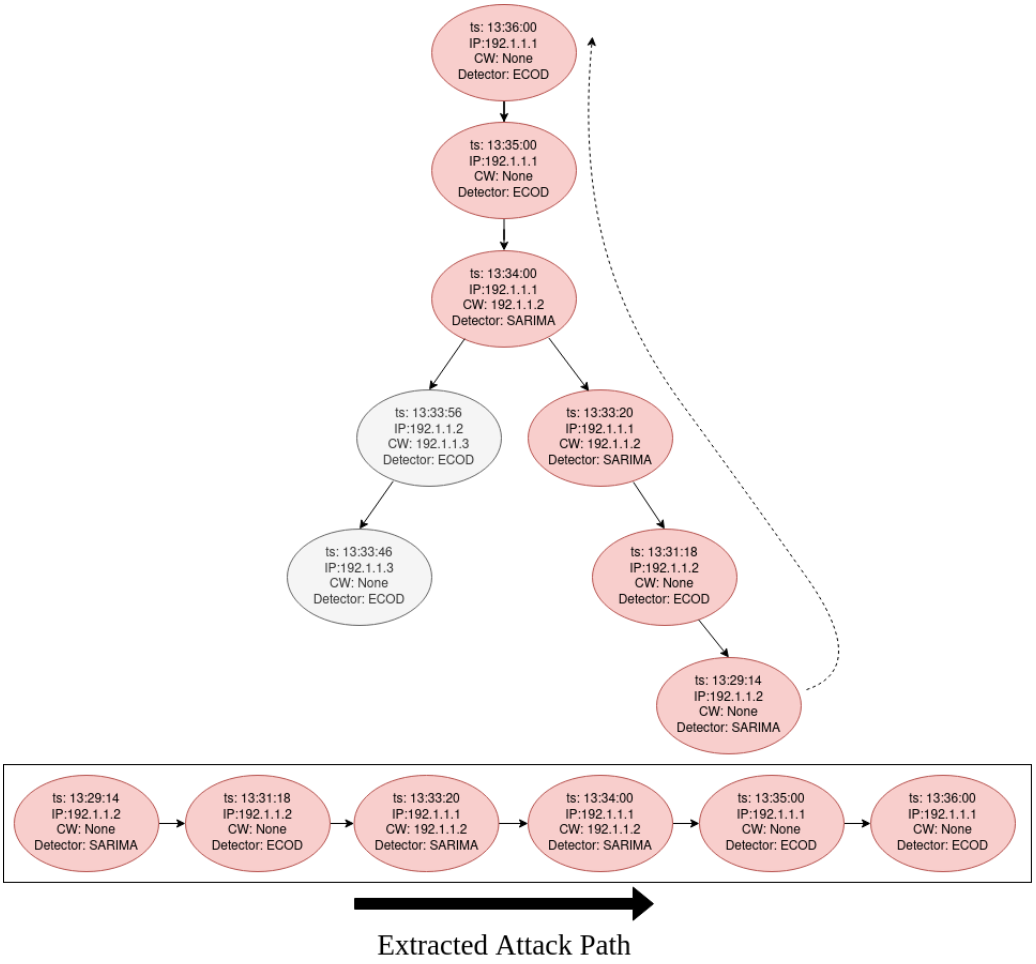


Figure 39: Extracting attack strategy example for *IP: 192.1.1.1*.

11 Evaluation of Merging Concept

To prove our merging concept, we ran an experiment ingesting the alerts raised by the previous attack detectors in Section 9 and the physical mock-up detector. Table 10 gives the count for all the previously obtained alerts and the alerts generated by the mock-up process aware detector where in total $|\mathcal{A}| = 115$ alerts raised in phase 1 of which 80 are network-based, and 35 are physical based alerts. Table 11 gives the count for the number of alerts according to the components of the monitored system and gives the longest path discovered for each system component in phase 3. Here, in phase 4, we find that the longest pathway has been established for the PLC1 - Raw Water Intake component, with a total of 78 correlated events. Table 12 shows the correlated path longest path starting with the first alert at *10:30:30.00* and the last attack alert on *13:25:00.00*. In appendix C.5 the binary tree can be found and the longest path in relation to the other alerts. An online version for the same figure can be found on GitHub [86]. Let's compare the event timeline Table 4 with the merger results in Table 12 from phase 4. We find that the extracted attack strategy accurately represents the actual attack steps. The overall number of alerts in the Swat A6 2019 dataset is reduced from 115 to 78, with a reduction of 32.2% without compromising on the extracted attack strategy.

Furthermore, we find that of the 78 alerts, a total of 4 alerts remain false positives, indicated by the numbers $\{40, 64, 68, 69\}$, 3 of the false positives are raised by the ECOD-based detector, and one alert is raised by the SARIMA exfiltration-based detector. This shows that this method reduces the total false positives by 81% from 21 to 4 on the *SWaT A6 2019* dataset. Looking at Table 12, a distinct pattern is present showing an attack strategy. First, from index 0 up to number 39, we can observe a distinct behaviour, reflecting the exfiltration of data. The pattern is followed by an alert from the ECOD-based detector demonstrating a false positive. Thereafter, however, we see a clear pattern emerging where we can observe a malicious command being sent to the SCADA workstation followed by a number of physical sensor alerts. This pattern repeats five times and corresponds to the event timeline in Table 4.

Here, the results show that our current merging method is capable of both reducing the number of false positives and identifying the multiple stages of the attack on the *SWaT A6 2019* dataset. Nevertheless, the results still leave room for improvement. In our case, there are still true positive alerts that do not appear in the longest path. Further development of the method would be beneficial as the proof-of-concept demonstrates promising results. Improvements can be to consider additional correlation features, something that was not done for the scope of this study. Furthermore, the temporal distance between alerts is now not considered in the correlation process. For example, alerts with a distance of a month are still correlated in this method. However, we anticipate the temporal distance may be important to the correlation process. As part of the future work, therefore, we may implement constraints such as $a.ts_1 - a.ts_2 > threshold$. Next, we only validated this approach against one dataset. And finally, in the *SWaT A6 2019* there is a clear pattern of consecutive attacks. It is unknown how this approach would hold against multiple parallel unstructured ICS attacks. We suspect that when multiple attackers target different components more alert trees would be generated. This results in the remaining question on how to prioritise between the various constructed alert trees.

Detector	***Number of Alerts Generated***
SARIMA Exfiltration detector	47
ECOD outlier detector	18
SARIMA Disruption detector	15
Mock-up Physical Process aware detector	35
Total	115

Table 10: Number of alerts raised according to Detector type for *SWaT A6 2019* dataset.

Name	***IP***	***Number of Alerts Generated***	***Longest Path***
SCADA - Engineering Workstation	192.168.1.201	54	70
PLC1 - RAW Water Intake	192.168.1.10	47	78
SCADA - Historian Server	162.168.1.200	10	70
SCADA - Touch Panel	192.168.1.100	1	54
PLC6 - RO Product	192.168.1.60	1	65
PLC3 - Ultra-Filtration	192.168.1.30	1	65
PLC2 - Pre Treatment	192.168.1.20	1	1
	Total	115	

Table 11: Alerts raised and longest path according to the component for *SWaT A6 2019* dataset.

number	*timestamp*	*component*	*ip*	*communicates_with*	*detector*	*type*
0	2019-12-06 10:30:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
1	2019-12-06 10:31:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
2	2019-12-06 10:31:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
3	2019-12-06 10:32:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
4	2019-12-06 10:32:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
5	2019-12-06 10:33:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
6	2019-12-06 10:33:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
7	2019-12-06 10:34:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
8	2019-12-06 10:34:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
9	2019-12-06 10:35:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
10	2019-12-06 10:45:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
11	2019-12-06 10:46:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
12	2019-12-06 10:46:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
13	2019-12-06 10:47:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
14	2019-12-06 10:47:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
15	2019-12-06 10:48:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
16	2019-12-06 10:48:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
17	2019-12-06 10:49:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
18	2019-12-06 10:49:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
19	2019-12-06 10:50:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
20	2019-12-06 11:00:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
21	2019-12-06 11:01:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
22	2019-12-06 11:01:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
23	2019-12-06 11:02:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
24	2019-12-06 11:02:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
25	2019-12-06 11:03:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
26	2019-12-06 11:03:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
27	2019-12-06 11:04:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
28	2019-12-06 11:04:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
29	2019-12-06 11:05:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
30	2019-12-06 11:15:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
31	2019-12-06 11:16:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
32	2019-12-06 11:16:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
33	2019-12-06 11:17:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
34	2019-12-06 11:17:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
35	2019-12-06 11:18:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
36	2019-12-06 11:18:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
37	2019-12-06 11:19:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
38	2019-12-06 11:19:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
39	2019-12-06 11:20:00.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA exfiltration	Network Alert
40	2019-12-06 12:30:00.529025	SCADA - Engineering Workstation	192.168.1.201	192.168.1.200	ECOD Second Malware	Network Alert
41	2019-12-06 12:30:30.000000	PLC1- RAW Water	192.168.1.10	192.168.1.201	SARIMA disruption	Network Alert
42	2019-12-06 12:30:30.000000	FIT101.Pv			Physical Detector	Physical Alert
43	2019-12-06 12:31:00.000000	FIT101.Pv			Physical Detector	Physical Alert
44	2019-12-06 12:31:30.000000	FIT101.Pv			Physical Detector	Physical Alert
45	2019-12-06 12:32:00.000000	FIT101.Pv			Physical Detector	Physical Alert
46	2019-12-06 12:32:30.000000	FIT101.Pv			Physical Detector	Physical Alert
47	2019-12-06 12:33:00.000000	FIT101.Pv			Physical Detector	Physical Alert
48	2019-12-06 12:43:00.000000	FIT101.Pv			Physical Detector	Physical Alert
49	2019-12-06 12:43:30.000000	SCADA - Engineering Workstation	192.168.1.201	192.168.1.10	SARIMA disruption	Network Alert
50	2019-12-06 12:43:30.000000	PLC1- RAW Water	192.168.1.10	192.168.1.201	SARIMA disruption	Network Alert
51	2019-12-06 12:44:00.000000	FIT101.Pv			Physical Detector	Physical Alert
52	2019-12-06 12:44:30.000000	FIT101.Pv			Physical Detector	Physical Alert
53	2019-12-06 12:45:00.000000	FIT101.Pv			Physical Detector	Physical Alert
54	2019-12-06 12:45:30.000000	FIT101.Pv			Physical Detector	Physical Alert
55	2019-12-06 12:46:00.000000	FIT101.Pv			Physical Detector	Physical Alert
56	2019-12-06 12:56:00.000000	FIT101.Pv			Physical Detector	Physical Alert
57	2019-12-06 12:56:30.000000	SCADA - Engineering Workstation	192.168.1.201	192.168.1.10	SARIMA disruption	Network Alert
58	2019-12-06 12:56:30.000000	PLC1- RAW Water	192.168.1.10	192.168.1.201	SARIMA disruption	Network Alert
59	2019-12-06 12:57:00.000000	FIT101.Pv			Physical Detector	Physical Alert
60	2019-12-06 12:57:30.000000	FIT101.Pv			Physical Detector	Physical Alert
61	2019-12-06 12:58:00.000000	FIT101.Pv			Physical Detector	Physical Alert
62	2019-12-06 12:58:30.000000	FIT101.Pv			Physical Detector	Physical Alert
63	2019-12-06 12:59:00.000000	FIT101.Pv			Physical Detector	Physical Alert
64	2019-12-06 13:00:00.329797	PLC1- RAW Water	192.168.1.10	192.168.1.30	ECOD Second Malware	Network Alert
65	2019-12-06 13:09:00.000000	FIT101.Pv			Physical Detector	Physical Alert
66	2019-12-06 13:09:30.000000	SCADA - Engineering Workstation	192.168.1.201	192.168.1.10	SARIMA disruption	Network Alert
67	2019-12-06 13:12:30.000000	SCADA - Engineering Workstation	192.168.1.201		SARIMA disruption	Network Alert
68	2019-12-06 13:15:24.885210	SCADA - Historian Server	192.168.1.200	192.168.1.201	ECOD Second Malware	Network Alert
69	2019-12-06 13:16:30.000000	SCADA - Engineering Workstation	192.168.1.201	192.168.1.200	SARIMA exfiltration	Network Alert
70	2019-12-06 13:22:30.000000	PLC1- RAW Water	192.168.1.10	192.168.1.201	SARIMA disruption	Network Alert
71	2019-12-06 13:22:30.000000	FIT101.Pv			Physical Detector	Physical Alert
72	2019-12-06 13:22:30.000000	PLC1- RAW Water	192.168.1.10		SARIMA exfiltration	Network Alert
73	2019-12-06 13:23:00.000000	FIT101.Pv			Physical Detector	Physical Alert
74	2019-12-06 13:23:30.000000	FIT101.Pv			Physical Detector	Physical Alert
75	2019-12-06 13:24:00.000000	FIT101.Pv			Physical Detector	Physical Alert
76	2019-12-06 13:24:30.000000	FIT101.Pv			Physical Detector	Physical Alert
77	2019-12-06 13:25:00.000000	FIT101.Pv			Physical Detector	Physical Alert

Table 12: Attack reconstruction longest path results: *PLC1 - Raw Water Intake (IP:192.168.1.10)*.

