Anomaly Diagnosis in Industrial Control Systems for Digital Forensics

Marco Montaldi Cook

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

School of Computing Science

College of Science and Engineering
University of Glasgow

May 2023

Declaration

I certify that the thesis presented here for examination for PhD degree of the University of Glasgow is solely my own work other than where I have clearly indicated that it is the work of others and that the thesis has not been edited by a third party beyond what is permitted by the University's PGR Code of Practice.

Printed Name: *Marco Montaldi Cook*

Signature:

Abstract

Over several decades, Industrial Control Systems (ICS) have become more interconnected and highly programmable. An increasing number of sophisticated cyber-attacks have targeted ICS with a view to cause tangible damage. Despite the stringent functional safety requirements mandated within ICS environments, critical national infrastructure (CNI) sectors and ICS vendors have been slow to address the growing cyber threat. In contrast with the design of information technology (IT) systems, security of controls systems have not typically been an intrinsic design principle for ICS components, such as Programmable Logic Controllers (PLCs). These factors have motivated substantial research addressing anomaly detection in the context of ICS. However, detecting incidents alone does not assist with the response and recovery activities that are necessary for ICS operators to resume normal service. Understanding the provenance of anomalies has the potential to enable the proactive implementation of security controls, and reduce the risk of future attacks. Digital forensics provides solutions by dissecting and reconstructing evidence from an incident. However, this has typically been positioned from a post-incident perspective, which inhibits rapid triaging, and effective response and recovery, an essential requirement in critical ICS.

This thesis focuses on anomaly diagnosis, which involves the analysis of and discrimination between different types of anomalous event, positioned at the intersection between anomaly detection and digital forensics. An anomaly diagnosis framework is proposed that includes mechanisms to aid ICS operators in the context of anomaly triaging and incident response. PLCs have a fundamental focus within this thesis due to their critical role and ubiquitous application in ICS. An examination of generalisable PLC data artefacts produced a taxonomy of artefact data types that focus on the device data generated and stored in PLC memory. Using the artefacts defined in this first stage, an anomaly contextualisation model is presented that differentiates between cyber-attack and system fault anomalies. Subsequently, an attack fingerprinting approach (PLCPrint) generates near real-time compositions of memory fingerprints within $200ms$, by correlating the static and dynamic behaviour of PLC registers. This establishes attack type and technique provenance, and maintains the chain-of-evidence for digital forensic investigations. To evaluate the efficacy of the framework, a physical ICS testbed modelled on a water treatment system is implemented. Multiple PLC models are evaluated to demonstrate vendor neutrality of the framework. Furthermore, several gener-

alised attack scenarios are conducted based on techniques identified from real PLC malware. The results indicate that PLC device artefacts are particularly powerful at detecting and contextualising an anomaly. In general, we achieve high F1 scores of at least 0.98 and 0.97 for anomaly detection and contextualisation, respectively, which are highly competitive with existing state-of-the-art literature. The performance of PLCPrint emphasises how PLC memory snapshots can precisely and rapidly provide provenance by classifying cyber-attacks with an accuracy of 0.97 in less than $400ms$. The proposed framework offers a much needed novel approach through which ICS components can be rapidly triaged for effective response.

Acknowledgements

The work reported in this thesis would not have been possible without the help, support, and guidance of many people.

Firstly, I'd like to greatly thank my supervisor, Dr. Angelos Marnerides, for his unwavering support and guidance throughout the Ph.D. I have learnt so much under your supervision. I would also like to extend my thanks to Prof. Dimitrios Pezaros for all of his time and invaluable counsel over recent years.

A special thanks also to Prof. Chris Johnson for introducing me to the field of Industrial Control Systems, initiating this Ph.D, and providing support in the years he was at the University of Glasgow.

I would also like to extend my gratitude to my industrial supervisors at the Defence Science and Technology Laboratory (Dstl), especially Andrew Deacon, Sarah Johnson, and Paul Caseley, who have been instrumental in providing support and guidance throughout the thesis and enabling critical, and enjoyable, industrial placements. A special thanks to the entire Dstl cyber defence team for the laughs and all that you have taught me during my time spent with you, particularly Matt Jarrett. A special thanks goes to everyone who has provided feedback on publications and the writing of this thesis, in particular Prof. Bob Madahar.

An extended thank you to my Viva examiners, Prof. Luciano Gaspary and Dr. José Cano Reyes, for providing very constructive feedback on the thesis. Thanks also to the Viva convener Dr. Oana Andrei, for organising the examination and arranging the corrections.

I would like to thank everyone who has been a part of the Glasgow Cyber Safety Lab – in particular Charlie Rutherford, Kelsey Collington and Robert Harkness - your constant support has been invaluable over the past years.

Last but certainly not least, I would like to extend a massive thanks to my friends and family - without their support, my surviving with some degree of sanity throughout the time of my Ph.D would not have been possible!

The research reported in this thesis was funded through an EPSRC iCASE award (Grant number: EP/R511936/1), supported by Dstl.

Dedication

This thesis is dedicated to Lillian 'Bambam' Cook, and Caroline 'Nonna' Montaldi.

Table of Contents

List of Tables

List of Tables

List of Figures

List of Publications

The following publications are related to this thesis. The related chapters are also stated. Papers in *italics* are currently under submission.

1. **Title:** Introducing a Forensic Data Type Taxonomy for Acquirable Artefacts from Programmable Logic Controllers (Chapter 4)
   Authors: Marco Cook, Ioannis Staravou, Sarah Dimmock, and Chris W. Johnson
   At: IEEE International Conference on Cyber Security and Protection of Digital Services (2020), pp. 1–8 [4]

2. **Title:** Anomaly Diagnosis in Cyber-Physical Systems (Chapter 5)
   Authors: Marco Cook, Cory Paterson, Angelos K. Marnerides, and Dimitrios Pezaros
   At: IEEE International Conference on Communications (ICC) (2022) [5]

3. **Title:** *A Survey on Industrial Control System Digital Forensics: Challenges, Advances and Future Directions* (Chapter 2)
   Authors: Marco Cook, Angelos K. Marnerides, Chris W. Johnson, and Dimitrios Pezaros
   In: IEEE Communications Surveys and Tutorials (2023)

4. **Title:** *PLCPrint: Fingerprinting Memory Attacks in Programmable Logic Controllers* (Chapter 6)
   Authors: Marco Cook, Angelos K. Marnerides, and Dimitrios Pezaros
   In: IEEE Transactions on Information Forensics and Security (2023)

List of Abbreviations

Abbreviation	Definition
AB-CLX	Allen-Bradley ControlLogix
ADT	Artefact Data Type
ARP	Address Resolution Protocol
ATP	Advanced Persistent Threat
CCDCOE	Cooperative Cyber Defence Centre of Excellence
CIA	Confidentiality, Integrity and Availability
CIP	Common Industrial Protocol
CNI	Critical National Infrastructure
CNN	Convolutional Neural Networks
COTS	Commercial Off The Shelf
CPU	Central Processing Unit
DCS	Distributed Control Systems
DDoS	Distributed Denial of Service
DMZ	De-Militarised Zone
DoS	Denial of Service
DT	Decision Tree
DFRWS	Digital Forensic Research Workshop
ENISA	European Network and Information Security Agency
EWS	Engineering Workstation
GE-SRTP	General Electric Service Request Transfer Protocol
GNB	Gaussian Naive Bayes
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications

List of Figures

GULP	Glasgow University Liquid Purification (testbed)
HDD	Hard Disk Drive
HMI	Human-machine Interface
ICS	Industrial Control Systems
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IF	Isolation Forest
IIoT	Industrial Internet of Things
I/O	Input and Output
IoT	Internet of Things
IP	Internet Protocol
IR	Incident Response
ISA	International Society of Automation
ISO	International Organisation for Standardisation
IT	Information Technology
JTAG	Joint Test Action Group
K-NN	K-Nearest Neighbour
LOF	Local Outlier Factor
LR	Logistic Regression
MC	Mapping Condition
MRL	Master Register List
ML	Machine Learning
NATO	North Atlantic Treaty Organisation
NIS	Network and Information Security
NIST	National Institute of Standards and Technology
OCSVM	One-Class Support Vector Machine
OT	Operational Technology
PCAP	Packet Capture File
PMRM	PLC Memory Register Map
PLC	Programmable Logic Controller

RAM	Random Access Memory
RF	Random Forest
ROM	Read-only Memory
RTU	Remote Terminal Unit
S7Comm	S7 Communications Protocol
SAP	Safety, Availability, and Performance
SCADA	Supervisory Control and Data Acquisition
SD	Secure Digital
SHAP	SHapley Additive ExPlanations
SIS	Safety Instrumented Systems
SSD	Solid State Drive
SVM	Support Vector Machine
TAP	Test Access Port
TCP	Transmission Control Protocol
TRL	Technology Readiness Level
TTPs	Tactics, Techniques and Procedures
UART	Universal Asynchronous Receiver Transmitter
VSD	Variable Speed Drive

Chapter 1

Introduction

Industrial Control Systems (ICS) are a set of fundamental electronic control and computing systems that drive and maintain the reliability and safety of real physical processes that are imperative to many domains of the modern industrial world [6]. One of these areas is Critical National Infrastructure (CNI), which comprises sectors including, but not limited to, electricity generation – including the civil nuclear sector, transport, water treatment and waste management, oil and gas facilities and manufacturing plants. The central importance of ICS and their underlying components to the continuity and integrity of modern every-day life makes such systems attractive targets for cyber adversaries. Furthermore, as ICS provide a bridge between the cyber and physical domains, successful cyber-attacks could result in devastating tangible consequences [7].

Over the past 15 years, there has been a significant rise in malware deployment that is designed to target ICS [8]. In 2010, the Stuxnet malware played a key role in highlighting the extent of the cyber threat that was then faced by insecure ICS, and the potentially devastating consequences that could result [9]. Indeed, since Stuxnet, ICS cyber attacks have become more common, targeting a range of different sectors. For example, the Ukranian electrical grid [10] and Kemuri water treatment [11] incidents in 2015 and 2016, respectively, and the Triton attack, targeting safety-critical petrochemical systems in 2017 [12]. In addition, attacks inhibiting the manufacturing of critical resources, such as that in 2013 on a German steel mill [13] and more recently the LockerGoga malware attacking aluminium manufacturing in 2019, have been perpetrated [14]. As depicted in a timeline of key and well-known ICS attacks in figure 1.1, there has also been a rise in ransomware attacks targeting CNI with

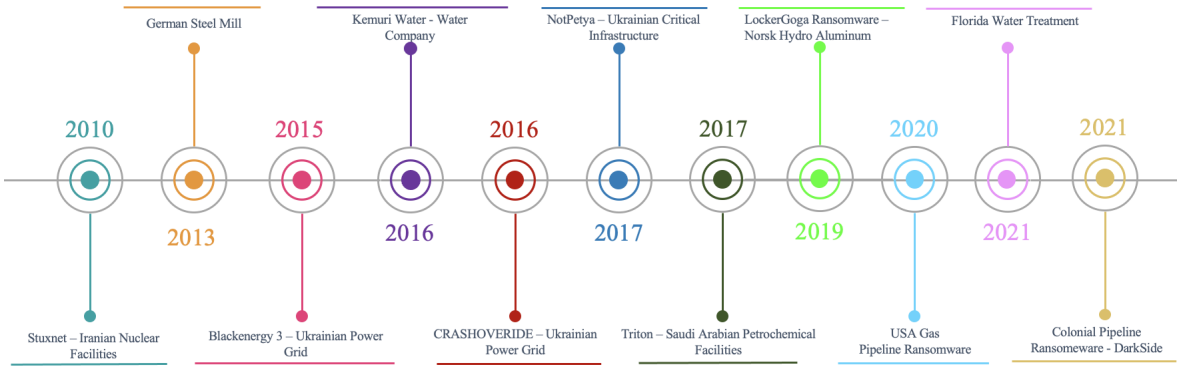


Figure 1.1: Timeline of Key Historic ICS Cyber Incidents

two recent incidents occurring in the gas and oil [15] sectors in the USA in 2020 and 2021, respectively, as well as the aforementioned LockerGoga attack.

The attacks discussed above, and other ICS cyber campaigns, were possible primarily due to the convergence and inter-connectivity of ICS components with enterprise Information Technology (IT) systems that use Internet-facing gateways and cloud technologies, further augmenting the cyber threat [16]. Advances in communications technologies and modern network architectures have resulted in ICS, which were originally air-gaped, becoming more interconnected with the external Internet-facing networks of an organisation. On the one hand, this has given engineers greater flexibility over the systems they manage and maintain, by facilitating remote control, as well as by providing the organisational management departments with higher levels of visibility over production levels and general operations. On the other hand, a greater scope for cyber threat has clearly been a consequence through new attack vectors that benefit from common types of vulnerability of ICS, which are typically designed with limited cyber security controls [17]. Furthermore, core ICS components such as Programmable Logic Controllers (PLCs), typically have limited inbuilt host or network-based incident detection and response mechanisms, with design requirements focusing more on ensuring reliability and human safety, due to the critical kinetic environments they operate in [18]. Logging capabilities and alarms are intended to assist engineers with fault and error troubleshooting rather than by providing detailed information useful in the assessment of the potential occurrence of a cyber-incident [19]. As an example, PLCs do not typically employ integrity checking when communicating with another device over a network. Therefore, anomalous commands issued to a PLC from a rogue device, such as a compromised

contractor's laptop, would be accepted and executed by the PLC due to the lack of digital signatures, as seen in the Stuxnet attack [9].

1.1 Problem Statement

Anomaly detection is a recognised field that is used to identify changes in patterns of behaviour of processes within a system that are defined as inconsistent with the normal operating conditions [20]. In other words, it uses data analysis techniques to identify events that display properties, or characteristics that do not conform with the norm and therefore are considered to be anomalous. While anomaly detection is used in many applications outside computing science, it is commonly associated with the cyber-security paradigm of intrusion detection where anomaly detection is used to identify and alert to the early stages of cyber-attacks, including those on ICS and Operational Technology (OT) environments in general.

However, a fundamental problem that arises when detecting anomalous behaviour within ICS environments is determining what an anomaly actually pertains to and how it originated, often referred to as its provenance. This is particularly challenging because different anomalies occurring in ICS can demonstrate similar empirical properties, with regards to changes in, and impacts on, the kinetic actions of the underlying physical system [21]. Therefore, a serious challenge exists in the important task of discriminating between the different types of anomaly that can occur in the behaviour of industrial components such as PLCs, purely from the empirical observation of changes in the physical process. Hence, anomaly detection in ICS is highly susceptible to failure, to false alarms and to incorrectly diagnosed incidents [22]. With an increasing number of cyber-attacks infiltrating ICS, it is clear that rapid anomaly and incident detection and response is critical to preventing extensive damage to physical environments, which could impact human safety. Critically, ICS threat actors have shown that PLCs are highly prone to sophisticated and stealthy attacks [23], which may evade existing network-based Intrusion Detection Systems (IDS) [24].

While anomaly detection within the context of cyber-security is intended to provide early warnings of an incident occurring, the area of *digital forensics* is employed as a post-incident activity. Furthermore, forensics is used to analyse evidence and provide a deeper examination pertaining to the behaviour and attribution of an incident, specifically by performing

acquisition, analysis and preservation of digital data [25]. Within the context of ICS, there are unique challenges that inhibit the use of forensic procedures on live industrial systems due to the reliability and safety requirements that are mandated for such systems [26]. Additionally, the resource-constrained design of PLCs further complicates the use of live forensic techniques to examine an evolving incident in real-time [27]. However, the demand for digital forensics and incident analysis specifically for ICS is now accelerated by the introduction of directives and standards, initiated by regulators from different countries and governing bodies, such as the European Commission's Network and Information Security (NIS) Directive [28]. The NIS Directive mandates that organisations considered to be operators of essential services must report any suspected cyber incidents to the technical authority. However, there are clear barriers preventing organisations from doing this with the limited forensic capabilities tailored for ICS, particularly regarding real-time incident analysis. Similar requirements also exist in other countries such as Japan, the USA, Australia, and throughout the European Union (EU), emphasising the breadth and scale of the challenge [29] [30].

There are a number of advantages to examining the specific signatures that represent different types of ICS anomaly. Firstly, the increasing frequency and complexity of cyber attacks targeting ICS requires that ICS operators and engineers, who will typically be the first to observe an anomaly occurring, rapidly respond in order to restore the system to a baseline state of operation. Therefore, the real-time triaging of incidents is critical in maintaining system functionality, a fundamental, and often safety requirement of many ICS use cases. Moreover, understanding the provenance of ICS data artefacts, such as network samples and memory snapshots, and how they correlate with different types of anomaly is important to the future development of next-generation intrusion detection systems for ICS. However, existing research primarily focuses purely on the detection of anomalies, rather than understanding how to profile the behaviours of contrasting anomalies. In fact, while many approaches demonstrate effective and novel detection mechanisms for ICS components, including PLCs, these typically are either suited to identify anomalies resulting from either attacks or system faults, but not both. Consequently, it is likely that such approaches would result in wrongly identifying occurrences of system faults as attacks, and vice-versa. Furthermore, as we will examine in Chapter 2, the features used to detect an ICS anomaly are also a key limitation in existing research, where studies have not provided indications on how deviations in feature values can be attributed to different types of anomalies, such as attack techniques. This is an area

we will explored in greater depth in Chapter 6.

The highly-specialised nature of ICS has resulted in the heterogeneity of ICS components, whereby many contrasting proprietary technologies have been developed by different vendors. Consequently, this deployment of components with very varied protocols, not only leads to technical challenges in cross-vendor communication, but also to the implementation of cyber-security and digital forensics. A key approach that addresses this challenge is the examination of the generalisable properties of ICS, which applies not only to the data generated by embedded devices but also to the measurement of the physical properties that pertain to the specific industrial processes. The latter is supported by the fact that ICS setups are often equipped with a large number of sensors and actuators that can provide datapoints used to model the dynamic operating behaviour of a system [31].