12 Conclusion

The objective of this thesis was to better detect multi-stage attacks by integrating physical-sensor data and network data in OT environments. Section 12.1 presents the main findings and contributions of this thesis, answering the research questions pointed in Section 1.1. Next, Section 12.2 addresses the limitations encountered during this research. Finally, in Section 12.3, we would like to highlight several directions that may be explored in future work.

12.1 Main Findings and Contributions

i *What is the current state-of-art on cyber-attack detection in ICS systems?*

In Section 5 we have presented an overview of the current literature to identify the gaps in the state-of-the-art. We observed that in the reviewed literature, the majority of papers focus on anomaly detection using physical sensor data in the OT domain [50, 52, 53, 58, 59, 60, 61, 62, 63, 64, 65]. The paper by Jadidi et al. [57] is the first among the consulted papers that introduced a method that integrates both physical sensor data and network data in the OT domain to detect ICS cyber-attacks. Their multi-layered system allows for detecting flooding attacks and was tested against three different publicly available ICS datasets, of which only the *S7comm factory* dataset [76] contains both physical sensors and network data during flooding attacks and process attacks. Next, we found the *SWaT 2015* dataset is frequently used in the surveyed studies. A number of these pieces [69, 60, 70, 56] introduce detection methods utilising advanced machine learning, typically introducing high computational requirements with the primary aim to achieve the highest possible detection performance. This suggests that methods for detecting attacks that exploit multi-domain input data in OT environments remain largely unexplored. Similarly, we observed that the detection of multi-stage attacks in ICS is also a relatively unexplored area.

ii *What public datasets are available for the development of a network and physical sensor-based attack detection method for ICS architectures?*

With the previous aim in mind, we provided an overview of datasets that support the research efforts on multi-domain attack detection and find that two datasets of the reviewed datasets enable this type of research. Firstly, the aforementioned *S7comm factory* dataset containing both physical sensor data and network data under a series of flooding attacks and process attacks. Secondly, the *SWaT A6 2019* dataset comprising of measurements during an experiment on a physical testbed with a multi-stage attacker traversing from higher levels to the lowest level of the Purdue model, resulting in deviating patterns in the network communication and the sensor measurements, properties that are of interest for the scope of this research. Next, we found that the *SWaT 2015* dataset is used extensively throughout the literature while the *SWaT A6 2019* dataset has, to the best of our knowledge, not been mentioned in the consulted papers making this paper first use this dataset. We continued examining alternative approaches for data acquisition based on the fact that we have only found two datasets that contain both physical sensor data and network data, of which only one contained a multi-stage attack strategy across various Purdue layers. One novel means of data acquisition is through the DHALSIM digital twin allowing the simulation of various ICS water distribution architectures and attacks. Using this novel digital twin, we generated our own ICS datasets (publicly available) and assessed the usability of the dataset. We found that the DHALSIM digital twin presents an effective approach for data acquisition but is still under development. After the assessment of the various datasets, we decided to continue with the *SWaT A6 2019* dataset since we observed during our dataset analysis that this dataset, as opposed to the *S7comm factory* dataset or the self-simulated DHALSIM dataset, provides more granular detail such as benign Windows update traffic and other benign background communication. In addition, the six-process SWAT testbed architecture has greater complexity than the *S7comm factory* testbed and represents a larger-scale critical infrastructure under the attack of a multi-stage attacker. Due to limited documentation often encountered during the assessment of the ICS datasets, including the *SWaT A6 2019 dataset*, a deeper dataset analysis for the *SWaT A6 2019* was performed to create an overview of how the attacker gains a foothold onto the lowest

level, affecting one of the physical processes. We also provide a more in-depth interpretation of the dataset that is currently missing in the documentation and explain the dataset according to the Spiral model by Hassanzadeh, and Burkeet al. [27]. Below gives a non-exhaustive summary of the usable datasets for this type of research:

DHALSIM Digital Twin: Digital Twin of water distribution systems allow for simulation of various ICS water distribution topologies and include three types of attack strategies to simulate with. The digital twin is still under development at the time of writing this thesis, so new topologies and attack strategies might be added.

SWaT A6 2019 Datasets: Dataset based on measurement of a physical testbed in Singapore representing six water filtration processes introducing a realistic scenario of a real-world deployment. The dataset contains raw packet captures and physical sensor data information for roughly three hours of operation under a total of (multi-stage) five attacks steps, including a malware infection through a USB device, a data exfiltration step, a second malware download, and an attack targeting one of the PLCs in the architecture.

S7comm Factory Dataset: Dataset contains the process data and network data (raw packet captures) of a small-scale factory testbed with 32 processes. A total of 21 attacks are available, most of which target physical processes.

iii *How to achieve full visibility in an ICS architecture?*

(a) *At what position in the Purdue model should each (sub)-detector be placed to achieve full visibility?*

First, we looked at the Purdue Model, which is a useful representation of a multi-layer ICS architecture. Here we found that to be able to detect a multi-stage attacker trying to traverse from the higher layers of the Purdue model to the lower layers and observe all the anomalous behaviour, several strategic positions within the model should be used for detection. In the case of the *SWaT A6 2019* dataset used in this study, we find that the testbed on which this dataset is based offers an accurate representation of the Purdue model. The attacker in this scenario can initiate the attack from outside the testbed or IT network by gaining a foothold onto level 3 following a series of attack steps traversing the OT part of the Purdue model to reach the lowest layers and cause physical impact. As such, we apply the general findings and place four detectors at different strategic positions in the Purdue model to detect a multi-step attacker. The positions given in Fig. 40 support the idea that a single detector will not be sufficient to achieve the ability to observe all the anomalous behaviour in an ICS architecture and obtain full visibility against a multi-stage ICS attack. The multi-stage attack in the *SWaT A6 2019* data targets the SCADA engineering workstation in level 3, and from there, the attacker reaches the lowest levels resulting in deviations of the physical process. The dataset does not contain any further data on attacks targeting level 2, so this is out of the scope for this research. Finally, to avoid over-fitting on the *SWaT A6 2019* dataset, the detector is placed on the location depicted in Fig. 40. We instead tested the detectors on all the components mentioned in the technical description, ensuring a more generic deployment.

(b) *What (sub)-detectors are required to obtain full visibility?*

Following the idea of placing various sub-detector at different positions in the Purdue Model discussed in the previous sub-question, we apply a total of four different ICS attack sub-detectors placed at levels 0 to 3 in the Purdue model depicted in Fig. 40 and test them against the *SWaT A6 2019* dataset. The first two detectors placed at level 3 aim at detecting large malicious file transfers to identify data exfiltration and a malware download after the initial comprise of the SCADA engineering workstation. The next detector we propose allows for monitoring the ICS control messages found in level 1 of the Purdue Model. In the case of the *SWaT A6 2019*, the Ethernet/IP communication is monitored for malicious commands targeting the PLCs responsible for the underlying physical process. The last detector is placed at level 0 in order to utilise the physical sensor readings for attack identification and impact understanding. As we have observed that there

are already some studies in the literature that focus on physical sensor-based detection, we decided for the scope of this research to introduce a mock-up detector for this part.

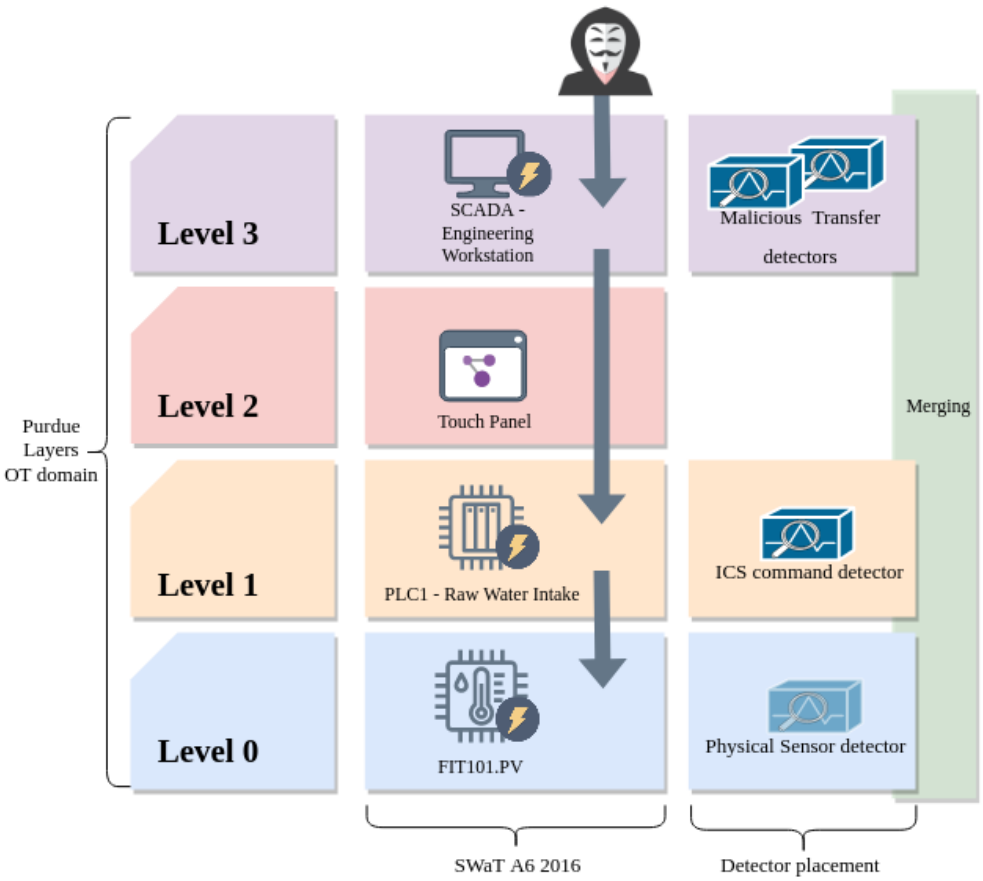


Figure 40: Sub-detector placement in *SWaT A6 2019* dataset.

(c) *What features are required for each (sub)-detector?*

The network data acquired in the form of raw packet captures is translated into Zeek logs. Zeek provides the ability to specify what network logs are generated, provides customisability, can be extended with additional ICS packet parsing scripts and introduces event-based information while maintaining low storage requirements. Hence, we apply Zeek features in the form of connection logs for all the proposed network-based attack detectors. Looking at Fig. 40 we introduced three network-based attack detectors. For the first detectors applied in the exfiltration detector in layer 3, we apply a mechanism forecasting the logging behaviour of Zeek connection logs for a given ICS component to monitor for data exfiltrations. Using lightweight seasonal Seasonal Arima for prediction of the connection logs, stateless differencing, combined with a pre-determined threshold, alerts are raised if the predicted behaviour does not fit the trained model. The same procedure is performed for the ICS command detector, where we only monitor the connection logs for the ICS commands. In the case of the *SWaT A6 2019* dataset, we monitored for ENIP command packets for each specified component present in the testbed. In other ICS architectures, this function can easily be replaced by other ICS protocols, such as Modbus. Next, the second level 3 detector is used for the identification of abnormal connections. Here we adopted ECOD, an efficient unsupervised outlier detector employing four Zeek connection log features describing the number of packets in a connection in both ways and the total number of bytes exchanged in a connection in both ways. The results showed that the detectors were successful in identifying all the attack events in the *SWaT A6 2019* dataset on a post-mortem basis. For the network-based detector applying SARIMA at level 3, one false positive was raised, while the

network detector applying SARIMA to monitor the ICS command behaviour achieved an accuracy of 1.0. The ECOD-based detector results indicate scalable operation in ICS architectures where we processed a maximum of 932367 individual connections with only 12 false positives. Furthermore, results showed that the unsupervised ECOD model is efficient against a large set of connections, with training time in all of the cases staying below a second. On the other hand, during the experiment, the number of normal data samples was relatively small, and the detector was only tested against one type of attack event, requiring a better investigation of this method.

iv *How to achieve full attack visibility characterised by all the observed anomalous behaviour?*

In the study, we have introduced and tested several individual detectors. Although the individual detectors showed good detection performance, false positives were still introduced. In addition, individual detectors do not provide much intelligence about a multi-step attacker. As a solution to go from full visibility to full-attack visibility and reduce the number of false positives, we have devised a method to correlate the individual alerts by means of a binary tree alert tree using both a similarity as well as a sequential-based method. We have found that the longest correlated alert path in that tree accurately reflects the multi-stage attack path we are looking for and that this method can greatly reduce the number of false positives. The results against the *SWaT A6 2019* dataset and our detectors showed that false positives can be lowered by 81%. At the same time, the proof-of-concept has been tested against one dataset leaving room open for more testing and additional improvements.

12.2 Limitations

Although this research has demonstrated that many of the proposed methodologies are effective, there are still a number of limitations we wish to highlight in this section.

1. **Dependence on forecasting model:** For one of the detectors, we are using the SARIMA model to learn the normal communication behaviour. We have observed that the communication behaviour during our study exhibits a certain periodicity and seasonality, which is well fit for the SARIMA model. However, assuming that all real-world systems exhibit this periodicity or seasonality is unreasonable. As such, SARIMA might be unfit for learning more complex communication behaviours that do not contain seasonality. Alternative forecasting algorithms that can be considered are Long Short-Term Memory or other novel approaches such as NeuralProphet [103].