The detection and subsequent analysis of ICS anomalies, particularly those occurring from cyber incidents, has become increasingly challenging as adversaries employ stealthier PLC attack techniques. The current lack of commercial ICS digital forensic tools and solutions prevents cyber practitioners from rapidly assessing an attack and determining how to restore an ICS to a state of normal, safe operation [32]. As one of the key challenges when detecting PLC anomalies is assessing and determining the type of anomaly, (because different anomalies can present similar observations both physically and virtually), anomalous behaviour resulting from a system fault can produce effects that are indistinguishable from a malicious attack conducted by an advanced cyber adversary. Furthermore, as stealth is often a key objective of contemporary cyber adversaries, many PLC attack vectors can be misidentified if anomaly detection is limited to the use of detection feature sets comprising properties of network metrics [24] [33]. Consequently, attacks would go undetected, thus delaying the triaging and recovery process, leading to further disruption and damage. Therefore, in contrast with anomaly detection, the research reported in this thesis also contributes to a newer and less established field known as *anomaly diagnosis*, which while drawing on anomaly detection, focuses on the analysis of the anomalous signatures to significantly improve all aspects of incident response. While this thesis focuses on how anomaly diagnosis can be used within the field of digital forensics, as we will explore in Chapter 3, it is important to note that the thesis contributions will aim to enable the successful implementation of digital forensic strategies rather than developing digital forensic solutions themselves.

Hence, this thesis takes the following research direction:

By drawing on the characteristics and contents of PLC data artefacts, this thesis proposes a vendor-independent framework for the detection and diagnosis of PLC anomalies, and aims to inform how they can be contextualised and classified into different types to provide provenance, while also exploring and identifying which are the most informative performance evaluations of the framework.

The research in this thesis is motivated by the following overarching research questions:

1. What data artefacts can be identified and obtained from PLCs that may be used in a digital forensics context, and how are these artefacts generalisable?
2. How can PLC data artefacts be used to diagnose anomalous PLC events and differentiate between cyber attacks and system faults?
3. To what extent can the specific type of attack targeting a PLC be identified in real-time, in order to facilitate attack provenance, and aid the efficiency of incident triaging and recovery?

1.2 Research Contributions

The following key research contributions are made in this thesis:

1. The design of a generalisable anomaly diagnosis framework facilitating PLC vendor-independent anomaly detection, contextualisation, and classification, introduced in Chapter 3.
2. The development of a PLC artefact data type taxonomy in Chapter 4, which comprises data that can be extracted from artefacts stored within PLC memory structures. Particular attention is given to identifying the extent of PLC vendor independence between artefact data types.
3. The implementation and evaluation of a register state-based approach, using semi-supervised machine learning, for the detection of PLC anomalies occurring from cyber-attack or system faults, presented in Chapter 5. Evaluations are conducted using

real PLC devices from leading vendors deployed within a representative physical ICS testbed emulating a water treatment facility, the design and implementation of which is described in Chapter 3.

4. The design and evaluation of an anomaly contextualisation algorithm used to discriminate between different PLC anomalies and classify them as either cyber attack or system fault through the use of supervised machine learning in Chapter 5. In this model, we use a synthesised dataset of run-time data generated by the PLC, and identify the most suitable feature deviations for optimised anomaly contextualisation.

5. The implementation and evaluation of a memory fingerprinting attack detection approach enabled through semi-supervised learning, is proposed in Chapter 6. We build on the register state-based detection model by using compositions of generalisable PLC memory artefacts, identified in Chapter 4, which were generated by correlating the static and dynamic behaviour of PLC memory registers. We refer to this process as PLC Memory Register Mapping (PMRM).

6. The development and evaluation of an attack fingerprinting methodology introduced in Chapter 6 that uses a combination of supervised binary and multi-class classification algorithms to perform attack type and technique provenance. With this approach, deviations of PMRM fingerprint Mapping Conditions (MC), which are fabricated from cyber-attack scenarios, are used as an input source.

1.3 Structure of Thesis

The work in this thesis takes the following chapter structure:

- Chapter 2 provides a technical background on ICS technologies, and a review of the related literature. The architecture and key components of ICS are first described, moving to provide a more detailed description of PLCs. The chapter presents a summary of anomaly detection and digital forensics research, within the context of ICS.

- Chapter 3 introduces the PLC anomaly diagnosis framework underpinning this thesis. A generalised ICS threat model that grounds the preceding research and from which

our attack scenarios are derived is also presented. Additionally, this chapter introduces the Glasgow University Liquid Purification (GULP) testbed, which is used to evaluate components of the anomaly diagnosis framework.

- Chapter 4 uses the GULP testbed to establish a taxonomy of PLC artefact data types that can be used to generate feature sets for PLC anomaly detection and diagnosis. A set of PLC Artefact Data Types (ADTs) are presented that categorise individual artefacts. The chapter then discusses the generalisation of artefacts between different PLC vendors and models.

- Chapter 5 introduces the PLC anomaly contextualisation model, which using a synthesis of PLC data artefacts derived from the ADTs presented in Chapter 4, and generated during PLC run-time operation, to detect and distinguish different types of anomalous behaviour.

- Chapter 6 introduces an approach, referred to as PLCPrint, for PLC attack detection and provenance through the use of semi-supervised and supervised classification, respectively.

- Finally, Chapter 7 concludes this thesis, providing a summary, some general interpretation, the key contributions of this thesis, and recommendations for future research directions.

Chapter 2

Background Technologies and Related Work

The following Chapter consists of two sections. The first section provides the key descriptions and explanations of the ICS technologies that are central to the content of the remaining chapters of this thesis. This chapter also provides a high-level structural overview of a CNI and the Operational Technologies (OT) and enterprise systems comprised within it. The technologies found within a typical ICS are also introduced, and as this work focuses on the anomaly diagnosis of PLCs within ICS contexts, a detailed description on PLCs is provided.

The second section of this chapter provides a review of anomaly detection and digital forensics research within the context of PLCs and more generally, of ICS. We examine how ICS cyber-security research is placed within the general context of cyber defence. Moreover, a detailed review of existing ICS anomaly detection and digital forensics literature is provided, where we discuss the shortcomings and limitations of existing work, in addition to the key challenges within these two fields. Specifically, we will examine in detail how existing studies have implemented anomaly detection for ICS, with particular focus given to machine learning techniques. Furthermore, we will explore the shortcomings of existing approaches, particularly within the context of anomaly diagnosis, and how different data artefacts have been used to develop ICS-specific feature sets for detection. A review of the fundamental and unique challenges pertaining to the development and deployment of forensic solutions for ICS will also be covered. Additionally, this section introduces general terminology related to anomaly detection and digital forensics.

2.1 Industrial Control Systems (ICS)

ICS can be highly-specialised systems that employ a wide range of embedded components, network communication protocols, bespoke software and proprietary technologies. As a subsystem of Cyber-Physical Systems (CPS), ICS control industrial processes that have a physical actuation or operation to them, such as water treatment, electrical generation, manufacturing or telecommunications. The primary difference, however, between CPS and ICS is that the latter focuses on the specific instrumentation used for industrial process control, often found in CNI sectors. Conversely, CPS covers a broader spectrum that includes systems involving physical actuation and programmable elements, including but not limited to, smart building systems, medical devices and self-driving vehicles [34].

2.1.1 IT, OT and ICS

The terms ICS and Operational Technology (OT) are often used interchangeably, however here their distinction highlights the specific industrial technologies that are the primary focus throughout this thesis. Figure 2.1 decomposes typical high-level IT and OT architectures, and illustrates how they differ from ICS. Within the organisation, IT computer systems and networks are used for corporate office tasks, which are often referred to as enterprise systems,

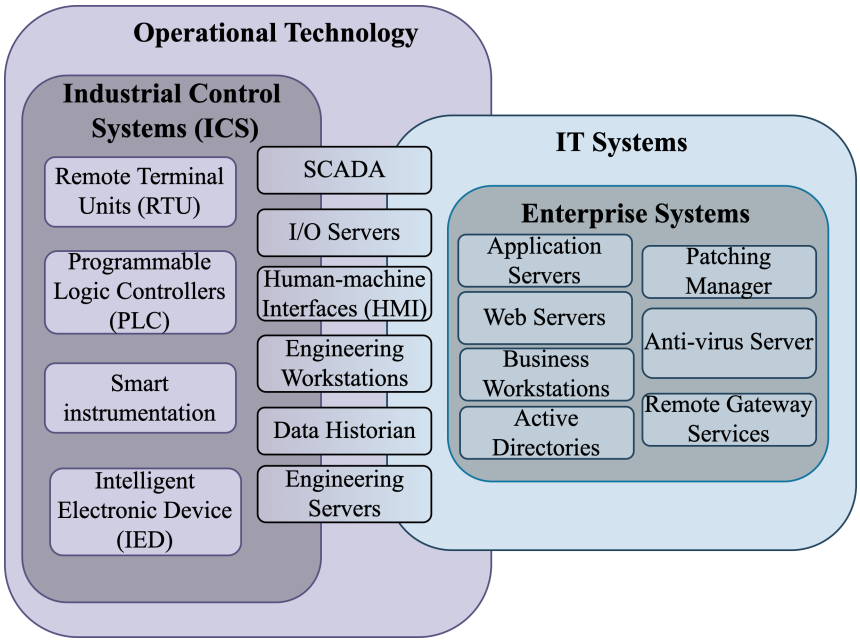


Figure 2.1: Structure of IT, OT and ICS (with example components in smaller boxes)

while an OT environment containing hardware and software is used to configure, control and manage the industrial processes. These processes are executed by the ICS, which is considered to be a subsystem of OT because, although an OT environment contains the industrial equipment and networks, it also includes some IT equipment that is commonly used for programming industrial components and monitoring aspects of the industrial processes. Additionally, these IT systems will be used to feed operational data back up to the corporate environment. In other words, a large number of IT systems are used in operational environments to support the ICS components and networks that are specifically designed to control the underlying physical process.

There are several fundamental differences between ICS and IT technologies and the properties of their respective systems. There are clear technology differences that distinguish IT from ICS as presented in figure 2.1. ICS employ a wide range of industrial technologies that often use proprietary file formats, network protocols, and firmware layers, for example. However, an increasing amount of IT technology is being used alongside ICS, including in particular, network communications, where protocols such as Transmission Control Protocol (TCP) and Internet Protocol (IP) are now used extensively within modern implementations of industrial protocols, for example Modbus-TCP and PROFINET. However, a wide range of vendor-specific technologies are still used, particularly within components such as PLCs. Furthermore, ICS manufacturers[1] often employ their own data structures, programming syntax and development environments that, while providing similar end-goal functionality, the formats used to achieve this can be very different.

The original operational requirements of ICS, such as availability and reliability, have resulted in security being a secondary consideration, while safety, availability and reliability have routinely been incorporated into the design of ICS components. Consequently, 'security by design' is not a concept that is commonly found in ICS, resulting in many ICS components, particularly legacy equipment, requiring security to be retrofitted to protect against cyber threats.

[1]The terms 'vendor' and manufacturer' are used synonymously unless otherwise stated

2.1.2 ICS Components

ICS often contain a wide range of different components and technologies, depending on the specific operational purpose. Many ICS components that are currently in operation are considered to be legacy equipment, primarily due to the long life-cycles of the products that ICS employ. These can extend up to 30 to 40 years in some cases, which is substantially longer than the three to five year-old life-cycles found in the IT domain. Conventionally, adaptations of the Purdue Model [35] are typically used as a reference architecture to describe the relationship and network links between different layers of an ICS, however it is often the case that systems do not strictly follow this model [36]. The top layers, four and five, are typically referred to as IT and enterprise systems, that comprise the servers and workstations used to conduct corporate business operations. Layer three and below represent the operational technology systems, and it is within these layers that the ICS technologies and networks reside. Figure 2.2 illustrates an example ICS architecture using the Purdue Model as a framework, and indicates what constitutes the IT, OT or ICS segments. The key ICS components that are referred to in this thesis are defined below.

Programmable logic controllers (PLCs) are specialised embedded computers that are designed to operate in highly critical and intense environments, often designed and installed with real-time operating systems (RTOS). PLCs use input/output (I/O) modules to control and manage field instrumentation actuators and sensors by performing logical computations based on an, often user-defined, application program. There are many different vendors of PLC such as Siemens, Rockwell Automation, Schneider Electric and Omron, with each vendor often implementing their own proprietary communication protocols and memory mapping into the PLC design. Some standardisation does exist for PLCs through the International Electrotechnical Commission (IEC) 61131 set of standards [37], including the programming languages used to configure them [38] and the functional safety requirements they must contain [39]. PLCs are discussed in greater depth later in Section 2.1.4.

Human-machine interfaces (HMIs) are screen-based devices used by engineers and system operators to monitor, manage and control the industrial control processes often found in lower layers of the network [40]. HMIs are typically used as a local interface to specific PLCs and other components, rather than providing an overall systemic view. Modern HMIs are often touchscreen graphical user interfaces that can be interacted with through common

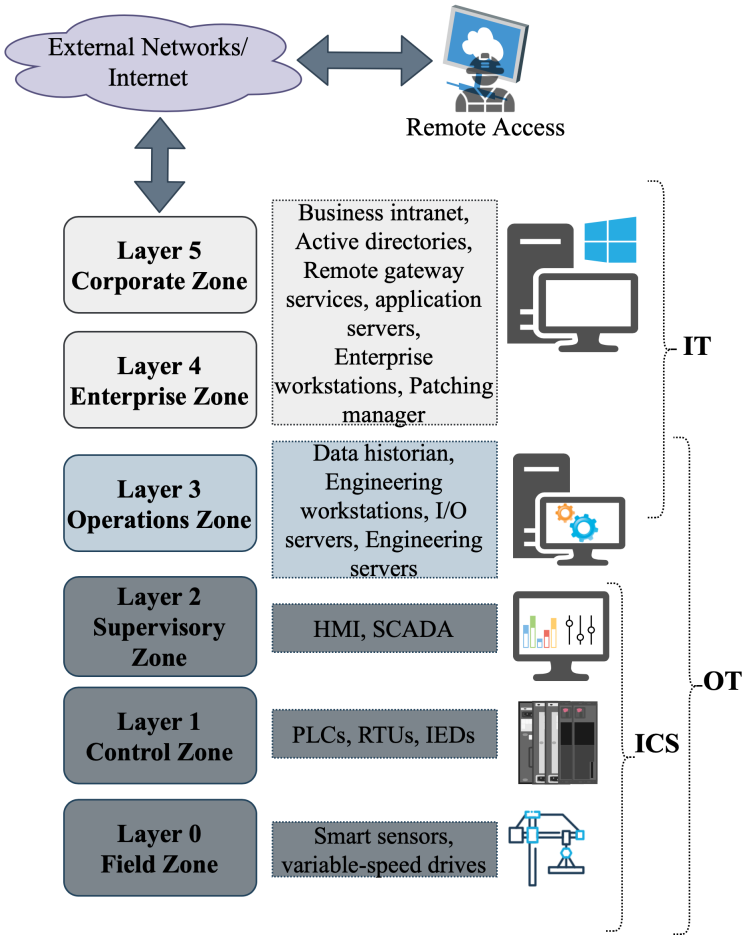


Figure 2.2: Example ICS architecture following the Purdue Model structure

computer peripherals such as a keyboard and mouse or via a touch screen. Many HMIs are dedicated embedded devices, however an alternative approach is to virtualise the HMI through software using a standard PC.

Supervisory Control and Data Acquisition (SCADA) is a computer system used to monitor and analyse real-time system data acquired from the industrial processes within the ICS. The function of SCADA is similar to an HMI, however SCADA is typically used for larger distributed and remote systems in order to provide data aggregation for such environments into a centralised control system. The data that is collected and monitored by SCADA includes but is not limited to, information on sensors and actuators (for example; motor speeds, water tank capacity levels, temperatures and pressures).

Intelligent Electronic Device (IED) is a general term that encompasses a range of devices that send and receive data from external sources, most commonly, field sensors and actuators used in the industrial processes of an ICS [40]. Examples include temperature monitors,

smart protection relays, pressure transducers and variable speed drives (VSDs). Many of these devices do contain processing units and programmable functions making them more versatile and increasing their functionality with PLCs.

Remote Terminal Unit (RTU) is similar to a PLC however RTUs are typically geographically distributed in environments away from the central control [40]. A field RTU physically connects to the sensors and field instrumentation that is remotely located. It gathers and transmits this data to a station (master) RTU, generally over wide area network (WAN) communication technologies including satellite communication networks, GPRS, and GPS. The station RTU then combines the received data with controller[2] commands to formulate an output to be executed by an actuator.

ICS field instrumentation is a general term that refers to the sensors and actuators connected within physical process of an ICS. Field sensors acquire readings of physical measures, such as temperature, pressure, or speed, and pass these onto a controller or an IED. Field actuators, such as valves, motors, and pumps, are controlled by the programmed application or configuration in a controller or an IED.

Engineering Workstation (EWS) is a standard desktop computing environment and therefore not specifically an ICS component, however they are used by engineers to commission, program and configure ICS devices such as PLCs, HMIs and IEDs. Therefore EWS often have a permanent physical network connection to components in layers 2 and 1 of the Purdue model. EWS often run a variant of the Windows operating system due to the software requirements of vendor-specific applications, such as PLC programming environments (described later in Section 2.1.4. In some instances, EWS will be connected to engineering servers that store the project and configuration files for ICS devices, such as PLC ladder logic, and HMI interface screens.

Data Historian software is used to collate data from the ICS processes into an organised database in real-time. Data historians are able to dissect a number of different ICS communication protocols, including vendor-specific communication protocols, and are generally installed on standard desktop computers, or servers that have access to the engineering networks of the ICS.