2. **Generalisation towards other ICS architectures:** Another limitation of our research is the fact that our approach is relatively specific to the SWaT A6 2019 dataset. The features used are derived based on our data analysis and are adapted to the used dataset. Therefore, it remains a valid question as to how we can tailor this approach to other ICS architectures with different data flows. This is something we have not covered in detail in our research. A first step towards applying this approach to other systems would be to examine critically how the ICS network is structured and what data flows is collected within the current system.

3. **Mock-up physical detector:** Finally, the last limitation we wish to highlight is the use of a physical sensor mock-up detector at level 0 of the Purdue model. Here we have assumed a perfect detector capable of detecting all attacks without any false positives. This is an assumption that affects our merging results. A physical sensor attack detector is not expected to obtain perfect detection performance in a real-world setting. As such, it could be that if an attack event is missed or not detected that the alert is not added to the alert tree. This could affect the selection of the longest path or the used strategy.

12.3 Future Work

In this study, we have made a first step to integrate physical and network data for ICS, carrying much potential for real-world deployment. During the study, we evaluated the approach detection and alert merging methods against only one dataset. As such, a more comprehensive evaluation would be beneficial in order to make choices on how to improve the introduced method. In the same way, we may further test the introduced methods on the *S7comm factory* dataset, the DHALSIM custom dataset or possibly other datasets. Similarly, we intend to evaluate the methods against other attacks in order to get a better sense of the detection capabilities and performance of the introduced methods. Next, it is worthwhile to investigate if the ECOD-based detectors can achieve a reduced false positive rate if we train the models with more normal operation data. Once this evaluation has been conducted, this would indicate several possible improvements in the method of merging alerts.

References

[1] Kaveh Paridari et al. “A Framework for Attack-Resilient Industrial Control Systems: Attack Detection and Controller Reconfiguration”. In: *Proceedings of the IEEE* 106.1 (2018), pp. 113–128. DOI: 10.1109/JPROC.2017.2725482.

[2] Yan Hu et al. “A survey of intrusion detection on industrial control systems”. In: *International Journal of Distributed Sensor Networks* 14 (Aug. 2018), p. 155014771879461. DOI: 10.1177/1550147718794615.

[3] Eric Luiijf and Marieke Klaver. “Resilience Approach to Critical Information Infrastructures: Theories, Methods, Tools and Technologies”. In: Jan. 2019, pp. 3–16. ISBN: 978-3-030-00023-3. DOI: 10.1007/978-3-030-00024-0_1.

[4] Mauro Conti, Denis Donadel, and Federico Turrin. “A Survey on Industrial Control System Testbeds and Datasets for Security Research”. In: *CoRR* abs/2102.05631 (2021). arXiv: 2102.05631. URL: https://arxiv.org/abs/2102.05631.

[5] Alexander Staves et al. “A Cyber Incident Response and Recovery Framework to Support Operators of Industrial Control Systems”. In: *International Journal of Critical Infrastructure Protection* 37 (2022), p. 100505. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2021.100505. URL: https://www.sciencedirect.com/science/article/pii/S187454822100086X.

[6] Keith Stouffer et al. *Guide to Industrial Control Systems (ICS) Security*. Tech. rep. 2015. DOI: 10.6028/NIST.SP.800-82r2. URL: https://doi.org/10.6028/NIST.SP.800-82r2.

[7] Mehrotra Turton. “Hackers Breached Colonial Pipeline Using Compromised Password”. In: *Bloomberg* (). URL: https://www.bloomberg.com/news/articles/2021-06-04/hackers-breached-colonial-pipeline-using-compromised-password.

[8] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano. “TRITON: The first ICS cyber attack on safety instrument systems”. In: *Proc. Black Hat USA*. Vol. 2018. 2018, pp. 1–26.

[9] DarkSide Ransomware: Best Practices for Preventing Business Disruption from Ransomware Attacks. “Alert (AA21-131A)”. In: *CIS* (). URL: https://www.cisa.gov/uscert/ics/alerts/IR-ALERT-H-16-056-01.

[10] David E. Whitehead et al. “Ukraine Cyber-Induced Power Outage”. In: 2017.

[11] Thomas Miller et al. “Looking back to look forward: Lessons learnt from cyber-attacks on Industrial Control Systems”. In: *International Journal of Critical Infrastructure Protection* 35 (2021), p. 100464. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2021.100464. URL: https://www.sciencedirect.com/science/article/pii/S1874548221000524.

[12] Glenn Murray, Michael N. Johnstone, and Craig Valli. “The convergence of IT and OT in critical infrastructure”. In: 2017.

[13] Oualid Koucham et al. “Cross-domain alert correlation methodology for industrial control systems”. In: *Computers Security* 118 (2022), p. 102723. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2022.102723. URL: https://www.sciencedirect.com/science/article/pii/S0167404822001183.

[14] Mingtao Wu and Young Moon. “Alert Correlation for Cyber-Manufacturing Intrusion Detection”. In: *Procedia Manufacturing* 34 (2019). 47th SME North American Manufacturing Research Conference, NAMRC 47, Pennsylvania, USA., pp. 820–831. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2019.06.197. URL: https://www.sciencedirect.com/science/article/pii/S2351978919309278.

[15] Luciana Obregon. “SANS Institute Information Security Reading Room Secure Architecture for Industrial Control Systems”. In: 2019.

[16] Allan Cook et al. “Attribution of Cyber Attacks on Industrial Control Systems”. In: *EAI Endorsed Trans. Ind. Networks Intell. Syst.* 3 (2016), e3.

[17] *Operational Technology Security - NIST*. https://csrc.nist.gov/projects/operational-technology-security. [Online Accessed: 22-01-2022].

[18] Otis D. Alexander, Mi-Hye Belisle, and Jacob Steele. “MITRE ATT&CK® for Industrial Control Systems: Design and Philosophy”. In: 2020.

[19] Zakarya Drias, Olivier Vogel, and Ahmed Serhrouchni. “Analysis of cyber security for industrial control systems”. In: Aug. 2015. DOI: 10.1109/SSIC.2015.7245330.

[20] Mohamad Kaouk et al. “A Review of Intrusion Detection Systems for Industrial Control Systems”. In: *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. 2019, pp. 1699–1704. DOI: 10.1109/CoDIT.2019.8820602.

[21] Nist and Emmanuel Aroms. *NIST Special Publication 800-39 Managing Information Security Risk*. Scotts Valley, CA: CreateSpace, 2012. ISBN: 1470110598.

[22] Mahdi Daghmehchi Firoozjaei et al. “An evaluation framework for industrial control system cyber incidents”. In: *International Journal of Critical Infrastructure Protection* 36 (2022), p. 100487. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2021.100487. URL: https://www.sciencedirect.com/science/article/pii/S1874548221000718.

[23] *TRITON Malware Remains Threat to Global Critical Infrastructure Industrial Control Systems (ICS)*. https://www.ic3.gov/Media/News/2022/220325.pdf. [Online Accessed: 27-06-2022].

[24] Brendan Galloway and Gerhard P. Hancke. “Introduction to Industrial Control Networks”. In: *IEEE Communications Surveys Tutorials* 15.2 (2013), pp. 860–880. DOI: 10.1109/SURV.2012.071812.00124.

[25] Dan Sullivan, Eric Luiijf, and Edward Colbert. “Components of Industrial Control Systems”. In: 66 (Aug. 2016), pp. 15–28. DOI: 10.1007/978-3-319-32125-7_2.

[26] Bradley Reaves and Thomas Morris. “Analysis and mitigation of vulnerabilities in short-range wireless communications for industrial control systems”. In: *International Journal of Critical Infrastructure Protection* 5.3 (2012), pp. 154–174. ISSN: 1874-5482. DOI: https://doi.org/10.1016/j.ijcip.2012.10.001. URL: https://www.sciencedirect.com/science/article/pii/S1874548212000492.

[27] Amin Hassanzadeh and Robin Burkett. “SAMIIT: Spiral Attack Model in IIoT Mapping Security Alerts to Attack Life Cycle Phases”. In: 2018.

[28] Cyber-Attack Against Ukrainian Critical Infrastructure. “ICS Alert”. In: *CIS* (). URL: https://www.cisa.gov/uscert/ics/alerts/IR-ALERT-H-16-056-01.

[29] Tejasvi Alladi, Vinay Chamola, and Sherali Zeadally. “Industrial Control Systems: Cyberattack trends and countermeasures”. In: *Computer Communications* 155 (2020), pp. 1–8. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2020.03.007. URL: https://www.sciencedirect.com/science/article/pii/S0140366419319991.

[30] Georgios Michail Makrakis et al. *Vulnerabilities and Attacks Against Industrial Control Systems and Critical Infrastructures*. 2021. DOI: 10.48550/ARXIV.2109.03945. URL: https://arxiv.org/abs/2109.03945.

[31] Emiliano Sisinni et al. “Industrial Internet of Things: Challenges, Opportunities, and Directions”. In: *IEEE Transactions on Industrial Informatics* 14.11 (2018), pp. 4724–4734. DOI: 10.1109/TII.2018.2852491.

[32] K. A. Scarfone and P. M. Mell. *Guide to Intrusion Detection and Prevention Systems (IDPS)*. Tech. rep. Gaithersburg, MD, 2007. DOI: 10.6028/NIST.SP.800-94. URL: https://doi.org/10.6028/NIST.SP.800-94.

[33] Robert Mitchell and Ing-Ray Chen. “A Survey of Intrusion Detection Techniques for Cyber-Physical Systems”. In: *ACM Comput. Surv.* 46.4 (Mar. 2014). ISSN: 0360-0300. DOI: 10.1145/2542049. URL: https://doi.org/10.1145/2542049.

[34] Jairo Giraldo et al. “A Survey of Physics-Based Attack Detection in Cyber-Physical Systems”. In: *ACM Comput. Surv.* 51.4 (July 2018). ISSN: 0360-0300. DOI: 10.1145/3203245. URL: https://doi.org/10.1145/3203245.

[35] Hervé Debar, Marc Dacier, and Andreas Wespi. “Towards a taxonomy of intrusion-detection systems”. In: *Computer Networks* 31.8 (1999), pp. 805–822. ISSN: 1389-1286. DOI: https://doi.org/10.1016/S1389-1286(98)00017-6. URL: https://www.sciencedirect.com/science/article/pii/S1389128698000176.

[36] Edward Colbert and Steve Hutchinson. “Intrusion Detection in Industrial Control Systems”. In: 66 (Aug. 2016), pp. 209–237. DOI: 10.1007/978-3-319-32125-7_11.

[37] A.A. Cardenas, J.S. Baras, and K. Seamon. “A framework for the evaluation of intrusion detection systems”. In: *2006 IEEE Symposium on Security and Privacy (S P’06)*. 2006, 15 pp.–77. DOI: 10.1109/SP.2006.2.

[38] Andres Murillo et al. “Co-Simulating Physical Processes and Network Data for High-Fidelity Cyber-Security Experiments”. In: *Sixth Annual Industrial Control System Security (ICSS) Workshop*. ICSS 2020. Austin, TX, USA: Association for Computing Machinery, 2020, pp. 13–20. ISBN: 9781450390026. DOI: 10.1145/3442144.3442147. URL: https://doi.org/10.1145/3442144.3442147.

[39] Daniele Antonioli and Nils Ole Tippenhauer. “MiniCPS: A Toolkit for Security Research on CPS Networks”. In: *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. CPS-SPC ’15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 91–100. ISBN: 9781450338271. DOI: 10.1145/2808705.2808715. URL: https://doi.org/10.1145/2808705.2808715.

[40] Murray and Haxton. “An Overview of the Water Network Tool for Resilience (WNTR)”. In: 2018. URL: https://wntr.readthedocs.io/en/latest/overview.html.

[41] *Mininet - An Instant Virtual Network on your Laptop*. http://mininet.org/. [Online Accessed: 24-01-2022].

[42] *EPANET - Application for Modeling Drinking Water Distribution Systems*. https://www.epa.gov/water-research/epanet. [Online Accessed: 24-01-2022].

[43] *DHALSIM - github*. ://github.com/afmurillo/DHALSIM.

[44] Rick Hofstede et al. “Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX”. In: *IEEE Communications Surveys Tutorials* 16.4 (2014), pp. 2037–2064. DOI: 10.1109/COMST.2014.2321898.

[45] *NetFlow v9 vs. NetFlow v5: What are the differences?* https://www.plixer.com/blog/netflow-v9-vs-netflow-v5/. [Online Accessed: 28-04-2022].

[46] *Network Flow Monitoring Explained: NetFlow vs sFlow vs IPFIX*. hhttps://www.varonis.com/blog/flow-monitoring. [Online Accessed: 28-04-2022].

[47] *Zeek - open source Network Security Monitoring Tool*. https://zeek.org/. [Online Accessed: 12-05-2022].

[48] Austin Jones, Zhaodan Kong, and Calin Belta. “Anomaly detection in cyber-physical systems: A formal methods approach”. In: *53rd IEEE Conference on Decision and Control*. Dec. 2014, pp. 848–853. DOI: 10.1109/CDC.2014.7039487.

[49] Asuka Terai et al. “Cyber-Attack Detection for Industrial Control System Monitoring with Support Vector Machine Based on Communication Profile”. In: *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*. Apr. 2017, pp. 132–138. DOI: 10.1109/EuroSPW.2017.62.