Safety-Instrumented System (SIS) is not an individual piece of technology, but rather a set

[2]Unless otherwise stated, the terms 'PLC' and 'controller' are used synonymously in this thesis.

of hardware and software used in critical processes to provide protection against health and safety consequences resulting from system malfunctions. If such a malfunction occurs, the SIS enters into a safe-state process condition, often referred to as fail-safe. The functionality of a SIS is usually determined from the output of the safety and hazard analysis performed prior to implementation.

2.1.3 ICS Network Technologies

ICS operate using a combination of serial and Internet Protocol (IP) communications, and often employ vendor-specific layers on top of these, particularly at the application, presentation and session layers of the Open Systems Interconnection (OSI) model. Consequently, certain models of a given type of device will likely use a different set of protocols to those used by other models from a different ICS vendor. Importantly, this impacts the digital forensic process, as bespoke knowledge of these protocols and how they are dissected is required to understand what data is being transported and what commands are being issued. In order for these different communication protocols to be used in tandem, and thus allowing different vendors of ICS devices to communicate with each other, ICS often utilise network gateways to translate from one protocol to another. For example, a particular ICS network might be required to translate communication traffic from the Siemens S7Communication (S7Comm) protocol to an EtherNet/IP device, commonly used by Allen-Bradley PLCs from Rockwell Automation. However, the serial communications used in ICS often have very limited security controls to prevent even the most simple attacks. Historically, industrial systems using serial-based communications were physically isolated from external networks, referred to as an air-gap, and many were even designed before the Internet emerged as a widely accessible platform. With the rise in inter-connectivity between the ICS network layers and the corporate IT layers through IP and Ethernet-based communications, the physical air-gap became less applicable and, therefore, the vulnerabilities of the serial protocols were suddenly exposed and internet-facing [16]. This is particularly the case for legacy communications, but it is a key challenge in ICS cyber security that extends to other elements of the system, not just network communications. A recent survey paper published in [41] provides a comprehensive and detailed discussion of ICS network protocols and their security properties.

2.1.4 PLC Architecture

PLCs are a fundamental component of ICS with responsibility for controlling the underlying physical processes of the system. Control commands are issued to actuators, and measurement readings are requested from sensors and sent to the PLC, as illustrated in figure 2.3. PLCs typically have limited memory storage capacity, particularly compared with IT systems with the amount being dependent on the selected CPU model of the PLC. For example, the 315-2 PN/DP CPU model for the Siemens S7-300 PLC has 384KB of integrated read-only memory (ROM). The non-volatile memory storage can often be expanded through flash-based memory cards, which can be used to store controller meta-data and diagnostics information. Different PLC models have varying types and sizes of memory containing a range of types of data, some that is user-created and others that implemented by the vendor and so cannot be altered by end-users. Some of the key components of the PLC architecture, as depicted in figure 2.3 are presented below.

Network Interfaces - Commonly referred to as communications processors, network interfaces are used to establish networks with instrumentation such as HMIs or remote I/O. PLC CPUs will typically have an integrated communications processor, such as an Ethernet based Modbus TCP or Profibus for serial networks, however most PLCs can have additional communications processors added on, to extend their network diversity.

Application Program - The PLC executes a user-defined program that consists of control logic, hereon referred to as the *application code*[3], which is typically written in a standardised PLC programming language as defined in international standard IEC 61131-3 [38]. Ladder logic is one example, however there are other languages such as Structured Text and Instruction List. The PLC application program as a whole typically comprises a number of function blocks that contain segments of control logic and sets of PLC variables. Furthermore, the application code itself is built using a range of logic objects, timers, counters and input/output (I/O) addresses written to perform a specific task, and is stored within the non-volatile memory of the PLC. This is usually electrically erasable programmable read-only memory (EEPROM).

Figure 2.4 illustrates an example of PLC application code programmed in ladder logic, in

[3]Unless stated otherwise, the terms 'control logic' and 'application code' are used interchangeably within this thesis.

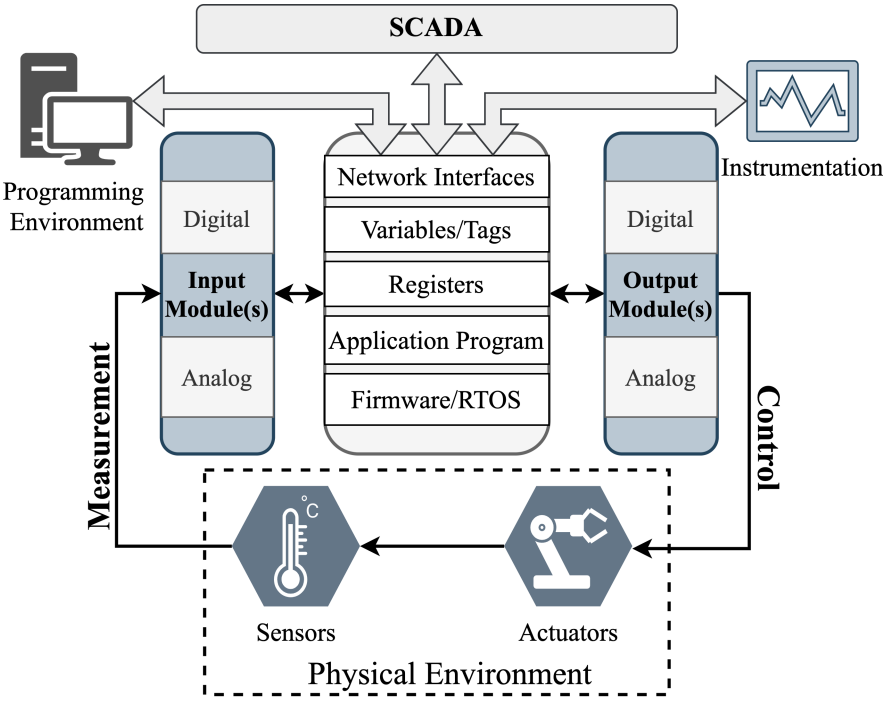


Figure 2.3: Architecture of PLC demonstrating control flow of actuators and sensors with hardware and software components

this case specifically for a Siemens S7 PLC model. Program objects are used to define the logic executed by the PLC. The example shown in the red square in figure 2.4 represents a normally open contact, which requires energising to activate it and close the contact such as through a mechanical switch. Each program object is then assigned a register address so that it can be stored and accessed from PLC memory. The register addresses also allow other ICS devices, such as HMIs and other PLCs to access specific values to either display to an operator or use as a dependency in a separate program. Different PLC vendors use contrasting addressing formats and syntax, although it is conventional to use 'I' for input registers and 'Q' for output registers. For example, in Figure 2.4 we can see that the registers

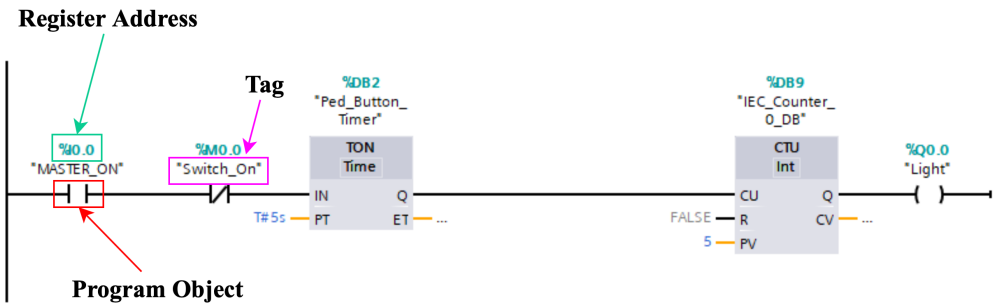


Figure 2.4: Example of PLC Ladder Logic code

'%I0.0' and '%Q0.0' are referenced, or tagged, as 'MASTER_ON' and 'Light', respectively. Tags are assigned to registers as a way of naming the variable, as illustrated in the purple box in figure 2.4, so that it can be more easily referenced to within the application code. The numerical values accompanying each register address letter (i.e. I, Q, etc.) represent the register addresses, for example *I0.0* and *Q0.0*, where the most significant value represents the byte and the least significant is the bit (0-7). Although PLC application code is generally visually and syntactically similar across different vendors, the binary representation of the code is in a proprietary format specific to the vendor, and even to the exact model of PLC.

PLC Programming Environment - In order to configure, program and manage the PLC, a specific programming environment, which is vendor-specific for most major PLC brands, is used on the EWS. For example, Siemens PLCs use Step 7 or Totally Integrated Automation (TIA) Portal, whereas Rockwell Allen-Bradley PLCs are programmed using Studio 5000. The network interface of the PLC is used to download any configurations or application program elements to the PLC, usually through an IP-based protocol, which again, is often vendor-specific. An upload function is also typically available to users which transfers the existing application program and configuration from the memory of the PLC and into the PLC programming environment, which is often used for code debugging.

PLC Project - The area in which PLCs are programmed and configured within the programming environment is referred to as the PLC's project. This is created by the user and is the offline version of the user program consisting of the PLC application code, hardware configuration and other information.

Input/Output Modules - PLCs are typically modular with multiple hardware options, including CPU model and network communications processors. Input and output (I/O) modules are used to control actuators and read sensor measurements either through digital (discrete) or analog (continuous) signals, which are then communicated to the CPU and memory structures for processing. Digital I/O modules, such as switches, solenoid valves, and capacitive sensors produce or are controlled through binary values (0 or 1), whereas analog I/O modules use voltage (0-10VDC) or current ranges (4-20mA) as signals that are scaled and normalised within the PLC application code to provide readings from sensors, or control commands for actuators.

Registers - Registers are memory locations within the PLC that are used by variables in the

application code to access memory.

Variables and Tags - The application code comprises a number of variables, which are containers that hold information and are often associated with the physical sensors and actuators through the use of PLC registers. Variables, often referred to as tags, consist of a name and data type, including but not limited to Boolean, float or String.

Firmware/RTOS - PLCs do not utilise the operating systems (OS) found in IT environments that comprise extensive functionality, such as Windows, but rather a firmware layer and often a Real-Time Operating System (RTOS) that controls all operations and tasks through the central processing unit (CPU). Firmware and RTOS environments are proprietary and predefined by the vendor with minimal customisation from the end-user, except regarding the version of firmware that is installed on the PLC.

2.2 Anomaly Detection

The life-cycle of cyber defence can be abstracted into five fundamental elements, defined in the Cyber-Security Framework (CSF) for Critical Infrastructure proposed by the National Institute for Standards and Technology (NIST) [42]. Namely, these framework elements are Identify, Protect, Detect, Respond, and Recover, as illustrated in the lower part of figure 2.5. Specifically, the NIST CSF is widely adopted and accepted as a leading framework for the implementation of cyber-security principles throughout the world, including the European Union (EU) and United States of America (USA) [30] [43]. Other country-specific standards, such as the Cyber Assessment Framework (CAF) developed by the National Cyber Security

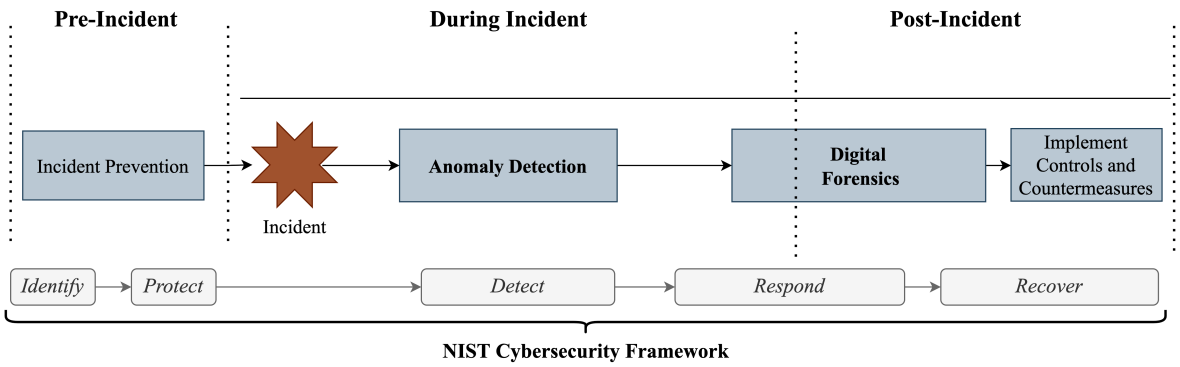


Figure 2.5: Core cyber security areas mapped onto five NIST Cyber Security Framework functions - bold areas represent themes discussed in this thesis

Center in the United Kingdom (U.K), provide similar structures for industries, governments and research to use as a common understanding.

Core high-level cyber-security research fields that focus on the prevention, detection and subsequent analysis of incidents can be categorised over the five NIST CSF functions to provide a representation of how existing literature addresses this framework. *Incident prevention* focuses on the proactive implementation of cyber security controls, policies, training and risk assessment, all linked under a similar objective of threat mitigation and preventing an incident from occurring. Intrusion Protection Systems (IPS) are an example of a technical control that would aim to achieve this goal through real-time monitoring. However, it is inevitable that not all incidents can be prevented and it is not practical to fully mitigate the cyber risk, particularly due to the extensive threat targeting OT environments, including ICS. Therefore, it is imperative that incidents are detected and analysed so that an organisation can restore the affected systems to a known good state, and proactively work towards implementing new controls and improvements to mitigate similar future incidents from occurring. This process starts with *anomaly detection*, a core theme of this thesis.

Across a wide variety of applications and domains, the objective of anomaly detection is generally to identify patterns within a given dataset that do not conform to a baseline of expected behaviour [44]. The subsequent discrepancies identified within these patterns are then referred to as anomalies or outliers, which can provide critical information about an abnormal, or anomalous event. Within cyber-security, anomaly detection has often be used to aide Intrusion Detection Systems (IDS) and other security controls used to identify malicious behaviour. *Intrusion detection* is a subcategory of anomaly detection that attempts to identify events of unauthorised activity within a system or network, specifically a cyber threat [45].

2.2.1 Detection Strategies

There are different approaches that can be adopted to perform ICS anomaly detection. While there are no formal taxonomies or groupings of approaches for anomaly detection, within the context of security we can generally separate anomaly detection techniques into two categories: 1) *Knowledge-based*, and 2) *Behaviour-based* [46].

Knowledge-based anomaly detection techniques, also referred to as signature detection

techniques, use existing datasets to identify the profile characteristics of known attacks to detect anomalies [46]. While these methods are effective in identifying commonly reported attack vectors, they are less suitable for detecting unpredictable anomalous events, including zero-day vulnerabilities, which are common in ICS attack vectors [47]. There are additional challenges with generalising attack signatures across domains, depending on how the detection algorithm is designed, as it will only flag up anomalies that match a specific pattern [46]. The signature database, or attack dictionary, must therefore be consistently updated when new attacks emerge in order to provide continuous detection, resulting in significant overheads. Within the context of ICS, attack signatures are very rare and often differ between component vendors and network protocols, further obfuscating barriers to the generalisation of algorithms. Other knowledge-based methods such as state estimation and rule-based detection, have also been explored to address normal state estimation tasks in ICS [48]. However, the major disadvantage of these approaches is that they require expert and domain-specific knowledge to implement them.

Conversely, **behaviour-based** anomaly detection strategies examine trends in system runtime activity to identify deviations from normal behaviour [46]. Detection systems that adopt this type of strategy can dynamically adapt to new attack vectors and identify anomalous events since it is typically the behaviour of the protected system that is defined as the baseline instead of the attack itself. In other words, behaviour-based anomaly detection models are particularly advantageous in security as they do not look for specific patterns, eliminating the requirement for a database comprising signatures of anomalous events, including attack vectors. Machine learning (ML) is used to optimise behavioural anomaly detection approaches and enable automated and often real-time classification or clustering of anomalies. ML techniques can be generally categorized under four paradigms; *supervised*, *semi-supervised*, *unsupervised*, and *reinforcement learning*, although semi-supervised is often considered a sub-branch of both supervised and unsupervised learning [49].

Supervised techniques involve the use of labeled data points, which can represent different categories related to the classes we aim to discriminate between, referred to as classification. There are different types of classification approaches. *Binary classification* involves two possible classes that an given data point can be classed as, for example normal or anomaly. Conversely, *multi-class classification* considers more than two possible classes, such as pre-

dicting a type of flower or animal. Multi-class classification can be performed through a 'One vs. One' (OvO) or 'One vs. Rest' (OvR) strategy. In OvR tasks, the multi-class dataset is split into multiple binary classification problems, where a classifier is then trained on each binary classification problem, with the samples of the 'One' class labeled positive and all other samples ('Rest') given negative labels. OvO is similar to OvR, however through the OvO method, the multi-class dataset is split into one dataset for each class and computed against every other class in the dataset.

A fully labeled training data set is used in supervised learning where baseline behaviour is distinguished from anomalous behaviour through positive and negative samples [50]. Labelling anomalous ICS data can be challenging due to the difficulty in acquiring realistic ICS attack data, particularly from live safety-critical ICS. In semi-supervised techniques, we utilise a small amount of labeled data along with a majority of unlabeled data points during training. Within the context of novelty detection, semi-supervised approaches often take the position that the training data only has labels for the normal or baseline instances, which makes such ML algorithms more widely applicable than supervised techniques [50]. Conversely, unsupervised learning approaches do not require any data labelling, however they often require datasets to contain a much larger ratio of normal instances to be present compared with supervised and semi-supervised methods. Unsupervised learning typically involves clustering and dimentionality reduction tasks within large feature sets, specifically using algorithms such as Principal Component Analysis (PCA). As unsupervised approaches do not rely on pre-existing class labels, they are particularly effective at detecting unknown or new attacks that have not been provided in the training or testing phases [51]. However, such approaches are highly sensitive to noise within datasets and therefore have been shown to generate higher rates of false positive and false negatives [51].