[50] Jonathan Goh et al. “Anomaly Detection in Cyber Physical Systems Using Recurrent Neural Networks”. In: *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. 2017, pp. 140–145. DOI: 10.1109/HASE.2017.36.

[51] *Secure Water Treatment - iTrust*. https://itrust.sutd.edu.sg/testbeds/secure-water-treatment-swat/. [Online Accessed: 8-02-2022].

[52] Dmitry Shalyga, Pavel Filonov, and Andrey Lavrentyev. “Anomaly Detection for Water Treatment System based on Neural Network with Automatic Architecture Optimization”. In: *ArXiv* abs/1807.07282 (2018).

[53] Moshe Kravchik and Asaf Shabtai. “Efficient Cyber Attack Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA”. In: *IEEE Transactions on Dependable and Secure Computing* (2021), pp. 1–1. DOI: 10.1109/TDSC.2021.3050101.

[54] *BATADAL Datasets*. https://www.batadal.net/data.html. [Online Accessed: 8-02-2022].

[55] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. “WADI: A Water Distribution Testbed for Research in the Design of Secure Cyber Physical Systems”. In: *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. CySWATER ’17. Pittsburgh,

Pennsylvania: Association for Computing Machinery, 2017, pp. 25–28. ISBN: 9781450349758. DOI: 10.1145/3055366.3055375. URL: https://doi.org/10.1145/3055366.3055375.

[56] Qinghua Xu, Shaukat Ali, and Tao Yue. "Digital Twin-based Anomaly Detection in Cyber-physical Systems". In: *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. 2021, pp. 205–216. DOI: 10.1109/ICST49551.2021.00031.

[57] Zahra Jadidi et al. "Automated detection-in-depth in industrial control systems". In: *The International Journal of Advanced Manufacturing Technology* 118.7 (Feb. 2022), pp. 2467–2479. ISSN: 1433-3015. DOI: 10.1007/s00170-021-08001-6. URL: https://doi.org/10.1007/s00170-021-08001-6.

[58] Jun Inoue et al. "Anomaly Detection for a Water Treatment System Using Unsupervised Machine Learning". In: *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. 2017, pp. 1058–1065. DOI: 10.1109/ICDMW.2017.149.

[59] Moshe Kravchik and Asaf Shabtai. "Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks". In: *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. CPS-SPC '18. Toronto, Canada: Association for Computing Machinery, 2018, pp. 72–83. ISBN: 9781450359924. DOI: 10.1145/3264888.3264896. URL: https://doi.org/10.1145/3264888.3264896.

[60] Dan Li et al. *Anomaly Detection with Generative Adversarial Networks for Multivariate Time Series*. 2018. DOI: 10.48550/ARXIV.1809.04758. URL: https://arxiv.org/abs/1809.04758.

[61] Mayra Macas and Chunming Wu. "An Unsupervised Framework for Anomaly Detection in a Water Treatment System". In: *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. 2019, pp. 1298–1305. DOI: 10.1109/ICMLA.2019.00212.

[62] Jonguk Kim, Jeong-Han Yun, and Hyoung Chun Kim. *Anomaly Detection for Industrial Control Systems Using Sequence-to-Sequence Neural Networks*. 2019. DOI: 10.48550/ARXIV.1911.04831. URL: https://arxiv.org/abs/1911.04831.

[63] Ángel Luis Perales Gómez et al. "MADICS: A Methodology for Anomaly Detection in Industrial Control Systems". In: *Symmetry* 12.10 (2020). ISSN: 2073-8994. DOI: 10.3390/sym12101583. URL: https://www.mdpi.com/2073-8994/12/10/1583.

[64] SungJin Kim, WooYeon Jo, and Taeshik Shon. "APAD: Autoencoder-based Payload Anomaly Detection for industrial IoE". In: *Applied Soft Computing* 88 (2020), p. 106017. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2019.106017. URL: https://www.sciencedirect.com/science/article/pii/S1568494619307999.

[65] M.R. Gauthama Raman, Wenjie Dong, and Aditya Mathur. "Deep autoencoders as anomaly detectors: Method and case study in a distributed water treatment plant". In: *Computers Security* 99 (2020), p. 102055. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2020.102055. URL: https://www.sciencedirect.com/science/article/pii/S016740482030328X.

[66] Masafumi Matta et al. "Industrial Control System Monitoring Based on Communication Profile". In: *JOURNAL OF CHEMICAL ENGINEERING OF JAPAN* 48.8 (2015), pp. 619–625. ISSN: 0021-9592. DOI: 10.1252/jcej.14we323. URL: https://ci.nii.ac.jp/naid/130005093996/en/.

[67] David Myers et al. "Anomaly detection for industrial control systems using process mining". In: *Computers Security* 78 (2018), pp. 103–125. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2018.06.002. URL: https://www.sciencedirect.com/science/article/pii/S0167404818306795.

[68] *WADI-DATASET-Dhalsim - github*. https://github.com/thehoodbuddha/WADI-DATASET-Dhalsim. [Online Accessed: 18-05-2022].

[69] Xuanhao Chen et al. "DAEMON: Unsupervised Anomaly Detection and Interpretation for Multivariate Time Series". In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 2021, pp. 2225–2230. DOI: 10.1109/ICDE51399.2021.00228.

[70] Paulo Freitas de Araujo-Filho et al. "Intrusion Detection for Cyber–Physical Systems Using Generative Adversarial Networks in Fog Environment". In: *IEEE Internet of Things Journal* 8.8 (2021), pp. 6247–6256. DOI: 10.1109/JIOT.2020.3024800.

[71] Ahmed Abdulaal and Tomer Lancewicki. “Real-Time Synchronization in Neural Networks for Multivariate Time Series Anomaly Detection”. In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 3570–3574. DOI: 10.1109/ICASSP39728.2021.9413847.

[72] Raihan Ul Islam, Mohammad Shahadat Hossain, and Karl Andersson. “A novel anomaly detection algorithm for sensor data under uncertainty”. In: *Soft Computing* 22.5 (Mar. 2018), pp. 1623–1639. ISSN: 1433-7479. DOI: 10.1007/s00500-016-2425-2. URL: https://doi.org/10.1007/s00500-016-2425-2.

[73] Aditya P. Mathur and Nils Ole Tippenhauer. “SWaT: a water treatment testbed for research and training on ICS security”. In: *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. 2016, pp. 31–36. DOI: 10.1109/CySWater.2016.7469060.

[74] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P. Mathur. “WADI: A Water Distribution Testbed for Research in the Design of Secure Cyber Physical Systems”. In: *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*. CySWATER ’17. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2017, pp. 25–28. ISBN: 9781450349758. DOI: 10.1145/3055366.3055375. URL: https://doi.org/10.1145/3055366.3055375.

[75] Riccardo Taormina et al. “Battle of the Attack Detection Algorithms: Disclosing Cyber Attacks on Water Distribution Networks”. In: *Journal of Water Resources Planning and Management* 144.8 (2018), p. 04018048. DOI: 10.1061/(ASCE)WR.1943-5452.0000969.

[76] Nicholas R. Rodofile et al. “Process Control Cyber-Attacks and Labelled Datasets on S7Comm Critical Infrastructure”. In: *Information Security and Privacy*. Ed. by Josef Pieprzyk and Suriadi Suriadi. Cham: Springer International Publishing, 2017, pp. 452–459. ISBN: 978-3-319-59870-3.

[77] *Secure Water Treatment - Technical detail v4.4.3*. https://itrust.sutd.edu.sg/wp-content/uploads/sites/3/2021/07/SWaT_technical_details-160720-v4.4.pdf. [Online Accessed: 16-05-2022].

[78] *Water Distribution (WADI) Testbed - Itrust*. https://itrust.sutd.edu.sg/wp-content/uploads/sites/3/2018/10/WADI_technical_details-161018-v1.2.pdf. [Online Accessed: 23-03-2022].

[79] *Scapy Project*. https://scapy.net/index. [Online Accessed: 16-05-2022].

[80] *NFdump - Github*. https://github.com/phaag/nfdump. [Online Accessed: 16-05-2022].

[81] *Pcap2zeek - github*. https://github.com/thehoodbuddha/PCAP2ZEEK. [Online Accessed: 22-06-2022].

[82] *Zeek Plugin ENIP - Github*. https://github.com/amzn/zeek-plugin-enip. [Online Accessed: 18-05-2022].

[83] *Zeek Plugin S7comm - Github*. https://github.com/amzn/zeek-plugin-s7comm. [Online Accessed: 18-05-2022].

[84] *Zeek Analysis Tools - Github*. https://github.com/SuperCowPowers/zat. [Online Accessed: 18-05-2022].

[85] Hamid Reza Ghaeini and Nils Ole Tippenhauer. “HAMIDS: Hierarchical Monitoring Intrusion Detection System for Industrial Control Systems”. In: *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. CPS-SPC ’16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 103–111. ISBN: 9781450345682. DOI: 10.1145/2994487.2994492. URL: https://doi.org/10.1145/2994487.2994492.

[86] *Github: Multi-domain-Cyber-attack-Detection-in-Industrial-Control-Systems-Resources*. https://github.com/thehoodbuddha/Multi-domain-Cyber-attack-Detection-in-Industrial-Control-Systems-Resources. [Online Accessed: 23-08-2022].

[87] *Why Choose Zeek? - ZEEK*. https://zeek.org/wp-content/uploads/2021/10/why_choose_zeek.pdf. [Online Accessed: 1-08-2022].

[88] Zheng Li et al. “ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions”. In: *IEEE Transactions on Knowledge and Data Engineering* (2022), pp. 1–1. DOI: 10.1109/TKDE.2022.3159580.

[89] George Box. "Box and Jenkins: Time Series Analysis, Forecasting and Control". In: Jan. 2013, pp. 161–215. ISBN: 978-1-349-35027-8. DOI: 10.1057/9781137291264_6.

[90] Ruey S. Tsay. "Time Series and Forecasting: Brief History and Future Research". In: *Journal of the American Statistical Association* 95.450 (2000), pp. 638–643. ISSN: 01621459. URL: http://www.jstor.org/stable/2669408 (visited on 08/01/2022).

[91] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. English. 2nd. Australia: OTexts, 2018.

[92] Yuchuan Lai and David A. Dzombak. "Use of the Autoregressive Integrated Moving Average (ARIMA) Model to Forecast Near-Term Regional Temperature and Precipitation". In: *Weather and Forecasting* 35.3 (2020), pp. 959–976. DOI: 10.1175/WAF-D-19-0158.1. URL: https://journals.ametsoc.org/view/journals/wefo/35/3/waf-d-19-0158.1.xml.

[93] D.C. Montgomery, C.L. Jennings, and M. Kulahci. *Introduction to Time Series Analysis and Forecasting*. Wiley Series in Probability and Statistics. Wiley, 2015. ISBN: 9781118745151. URL: https://books.google.nl/books?id=Xeh8CAAAQBAJ.

[94] Aadyot Bhatnagar et al. "Merlion: A Machine Learning Library for Time Series". In: (2021). arXiv: 2109.09265 [cs.LG].

[95] *Merlion - github*. https://github.com/salesforce/Merlion. [Online Accessed: 12-07-2022].

[96] V. Assimakopoulos and K. Nikolopoulos. "The theta model: a decomposition approach to forecasting". In: *International Journal of Forecasting* 16.4 (2000). The M3- Competition, pp. 521–530. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(00)00066-2. URL: https://www.sciencedirect.com/science/article/pii/S0169207000000662.

[97] Rb Cleveland et al. "STL: A seasonal-trend decomposition procedure based on loess (with discussion)". In: 1990.

[98] Yue Zhao, Zain Nasrullah, and Zheng Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[99] *Python Outlier Detection (PyOD) - github*. https://github.com/yzhao062/Pyod. [Online Accessed: 18-07-2022].

[100] Bin Zhu and Ali A. Ghorbani. "Alert Correlation for Extracting Attack Strategies". In: *Int. J. Netw. Secur.* 3 (2006), pp. 244–258.

[101] Claudio Toshio Kawakani, Sylvio Barbon Junior, and Rodrigo Sanches Miani. "Intrusion Alert Correlation to Support Security Management". In: *Proceedings of the XII Brazilian Symposium on Information Systems on Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era - Volume 1*. SBSI 2016. Florianopolis, Santa Catarina, Brazil: Brazilian Computer Society, 2016, pp. 313–320. ISBN: 9788576693178.

[102] *MongoDB*. https://www.mongodb.com/docs/. [Online Accessed: 17-08-2022].

[103] Oskar Triebe et al. "NeuralProphet: Explainable Forecasting at Scale". In: *CoRR* abs/2111.15397 (2021). arXiv: 2111.15397. URL: https://arxiv.org/abs/2111.15397.

[104] Jonathan Goh et al. "A Dataset to Support Research in the Design of Secure Water Treatment Systems". In: *Critical Information Infrastructures Security*. Ed. by Grigore Havarneanu et al. Cham: Springer International Publishing, 2017, pp. 88–99. ISBN: 978-3-319-71368-7.

[105] *Batadal datasets*. https://www.batadal.net/data.html. [Online Accessed: 6-06-2022].