Deep learning is an increasingly popular area of ML, suitable for complex classification tasks. It uses multiple levels of representation, starting with raw input data and transforming them into layers of greater abstraction. *Deep learning* approaches can be either supervised or unsupervised [52]. By using multiple layers instead of a single layer, as found in many conventional ML methods, the elements of the raw input data that provide the stronger features for discriminating different classes can be emphasised, while those elements that are weaker can be omitted. Deep learning has many different applications, however those in

the areas of image processing, computer vision, and medical imaging are particularly common [53]. Reinforcement learning offers a different approach where the goal is to enable an agent to learn an optimal set of conditions called policies based on environmental factors that maximise the long-term reward function [54]. Both deep learning and reinforcement learning require a large amount of training and test data to reduce the ratio of false positives occurring, as well as demanding significant computational power, and therefore can be challenging to implement within the context of ICS [48].

Regardless of the ML approach that is selected, hyper-parameter tuning can be employed to optimise parameters of the chosen ML model to better suit the problem area that it is addressing. The hyper-parameter tuning process is performed before the training stage since the parameters themselves define the architecture of the model [55]. Random search and grid search are two of the most commonly used techniques for hyper-parameter tuning. Specifically, grid search performs an exhaustive search for the optimal configuration of a given pre-defined set of hyper-parameters that are to be evaluated. Therefore, the performance of the grid search is impacted by the number of features a dataset contains, and so is less suitable for ML problems that use a high number of features [55]. Conversely, random search selects hyper-parameter configurations at random from the pre-defined set of test parameters. While the random search approach can computationally outperform grid search, particularly when we consider high-dimentionality feature set, however random search has been demonstrated to provide lower accuracy for optimised hyper-parameter tuning [56].

2.2.2 ML for ICS Anomaly Detection

Within the context of ICS, all three types of ML behaviour-based anomaly detection have been implemented in previous research. For example, novelty detection is one type of semi-supervised anomaly detection that examines new observation samples to assess whether they fit within the pattern of an existing dataset or whether they are outliers [57]. The work reported in [18] puts forward a semi-supervised model for the detection of anomalous PLC behaviour through the use of One-Class Support Vector Machine (OCSVM), a single-class novelty detection algorithm that can be used in many different novelty detection applications. OCSVM [58] is a widely adopted novelty detection technique, and is used to train anomaly detection models with datasets that comprise only one class of samples. The ap-

proach is similar to that of Support Vector Machine (SVM) [59], however instead of finding a hyper-plane to separate two classes in vector space, OCSVM attempts to fit a hyper-sphere around normal data and classifies anything outwith the hyper-sphere as abnormal. In [18], the authors demonstrate that Polynomial kernels for the OCSVM algorithm are particularly high-performing, achieving detection scores (F1) of 0.9 through three test datasets comprising PLC data simulating a road traffic light crossing. In general, the authors note that semi-supervised approaches are particularly advantageous for ICS and PLCs as they do not require extensive data labelling for either normal or anomalous samples.

The work in [60] proposes a framework for attack detection utilising a dual Isolation Forest (IF) method for water treatment environments. IF are a commonly used unsupervised learning technique [61], developed as an adaptation of the Random Forests (RF) supervised learning algorithm [62]. The key difference between IF and RF is that the latter aims to learn trends in labelled normal data and classify abnormal trends from this baseline, whereas IF isolates anomalies by analysing attributes of a given data point. IF utilises principles of decision tree algorithms, isolating outliers by randomly selecting a feature and subsequently partitioning random split values on that given feature. The partitioning can be represented in a tree structure, with anomalies producing smaller paths than others. The framework reported in [60] employs IF in a semi-supervised manner by using labelled normal data and plots the clusters of anomalous data through normalisation and Principle Component Analysis (PCA) to identify outliers. While the resulting detection F1 scores are an improvement on alternative models in the literature, the authors argue that a key benefit of the dual IF method is the low computational complexity required to perform data training and detection.

Supervised learning approaches are often used to address classification problems, and are particularly beneficial as they enable greater control over the definition of the training algorithms, compared to unsupervised learning. Within the context of ICS anomaly detection, an important study explored how the behaviour of PLCs can be fingerprinted through power signatures that are unique to the program a given PLC is executing [63]. The authors note that as ICS training datasets can be very large, with an extensive number of features, the RF approach is particularly suited for classification. RFs build on the Decision Tree (DT) algorithm, which uses a non-parametric divide and conquer approach to recursively partition data into sections based on the split that best discriminate between different classes [64]. A

RF classifier is composed of numerous decision trees, where the output of the classifier is the class that was predicted the most by the individual decision trees, using an aggregated approach. The method proposed in [63] demonstrates that RF classifiers were particularly susceptible to noise within datasets, which can be common with ICS especially if the features related to physical quantities and measures such as voltage, temperature or pressure, are used. The authors address this limitation by using rolling averages and a Butterworth Filter to remove frequency ripples within their feature metrics.

A comparative approach reported in [65] demonstrates the use of two different supervised techniques, specifically Decision Trees and SVM, for PLC anomaly detection using PLC register states as a feature set. The authors justify the use of Decision Trees as the feature set is small and comprises a few PLC register values, however, it is expected that the number of registers would be much greater in real-world ICS, and the use of Decision Trees for this reason may limit the scalibility of the proposed model. Conversely, SVM are employed because they benefit from smaller training datasets, which can be advantageous in ICS where acquiring large amounts of training data can be logistically problematic if the data acquisition approach is active and could degrade system reliability. In the context of [65], both supervised algorithms are reported to perform similarly, however the authors only utilise the accuracy metric for evaluation. Thus, there is limited information regarding different types of performance for the proposed model. While classification through supervised learning techniques provide particular benefits for anomaly detection, there are limitations when using these in the context of ICS. Data labeling can induce significant overheads within data pre-processing stages and it can often be logistically challenging to accumulate sufficient amounts of labeled test data points that contain anomalies. Furthermore, the large number of different types of attack vectors and techniques that can be used to target an ICS and the underlying components, leads to challenges of having sufficient samples of test data points that accurately represent the array of impacts these techniques could have on a feature set.

An additional study reported in [66] proposes a graphical model-based anomaly detection approach called TABOR, through which behaviour profiling of normal ICS operations is conducted. Specifically, the authors explore how one-class supervised learning can be used to generate a root-cause analysis of an anomaly by localising the particular sensor or actuator that is targeted in the attack. TABOR uses timed Automata learning and Bayesian network

inference to model interactions between field-devices such as sensors and actuators. Thus, the attack detection approach can be generalised to multiple different types of ICS deployment. While the study provides diagnosis at layer zero of the Purdue model, there are still general limitations with addressing anomaly detection or diagnosis challenges specifically for PLCs, such as with device triaging and recovery.

Unsupervised learning offers an alternative approach without the use of class labels, and is primarily achieved through clustering methods. Work proposed by [67] uses the OCSVM algorithm in an unsupervised manner to counteract the challenges of requiring labelled data for training and testing. Consequently, this resulted in high detection scores, and particularly in a low false positive rate (FPR), however a large amount of training data was required in order for the model to accurately identify outliers from normal operational behaviour. Therefore, this approach demonstrates the importance of careful consideration when planning to use ML algorithmic design for ICS anomaly detection.

One method of deep learning that has been used for ICS anomaly detection is that of neural networks [68] [69] [70], which involve an architecture that includes node layers, containing an input layer, one or more hidden layers, and an output layer. With enough training data and parameter tuning, neural networks can provide solutions to rapid classification and clustering that may take much longer with conventional ML methods [71]. One study presented an unsupervised approach in [69] and demonstrated that deep neural networks (DNN) can outperform classification algorithms that do not use deep learning for anomaly detection, the example used was OCSVM. However, the authors note that the computational overheads required for initially training DNNs can out-weight the gains in detection performance, with their OCSVM training only taking 30 minutes in comparison to the DNN's two weeks [69]. Furthermore, the average detection time for DNNs was significantly longer during attack scenarios, resulting in limited use in the production of a rapid, real-time response to a cyber attack, which could be critical in preventing cascading impacts to the ICS, including financial losses and degradation of safety. An additional study that explores the use of deep learning in ICS anomaly detection is described in [72] where the authors use a Restricted Boltzmann Machine (RBM) to model spatio-temporal data patterns and behaviour within the ICS, specifically for fault detection. RBMs learn probability distribution functions by calculating weights and biases for different feature configurations, where normal system operations are

identified with high probability.

The potential value of combining different types of ML algorithms has also been examined for ICS anomaly detection. One such study [73] presents a two-phase behavioural fingerprinting approach that exploits the low latency and deterministic properties of ICS networks to generate an overarching system fingerprint. The authors utilise both supervised and unsupervised learning techniques to evaluate the performance of the fingerprinting approach. They first use a supervised multi-nominal naive Bayes classifier and then for the unsupervised approach, which was employed to address limitations of legacy devices running over networks where there is limited pre-existing architecture knowledge. Specifically, the authors used Gaussian mixture model (GMM) clustering. The latter of these two approaches is particularly beneficial to ICS networks where there are often high amounts of legacy equipment operating due to the adoption of long equipment life cycles.