[106] *2017QUT$_S$7comm – Github*. https://github.com/qut-infosec/2017QUT_S7comm. [Online Accessed: 26-05-2022].

[107] *2017QUT$_S$7commV2 – CloudStor*. https://cloudstor.aarnet.edu.au/plus/index.php/s/9qFfeVmfX7K5IDH. [Online Accessed: 16-05-2022].

Appendices

A Datasets

A.1 SWaT

Name set SWaT	*Physical data*	*(raw) Network data*	*Labelled*	*Number of Attacks*	*Size*	*Description*	*Notes*
SWaT.A1 & A2_Dec 2015	Yes	No	Yes	41	7 days of normal capture 4 days of attack capture (under different scenarios) In total, 946 722 samples compromising of 51 attributes were collected over 11 days.	Data collection for total of 11 days of which SWaT was functioning non-stop 24 hours/day. During the first seven days no attack was launched, while for the last 4 days attacks scenarios are included. During 4 days capture 41 attacks were launched. Capture range: start: 4:00 PM 22/12/2015 end: 3:00 PM 2/1/2015	Dataset contains network traffic data. However, the provided data are not raw packet captures. Contains various attack strategies for full-description of the attack strategies we refer to the publication by Goh et al. [104].
SWaT.A3_Jun 2017	yes	yes	No	0	In total: 13661 samples $\approx 300GB$ of packet capture	136 hours of network traffic and historian data from continuously running SWaT (no attacks) was collected over 6 days. Capture range: Physical data: start 4.23 PM 13/06/2017 8:40 PM 19/06/2017 (Plant stopped) Network traffic capturing time start: 6.30 PM 13/06/2017 restart: 12.40 PM 15/06/2017 end: 5.00 PM 22/06/2017	Dataset contains raw packet captures of the network traffic. However, this set does not include any attack data.
SWaT.A4 & A5_Jul 2019	Yes	No	Partially	6	In total: unkown number of samples	Dataset contains data of Plant normal run without any attacks: 12:35 PM to 2:50 PM. Six attacks, were carried out between 3:08 PM to 4:16 PM Capture range: start: 12:35 PM 20/07/2019 end: 4:35 PM 20/07/2019	No traffic data available for this set.
SWaT.A6_Dec 2019	Yes	Yes	Partially	6	In total: 13201 samples 41.5 GB of packet capture	The dataset consists of pcap and Historian Data (.csv) files. The dataset records a series of malware infection attacks on the SWaT Engineering Workstation. The malware attacks include Historian Data Exfiltration attack and Process Disruption attacks. Capture range: start: 10:05 PM 06/12/2019 end: 13:45 PM 06/12/2019	Only 15-minutes of normal operation. According to the documentation 3 hour of normal operation is present. However, this is not True if we consider network data.
SWaT.A7_Jun 2020	Yes	No	No	0	In total: 4 separate captures with variable sample sizes.	SWaT was run on 4 occasions (no attack) Capture range: date: 22/06/2020 and 29/06/2020 Each run lasted either 2 or 4 hours. Network traffic data was captured for the 4 runs.	Although stated; no network traffic data could be found for the runs.
5 Day PCAP	No	Yes	No	Unkown	Unkown	Packet capture over a five-day period Capture range: Unkown	Limited documentation available for this set. Unclear what the 5 day PCAP respre- sents and if it contains attack data.

Table 13: Overview of SWaT datasets.

A.2 WADI

Name dataset Wadi	*Physical data*	*(raw) Network data*	*Labelled*	*Number of Attacks*	*Size*	*Description*	*Notes*
WADI.A1-9 OCT 2017	Yes	No	No	15	In total: 120905 samples	14 days of normal operation and two days of different attack scenarios. Furthermore, contains data of 123 sensor and actuators. During the two days of attack a total of 15 attack were launched.	This set does not contain any network traffic data.
WADI.A2_19 Nov 2019	Yes	No	Yes	15	In total: 784571 samples (WADI_14day_new.csv) 172803 samples (WADI_attackdataLable .csv)	14 days of normal operation and two days of different attack scenarios. Furthermore, contains data of 123 sensor and actuators. During the two days of attack a total of 15 attack were launched. ”As the plant was unstable for certain periods during the operation, the affected readings have been removed and a new csv file “WADI_14days_new.csv” uploaded. A second csv file “WADI_attackdataLABLE.csv” now contains labels on whether there was an attack (-1) or not (1).The updated attack table with the corrected dates has also been uploaded.”	This set does not contain any network traffic data. This set contains update to the previous WADI. A1-9 Oct 2017 set.

Table 14: Overview of WADI datasets.

A.3 Batadal

Name set Batadal	*Physical data*	*(raw) Network data*	*Labelled*	*Number of Attacks*	*Size*	*Description*	*Notes*
Training dataset 1	yes	no	yes	0	8761 samples 44 features 2.9MB log files.	"This dataset was released on November 20 2016, and it was generated from a one-year-long simulation. The dataset does not contain any attacks, i.e. all the data pertains to C-Town normal operations. A key aspect of the data set was the absence of cyber attacks, which made it is suitable for studying the operations of the water distribution system under normal operating conditions." Sampling interval = 1 hour	Can be obtained from [105]
Training dataset 2	yes	No	yes	7	4177 samples 44 features 1.2MB log files	"This dataset with partially labeled data was released on November 28 2016. The dataset is around 6 months long and contains several attacks, some of which are approximately labeled. Training Data Set 2 contained 7 attacks, spanning 492 hourly time steps. One attack was entirely revealed to the participants (by appropriately labeling the corresponding time steps), whereas the remaining attacks were either partially revealed or hidden. This corresponds to a postattack the scenario in which forensics experts carry out an investigation to determine whether, when, and where the water the distribution system has been affected. " Sampling interval = 1 hour	Can be obtained from [105]
Test dataset	yes	no	yes	7	2089 samples 44 features 383.8kB log files	"This 3-months long dataset contains several attacks but no labels. The dataset was released on February 20 2017, and it is used to compare the performance of the algorithms (see rules document for details)." Sampling interval = 1 hour	Can be obtained from [105]

Table 15: Overview of Batadal datasets.

A.4 S7comm Factory

Name set S7comm	*Physical data*	*(raw) Network data*	*Labelled*	*Number of Attacks*	*Size*	*Description*	*Notes*
2017QUT_S7comm (V1)	yes	yes	yes	21	Unkown (no clear log files)	"As we have discussed the attacks that were implemented against the test-bed, in this section, we present our attack datasets. The attack dataset, collected approximately over 9 h, consists of network traffic and process log data with 30 processes. The control dataset contains approximately 8.5 h of network traffic and process log data, with 32 processes. The datasets consist of network traffic from the perspective of the Master switch and the HMI. [76]"	This dataset is what we consider the base version. The obtainable dataset is to a limited extent documented and contains many subfiles. More assessment required Can be obtained from [106]
2017QUT_S7_Dataset (V2)	yes	yes	yes	21	3.4GB network data 368 MB log files.	Dataset contains "ControlDataset" containing control dataset of S7-1200 PLC, and series of logs from the four devices depicted in Fig. 18. "AttackDataset" contains data during several attacks on the ICS architecture. Both contain PCAPs and device logs for each individual component. "A total of 21 cyber attacks were conducted on the Siemens S7-1200 PLCs. These attacks comprised of 17 unique attacks, consisting of two major types; injection attacks, and flooding attacks, both with the goal of changing or disrupting the running industrial process."	This dataset version is based on the the version below: 2017QUT_S7Comm and contains the same attack strategies. Exact differences between the two are unknown. However, we assume this version has been pre-processed based on the version given below. Can be obtained from [107]

Table 16: Overview of S7comm factory mining refinery datasets.

A.5 DHALSIM

Name	Description	Simulation Iterations	Elapsed Time (h:m:s)	Notes
normal-operation-no-noise	Represents small dataset without any active attacks.	6000	0:57:24	
normal-operation-no-noise-medium	Represents same configuration as the dataset above. However, has increased iterations to simulate over an extended period.	12000	2:17:07	
Attack-mitm-medium	Direct device attack on valve: *V_ER2i* sending open command and MITM targeting sensor *T2* spoofing a value of 0.0. Both attacks between iterations: 2500-3500.	12000	2:31:26	
Attack-device-between-small	Experiment with attacks targeting pump: *P_RAW1* between iteration 500-1000. Furthermore, two attacks based on triggers: close *P_RAW1* if $T2 < 0.16$ close *P_RAW1* if $0.10 < T20.16$.	1500	0:14:44	
batches-no-attack-different-initial-tank	The experiment runs without any active attacks. Furthermore, batch mode activated for running the same experiment 10 times but with different initial tank levels.	1000	1:16:43	batch size: 10

Table 17: Custom synthetic generated ICS water distribution data using DHALSIM digital twin.

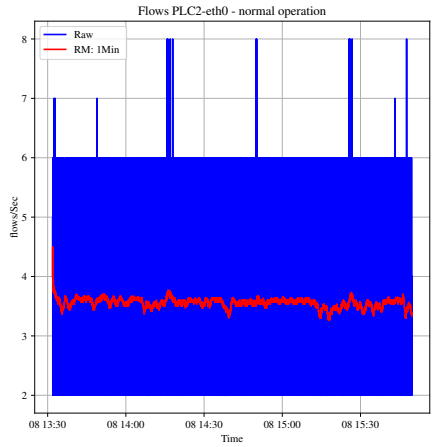
B Feature Exploration Extended

B.1 Zeek Logs overview

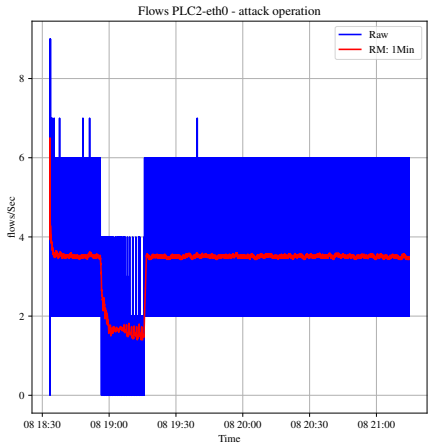
Log file	***Fields present in Log File***
arp.log	timstamp, operation, source mac address, destination mac address, src IP, dst IP, source hardware address, response hardware address
conn.log	timestamp, uid, src IP, dst IP, src port, dst port, protocol, service, duration, src bytes, resp bytes, connection state, local_orig, local_resp, missed_bytes (packet loss), history, src packets, src bytes, resp packets, resp bytes, tunnel_parents
dns.log	timestamp, uid, src IP, dst IP, src port, dst port, protocol, trans_id, rtt, query, qclass, qclass_name, qtype, qtype_name. rcode, rcode_name, AA, TC, RD RA, Z, answers, TTLs, rejected
http.log	timestamp, uid, src IP, dst IP, src port, dst port, trans_depth, method, host, uri, referrer, version, user_agent, origin, request_body_len, response_body_len, status_code, status_msg, info_code, info_msg, tags, username, password, proxied, orig_fluids, orig_filenames, orig_mime_types, resp_fluids, resp_filenames, resp_mime_types
cip.log	timestamp, uid, src IP, dst IP, src port, dst port, cip_sequence_count, direction, cip_service_code, cip_service, cip_status, class_id, class_name, instance_id, attribute_id
enip.log	timestamp, uid, src IP, dst IP, src port, dst port, enip_command_code, enip_command, length, session_handle, enip_status, sender_context, options
s7comm.log	timestamp, uid, src IP, dst IP, src port, dst port, rosctr, parameter, item_count, data_info
iso_cotp.log	timestamp, uid, src IP, dst IP, src port, dst port, pdu_type

Table 18: Attribute overview Zeek logs.

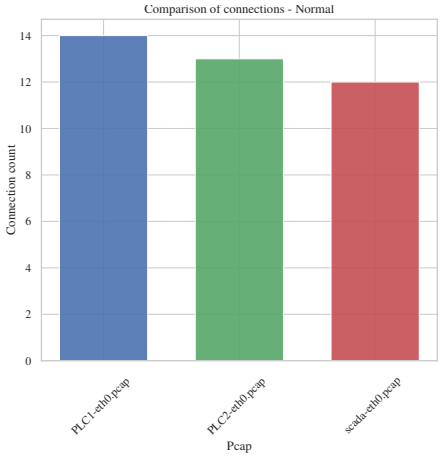
B.2 DHALSIM - Netflow V5



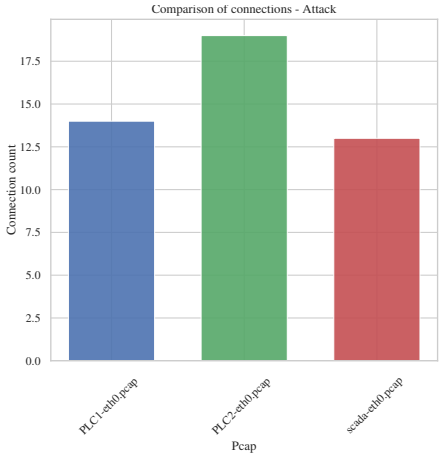
(a) Flows DHALSIM - normal.



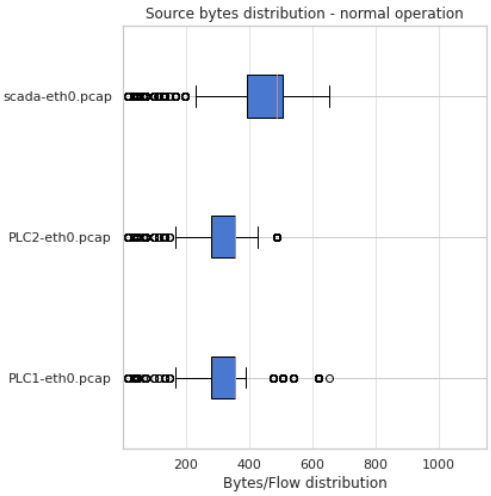
(b) Flows DHALSIM - attack.



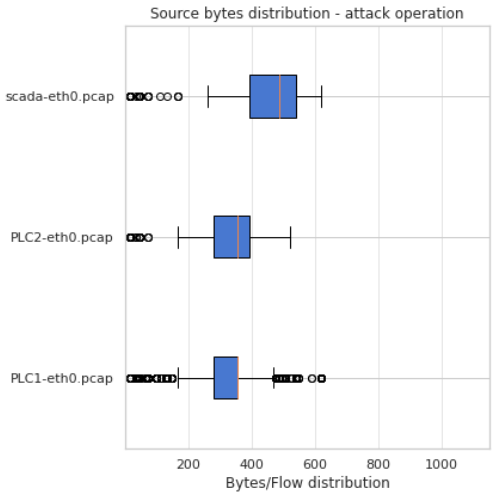
(c) Unique connections - normal.



(d) Unique connections - attack.

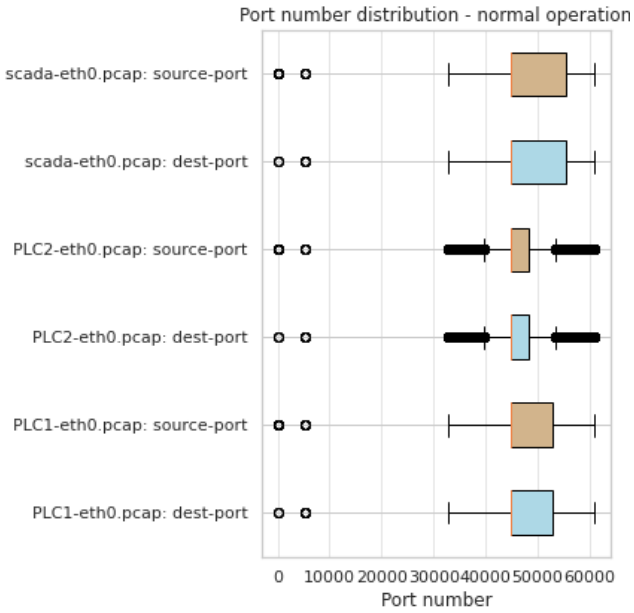


(e) Flows/connection - normal.

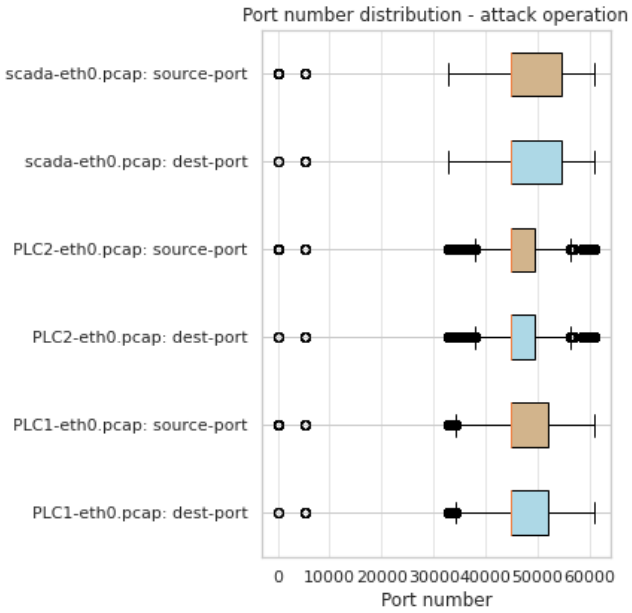


(f) Flows/connection - attack.

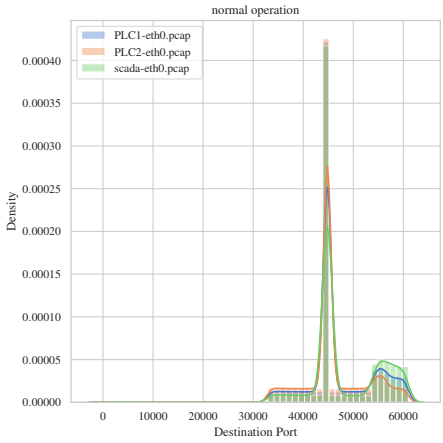
Figure 41: Dhalsim Netflow V5 feature set exploration - extended.



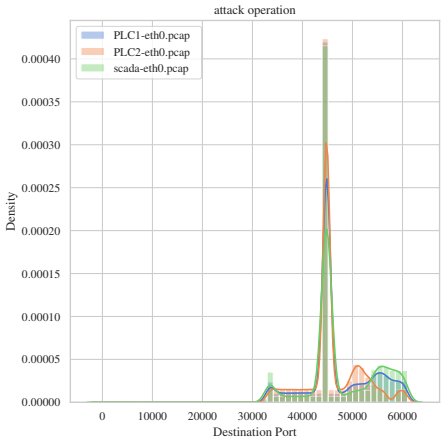
(a) Port number distribution - normal.



(b) Port number distribution - attack.



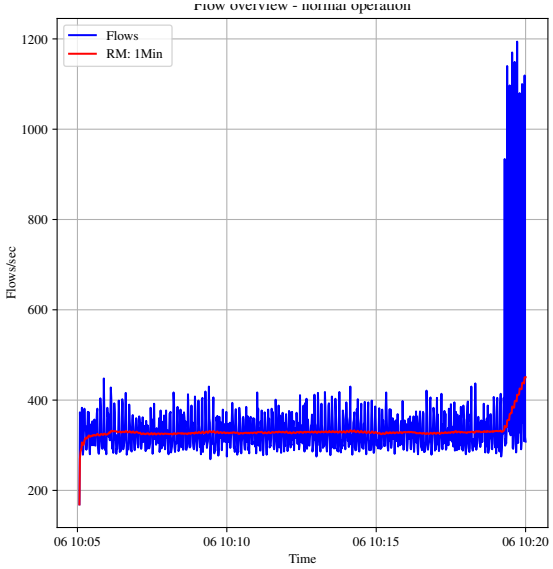
(c) Port number density (destination) - normal.



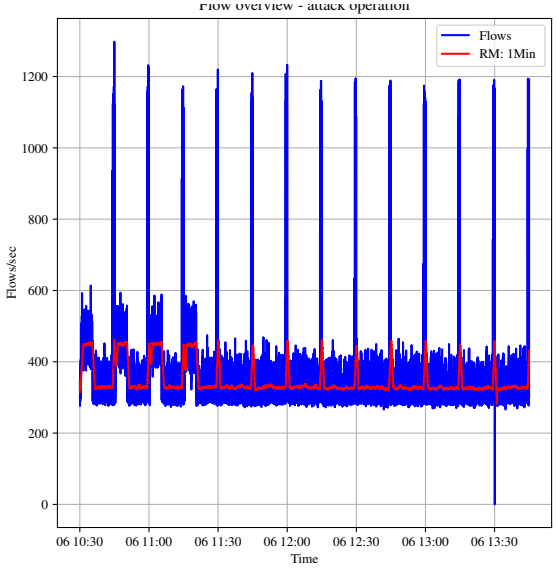
(d) Port number density (destination) - attack.

Figure 42: DHALSIM Netflow V5 feature set exploration - extended, second page.

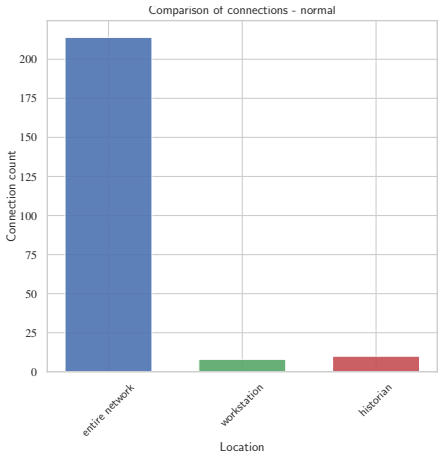
B.3 SWaT A6 2019 - Netflow V5



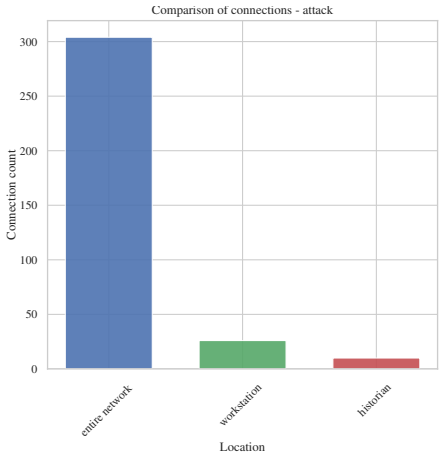
(a) Flows DHALSIM - normal.



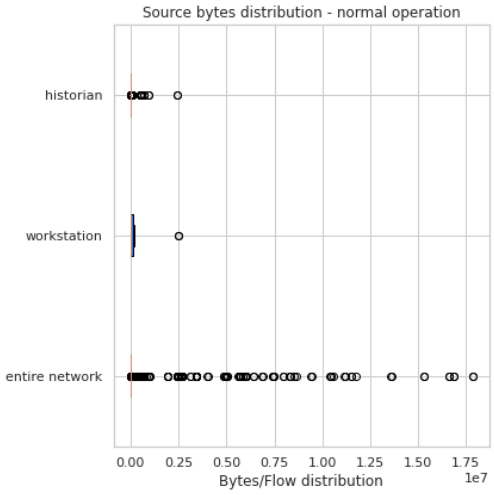
(b) Flows DHALSIM - attack.



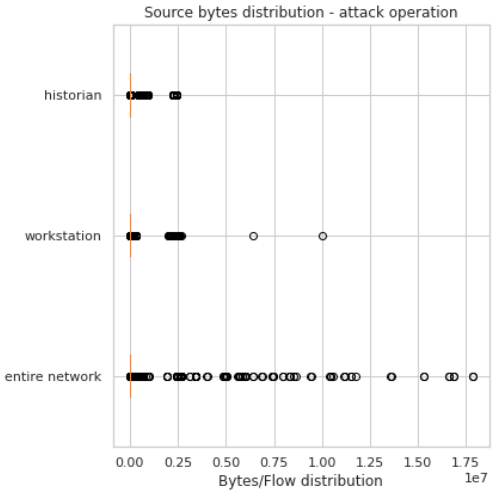
(c) Unique connections - normal.



(d) Unique connections - attack.

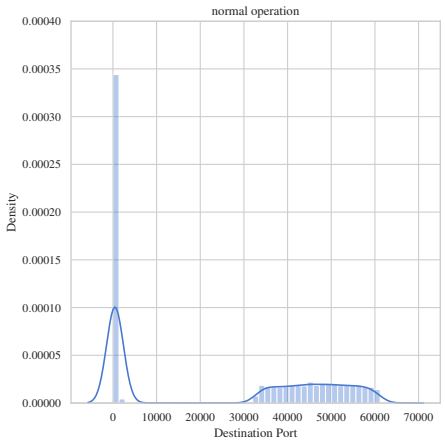


(e) Flows/connection - normal.

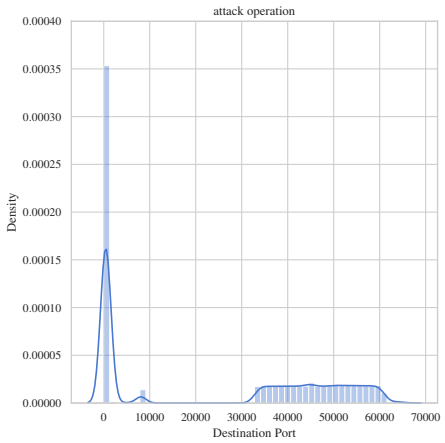


(f) Flows/connection - attack.

Figure 43: Feature and dataset exploration Netflow V5 *SWaT A6 2019*.



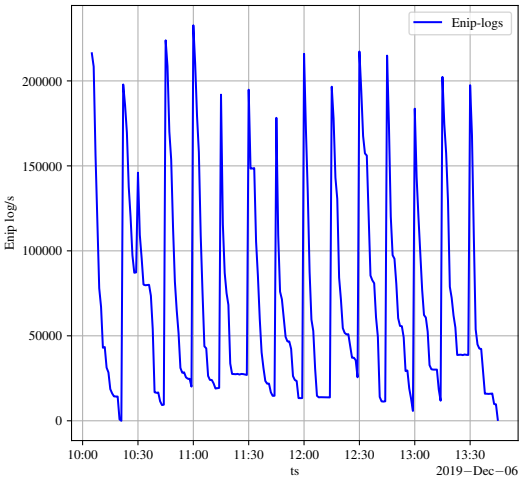
(a) Port number density (destination) - normal.



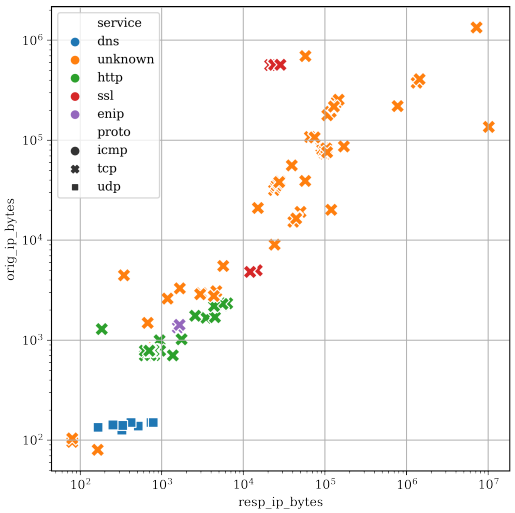
(b) Port number density (destination) - attack.

Figure 44: Feature and dataset exploration Netflow V5 *SWaT A6 2019* - 2.

B.4 SWaT A6 2019 - Zeek



(a) Enip communication events over entire *SWaT A6 2019* network *enip.log*.



(b) send vs receive in Bytes - *Workstation conn.log*.

Figure 45: Feature and dataset exploration Zeek *SWaT A6 2019* - 2.

C Evaluation

C.1 Hyper-parameters Evaluation

Component	*Training Interval*	*Test Interval*	*Traing Size*	*Test Size*	*SARIMA Model*	*Training Duration*
PLC1 - RAW Water	10:25-12:29	12:29-13:44	249	151	0, 0, 0)(0, 1, 0, 30)	0:00:11.839214
PLC1 - RAW Water - Secondary	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.102516
PLC2 - Pre Treatment	10:25-12:29	12:29-13:44	249	151	(0, 0, 0)(2, 1, 0, 30)	0:03:29.000795
PLC3 - Ultra-Filtration	10:25-12:29	12:29-13:44	249	151	(0, 0, 0)(2, 1, 0, 30)	0:03:30.285986
PLC4 - De-Chlorination	10:25-12:29	12:29-13:44	249	151	(1, 0, 2)(1, 1, 0, 30)	0:03:50.871373
PLC5 - Reverse Osmosis	10:25-12:29	12:29-13:44	249	151	(0, 0, 3)(1, 1, 0, 30)	0:06:35.421532
PLC6 - RO Product	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:09.002361
PLC7 - SIS	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.103357
SCADA - Touch Panel	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.252585
SCADA - Historian Server	10:25-12:29	12:29-13:44	249	151	(3, 0, 3)(0, 1, 0, 30)	0:00:54.970352
SCADA - Engineering Workstation	10:05-10:25	10:25-13:44	41	399	(1, 0, 0)(0, 0, 0, 0)	0:00:00.601267
					Total	0:18:42.451339

(a) Exfiltration experiment SARIMA, Outgoing (*id.orig_h*).

Component	*Training Interval*	*Test Interval*	*Traing Size*	*Test Size*	*SARIMA Model*	*Training Duration*
PLC1 - RAW Water	10:25-12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:17.984289
PLC1 - RAW Water - Secondary	10:25:12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.276536
PLC2 - Pre Treatment	10:25:12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:14.365361
PLC3 - Ultra-Filtration	10:25:12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:29.431688
PLC4 - De-Chlorination	10:25:12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:02:36.627069
PLC5 - Reverse Osmosis	10:25:12:29	12:29-13:44	249	151	(2, 0, 0)(0, 1, 0, 30)	0:00:22.254033
PLC6 - RO Product	10:25:12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:15.886671
PLC7 - SIS	10:25:12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.131426
SCADA - Touch Panel	10:25:12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.109287
SCADA - Historian Server	10:25:12:29	12:29-13:44	249	151	(0, 0, 0)(0, 1, 0, 30)	0:00:08.466212
SCADA - Engineering Workstation	10:25:12:29	12:29-13:44	249	151	(2, 0, 2)(1, 1, 1, 30)	0:01:26.033019
					Total	0:09:51.565591

(b) Exfiltration experiment SARIMA, Incoming (*id.resp_h*).

Component	*Training Interval*	*Test Interval*	*Traing Size*	*Test Size*	*Training Duration*
PLC1 - RAW Water	10:25-12:20	12:20-13:44	107	52	0:00:00.004207
PLC2 - Pre Treatment	10:25-12:20	12:20-13:44	62	31	0:00:00.003487
PLC3 - Ultra-Filtration	10:25-12:20	12:20-13:44	62	34	0:00:00.003529
PLC4 - De-Chlorination	10:25-12:20	12:20-13:44	35	18	0:00:00.003323
PLC5 - Reverse Osmosis	10:25-12:20	12:20-13:44	25	12	0:00:00.003141
PLC6 - RO Product	10:25-12:20	12:20-13:44	43	23	0:00:00.003318
SCADA - Touch Panel	10:25-12:20	12:20-13:44	70	35	0:00:00.003429
SCADA - Historian Server	10:25-12:20	12:20-13:44	621	396	0:00:00.003752
SCADA - Engineering Workstation	10:25-12:20	12:20-13:44	75358	189	0:00:00.036213
			Total		0:00:00.064400

(c) Second Malware Download Experiment ECOD, Outgoing (*id.orig_h*).

Component	*Training Interval*	*Test Interval*	*Traing Size*	*Test Size*	*Training Duration*
PLC1 - RAW Water	10:25-1220	1220-1344	1182	781	0:00:00.003784
P1 - RAW Water - Secondary	10:25-1220	12:20-13:44	10	5	0:00:00.003784
PLC2 - Pre Treatment	10:25-1220	12:20-13:44	2218	1409	0:00:00.004383
PLC3 - Ultra-Filtration	10:25-1220	12:20-13:44	2213	1399	0:00:00.004861
PLC4 - De-Chlorination	10:25-1220	12:20-13:44	3235	2054	0:00:00.005615
PLC5 - Reverse Osmosis	10:25-1220	12:20-13:44	1223	771	0:00:00.005206
PLC6 - RO Product	10:25-1220	12:20-13:44	2202	1395	0:00:00.004352
SCADA - Historian Server	10:25-1220	12:20-13:44	1492913	924313	0:00:00.880833
SCADA - Engineering Workstation	10:25-1220	12:20-13:44	383	252	0:00:00.004428
			Total		0:00:00.916647

(d) Second Malware Download Experiment ECOD, Incoming (*id.resp_h*).

Component	*Training Interval*	*Test Interval*	*Traing Size*	*Test Size*	*SARIMA Model*	*Training Duration*
PLC1 - RAW Water	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.275305
PLC1 - RAW Water - Secondary	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.103556
PLC2 - Pre Treatment	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.270207
PLC3 - Ultra-Filtration	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.297948
PLC4 - De-Chlorination	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.267758
PLC5 - Reverse Osmosis	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.262490
PLC6 - RO Product	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.262172
PLC7 - SIS	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.101598
SCADA - Touch Panel	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.273617
SCADA - Historian Server	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 1, 0, 30)	0:00:00.306503
SCADA - Engineering Workstation	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.102615
					Total	0:00:02.523769

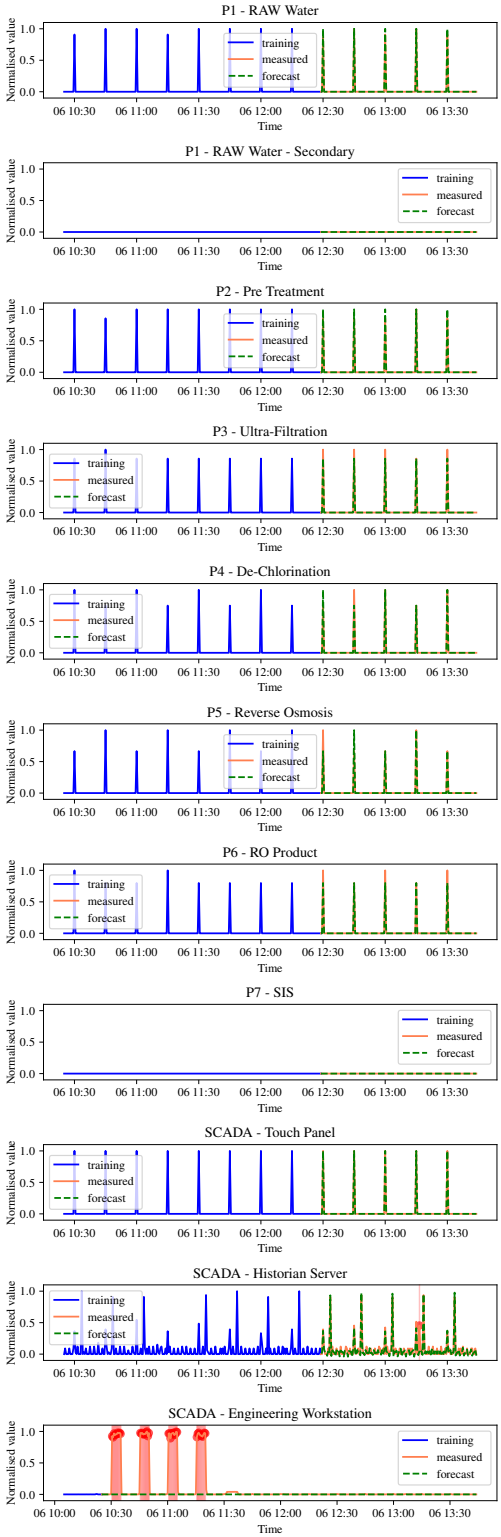
(e) Disruption experiment SARIMA, Outgoing (*id.orig_h*).

Component	*Training Interval*	*Test Interval*	*Traing Size*	*Test Size*	*SARIMA Model*	*Training Duration*
PLC1 - RAW Water	10:25-12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:12.602968
PLC1 - RAW Water - Secondary	10:25-12:29	12:29-13:44	249	151	(0, 0, 0)(0, 1, 0, 30)	0:00:00.282774
PLC2 - Pre Treatment	10:25-12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:02.932508
PLC3 - Ultra-Filtration	10:25-12v29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:24.755334
PLC4 - De-Chlorination	10:25-12:29	12:29-13:44	249	151	(2, 0, 1)(0, 1, 0, 30)	0:00:42.944321
PLC5 - Reverse Osmosis	10:25-12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:01:36.224020
PLC6 - RO Product	10:25-12:29	12:29-13:44	249	151	(0, 0, 1)(1, 1, 0, 30)	0:00:36.031126
PLC7 - SIS	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.132614
SCADA - Touch Panel	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.101284
SCADA - Historian Server	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.101405
SCADA - Engineering Workstation	10:25-12:29	12:29-13:44	249	151	(0, 0, 0](0, 0, 0, 0)	0:00:00.099059
					Total	0:06:36.207412

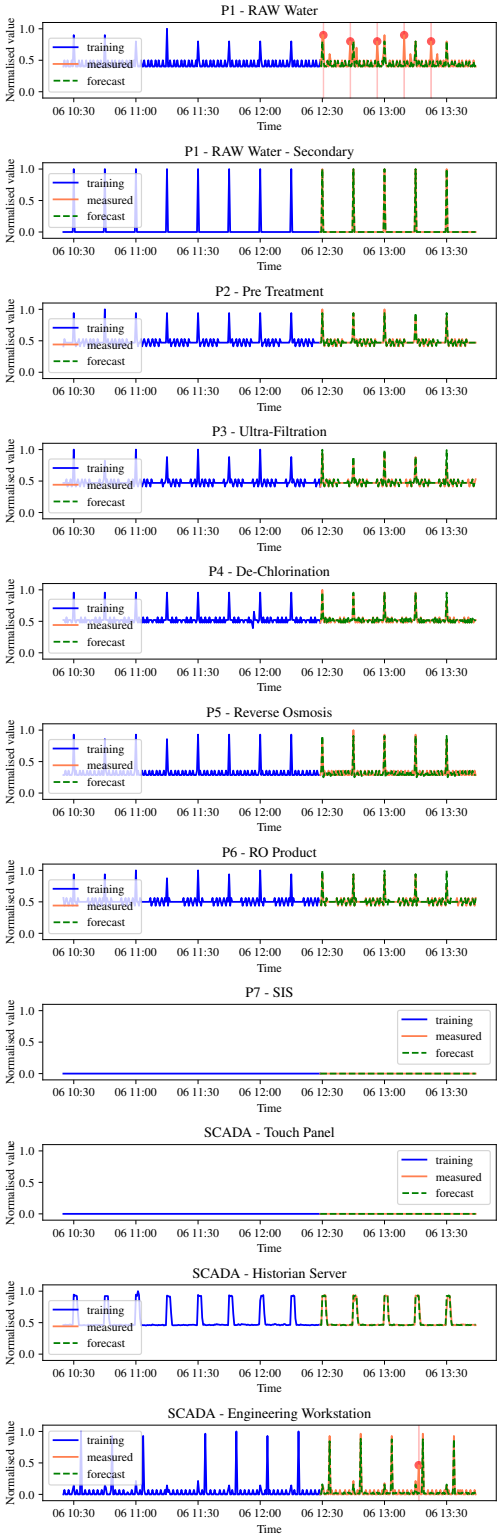
(f) Disruption experiment SARIMA, Incoming (*id.resp_h*).

Table 19: Training Intervals, testing Intervals, hyper-parameters and training duration for SARIMA Detector and ECOD detector against the three attack events.

C.2 Evaluation on Exfiltration of Historian Data Event - Extended



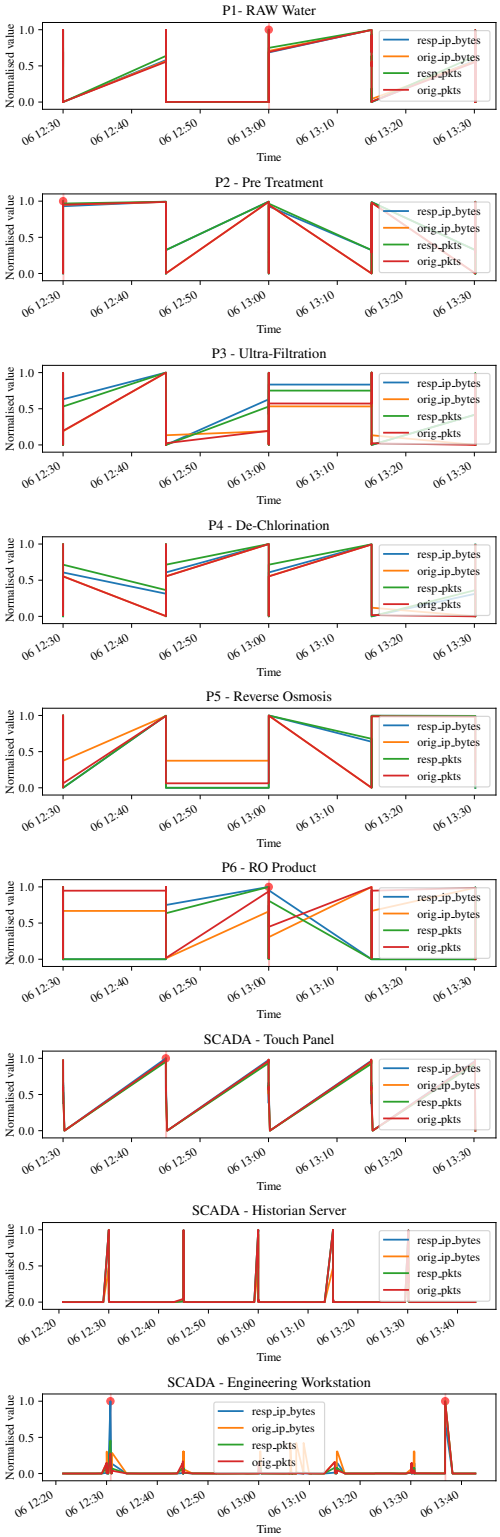
(a) Exfiltration Detection: *id.orig_h* (outgoing).



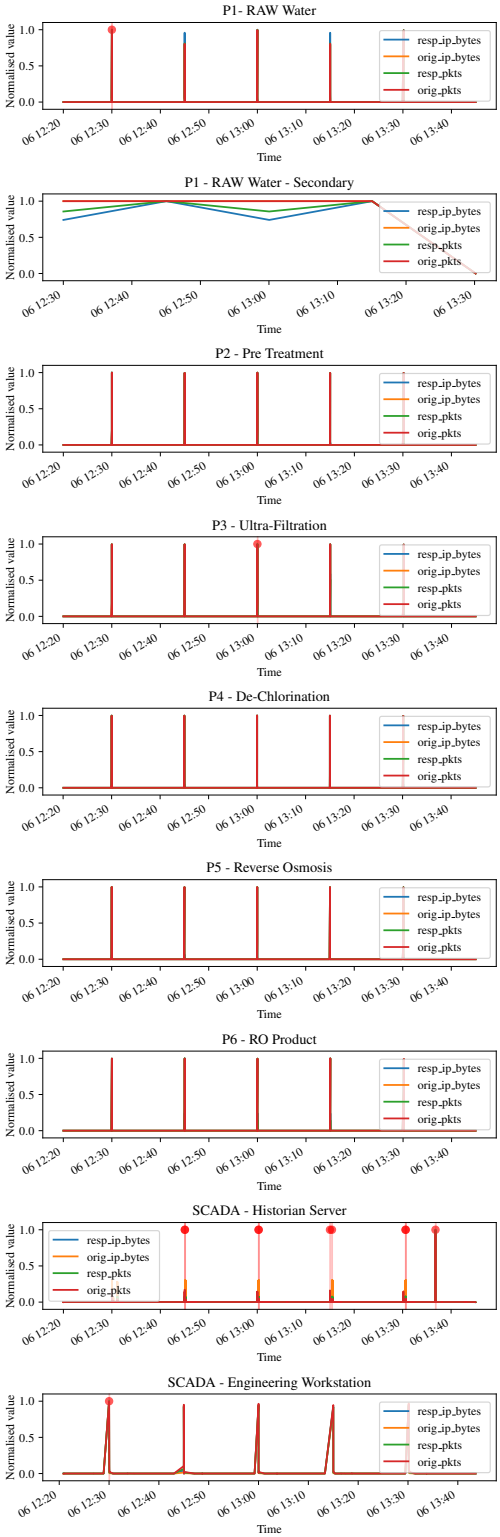
(b) Exfiltration Detection: *id.resp_h* (incoming).

Figure 46: Exfiltration experiment all components.

C.3 Evaluation on Second Malware Download Event - Extended



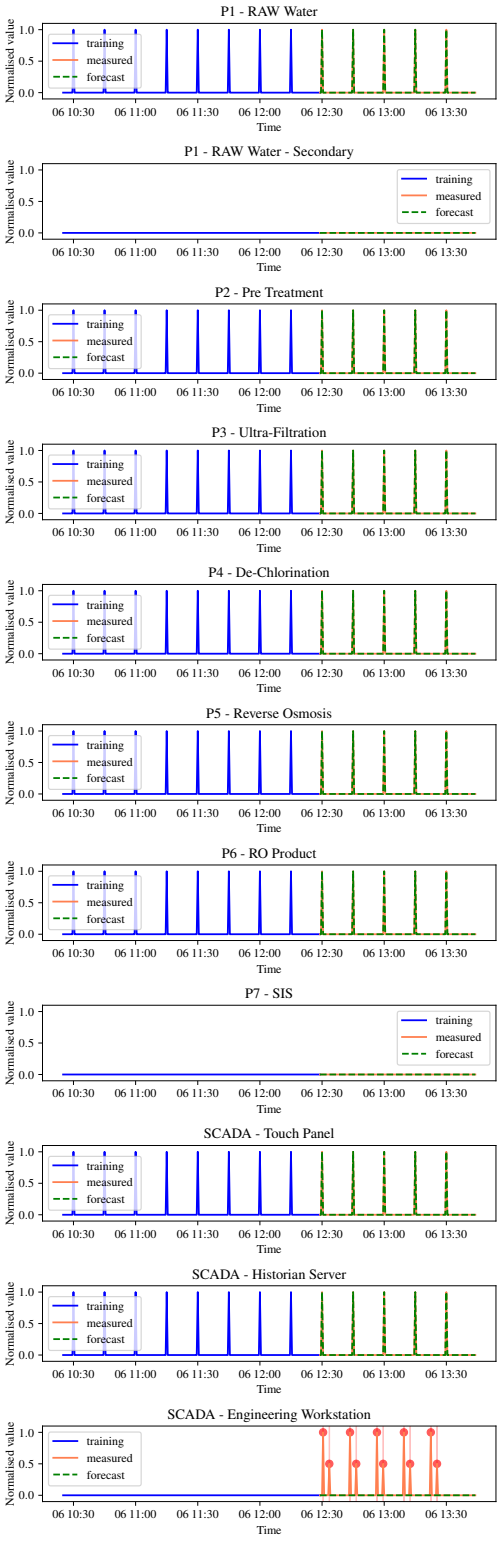
(a) ECOD detector *id.orig_h* (outgoing).



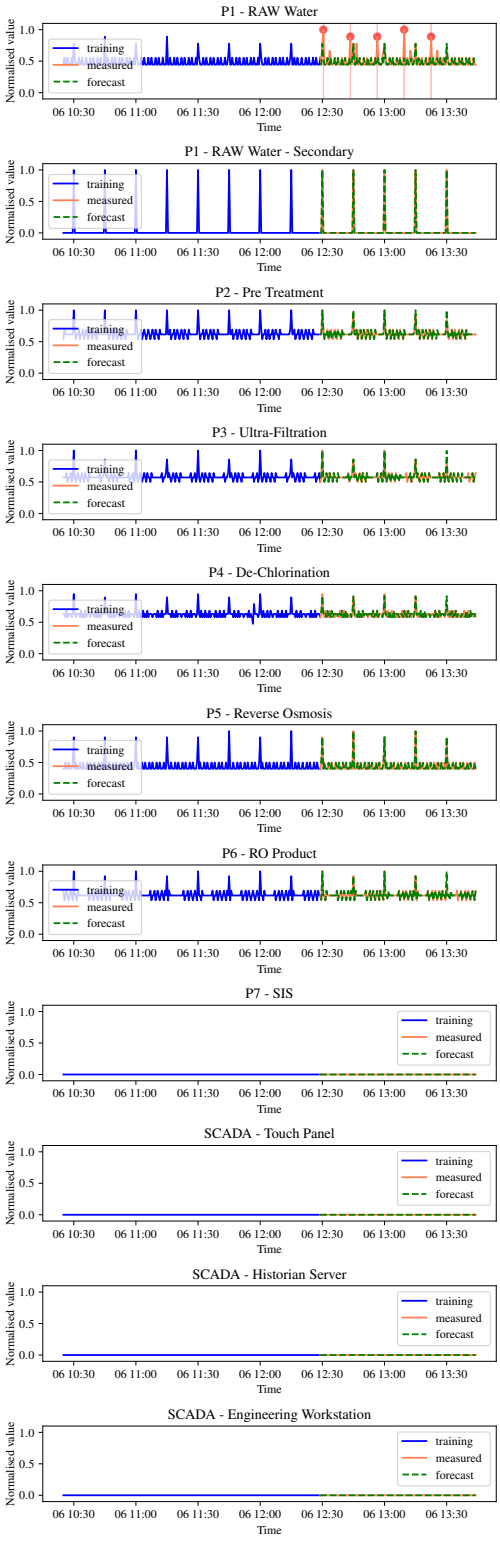
(b) ECOD detector *id.resp_h* (incoming).

Figure 47: ECOD detector experiment all components.

C.4 Evaluation on Disrupt Sensor Actuator Event - Extended



(a) Disruption Detection: *id.orig_h* (outgoing).



(b) Disruption Detection: *id.resp_h* (incoming).

Figure 48: Disruption experiment all components.

C.5 Merging Experiment

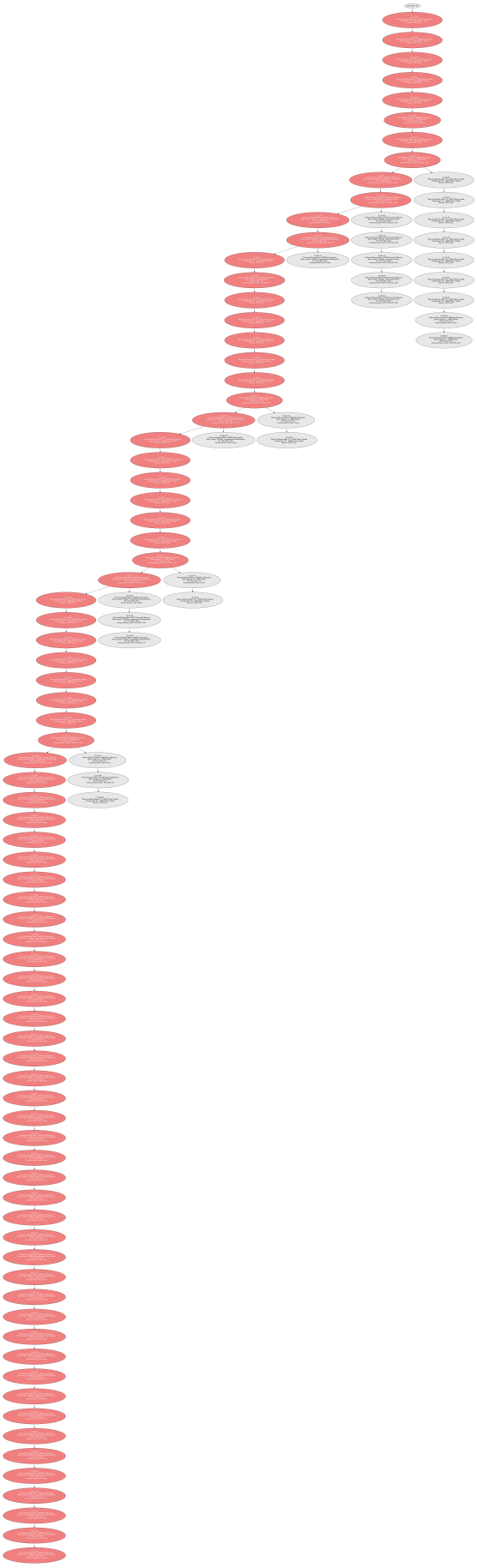


Figure 49: Attack extraction, longest alert path for *SWaT A6 2019*,