In this section, we have examined the use of ML for ICS anomaly detection and demonstrated that is a popular and effective method. Several approaches have utilised supervised or semi-supervised approaches with labeled datasets, as presented in table 2.1, demonstrating that despite the challenges with obtaining anomalous data, particularly in ICS environments, high accuracy and performance measurements are often achieved. In this thesis, we explore a combination of different ML techniques for the various stages of anomaly diagnosis presented in the subsequent chapters. Specifically, supervised learning will be used to classify different types of anomaly and attack targeting PLCs, primarily due to the accuracy benefits compared with unsupervised approaches. From the comparison table, we also see that both simulation and physical ICS testbed are frequently used. As we will see later in Chapter 3, a physical testbed will be used to evaluate the methodology and techniques proposed in this thesis. Although physical testbeds are more costly and can involve greater complexities, they are also typically more representative of real ICS environments, and therefore preferable for research evaluations. We discuss this in greater detail later in Section 2.2.4.

2.2.3 Anomaly Detection Objectives

Anomalies within ICS can be broadly categorised as being caused by either *attacks* or *faults* [48]. The high-level difference between these two types of anomaly is that attacks are conducted with malevolent motivations and are thus malicious. Conversely, faults occur

Table 2.1: Comparison and summary of machine learning and evaluation ICS anomaly detection studies **Key:** ●= Coverage, ❍= No Coverage. For Detection Method: *Sup* = Supervised Learning, *Semi-Sup* = Semi-Supervised Learning, *Unsup* = Unsupervised Learning. For Evaluation: *Sim* = Simulation, *Phys* = Physical testbed/components

Approach	Detection Method				Evaluation	
	Sup	Semi-Sup	Unsup	Other	Sim	Phys
Formby *et al* (2016) [73]	●	❍	❍	●	●	❍
Liu *et al* [72]	❍	❍	●	❍	●	❍
Feng *et al* [74]	●	❍	❍	❍	❍	●
Inoue *et al* [69]	❍	❍	●	❍	●	❍
Xiao *et al* [68]	❍	❍	❍	●	●	❍
Yau *et al* (2017) [18]	❍	●	❍	❍	❍	●
Yau & Chow (2017) [65]	●	❍	❍	❍	❍	●
Lin *et al* [66]	●	❍	❍	❍	❍	●
Chan *et al* (2019) [67]	❍	❍	●	❍	●	❍
Stockman *et al* (2019) [63]	●	❍	❍	❍	❍	●
Elnour *et al* (2020) [60]	❍	●	❍	❍	❍	●
Formby *et al* (2020) [75]	❍	❍	❍	●	●	❍
Abdelaty *et al* (2021) [70]	❍	❍	❍	●	❍	●

more naturally as a result of system failures or software errors, and are typically unexpected. Furthermore, the cause of a fault, in this context, is often due to the complexity of the systems and physical processes that ICS components enable. Environmental factors can also play an important role in the causation of faults as many ICS devices operate in harsh and high-intensity conditions that can make mechanical components vulnerable to failures.

A subset of the ICS anomaly detection literature has been dedicated to the identification of faults, although much of this literature is grounded in mechanical engineering domains instead of Cyber Security, and does not consider how to identify faults from other sources of failure and other forms of anomalous behaviour. Sensors and actuators are a common sources of fault occurrence and can cause discrepancies in control signal behaviour, leading some studies to propose novel techniques involving predictive modelling of a given ICS controlled by PLCs [76] [77]. The work presented in [76] introduces the PLC Log-data Analysis Tool (PLAT), which uses state-based modelling of changes in sensor and actuator data logged through a historian system. PLAT uses similar data sources to those used by PLC anomaly detection studies that focus on identifying attacks instead of faults, highlighting the versatility of using PLC registers and variables for different anomaly detection objectives.

A similar study proposes the Fault and Behavior Monitoring Tool for PLC (FBMTP), which builds on the PLAT approach proposed by the same authors [77]. The study uses deterministic finite-state automaton (DFA) and time-out functions to generate states of control signal behaviours by acquiring process log data, with the aim of isolating faults, in addition to detecting them. An additional study described in [78] proposes the use of Bayesian networks to model the causal and temporal correlations between physical variables and their digital representations within the ICS, using only unlabeled data points. The key novelty with this work is that the authors extend the use of anomaly detection for root-cause isolation, specifically identifying a cyber-attack or a physical fault occurrence through a Markov Blanket that formulates the the scope of the root-cause by examining parent and child nodes that are mapped to the original anomalous node. Further work has explored the use of Evidential Network Modeling for detecting both cyber-attacks and erroneous behaviour resulting from system malfunctions and software errors in CPS environments [21]. Evidential Network Modelling uses a graph structure approach to describe the relationships between variables within multivariate systems, and uses graph nodes and edges to encode information about these variables and the dependencies between them. Through their approach, the authors demonstrate high true positive rates, particularly when compared with classical Bayesian Network approaches, although also reached moderate false positive rates, in some cases as high as 20% for detecting erroneous behaviour. While this study enables the detection of different types of anomaly that can occur in CPS, there is no subsequent diagnosis to elicit the specific type of anomalous behaviour that is occurring, and therefore has limited applications to incident triaging.

2.2.4 Evaluation Principles

Testbed Environments

Testbeds are typically scaled-down versions of what they are models of, yet representative functional environments where experiments can be conducted to generate datasets without the increased safety risk of damaging real ICS. Using physical testbeds with real components is significantly beneficial to the research evaluation process as it reflects the intrinsic principles of real systems [48]. In particular, this not only enables the evaluation of key the-

ory and concepts, but also of computational performance. An alternative to using physical representations is to simulate real systems through software and virtualisation. While this approach is often cheaper and quicker to implement, the extent to which simulated environments can capture problems that occur in real systems is limited. A realistic data-generation environment is essential to achieving relevant and applicable results for ICS cyber-security research. Regardless of whether physical or simulated environments are used, ICS testbeds are increasingly being implemented by academia, Governments, and private industry for research and proof-of-concept evaluation due to the safety risks and logistical challenges that would be encountered with similar evaluation in real ICS environments [79].

Implementing representative large-scale ICS testbed facilities that can be used to evaluate research outputs and proofs of concept, brings several unique challenges with it. Large upfront costs for ICS equipment and software can deter organisations, such as universities, from implementing physical testbeds, and encourage reliance on software simulation instead [80]. Where organisations can afford to purchase real ICS equipment, and have personnel with the skills required to implement a physical testbed, there are additional challenges regarding scalability and safety. Many testbeds focus on particular ICS vendors, such as Siemens or Schneider Electric, and extending testbeds to include multiple vendors of both ICS hardware and software can be very costly indeed [79]. Regarding safety considerations, many real-world physical processes, such as those found in nuclear power facilities and manufacturing, cannot be easily reproduced within laboratory facilities, reducing the opportunity for validating potential real-world incident response scenarios and forensic actions/processes [19]. A more comprehensive discussion on ICS testbeds, is offered by literature that provides detailed reviews of the existing capabilities in this area [81] [79].

Table 2.2: Example Confusion Matrix

		Predicted Condition	
		Positive	Negative
Actual Condition	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Detection Evaluation Metrics

There are a number of metrics that can be used to evaluate anomaly detection models, with each metric providing information on particular performance properties. Generally, these metrics conduct statistical hypothesis testing which evaluate type 1 errors, where an actually true null hypothesis H_0 case is mistakenly rejected, and type 2 errors where a H_0 case that is actually false is mistakenly not rejected. These errors are more commonly referred to as *false negative* (FP) and *false positive* (FN), respectively. A Confusion Matrix can be used to visualise of the performance of a classification algorithm within the context of the H_0, illustrated in table 2.2.

From the confusion matrix, we can perform various calculations to ascertain the performance of a particular test against a defined H_0. The *precision* is calculated as the ratio of correctly classified positive events to the total number of positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall on the other hand is calculated as the ratio of correctly classified anomalous events to all observations in the actual class. Recall is often referred to as the *true positive rate* (TPR).

$$Recall(TPR) = \frac{TP}{TP + FN} \tag{2.2}$$

The *false positive rate* (FPR), which measures the probability of falsely rejecting the H_0 for a given test. The ***FPR*** is calculated as the ratio between the number of FP and the total number of real negative events, regardless of what classification they have been assigned.

$$FPR = \frac{FP}{FP + TN} \tag{2.3}$$

Accuracy is another statistical metric used to measure the performance of how well a classification test correctly identifies or omits a condition. In other words, accuracy is the proportion of correct predictions ($TP + TN$) within the set containing the total number of cases examined within the test.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.4}$$

One of the most commonly used evaluation metrics for anomaly detection is the *F-score*, which measures the accuracy of a particular classification test. Specifically, the F_1-score, which is the most commonly used as it provides a balanced weighting, is calculated as the harmonic mean of both the precision and recall values:

$$F_1 = 2 * \frac{(precision * recall)}{(precision + recall)} \tag{2.5}$$

A more generic score referred to as $F\beta$ enables additional weights to be added for cases where valuing one of precision or recall more than the other is important. In other words, the β parameter, which is defined as a positive real number $\beta = \mathbb{R}_{\geq 0}$, determines the weight of recall in the combined score. Two commonly used values for β are $\beta = 2$ which weighs recall higher than precision, and $\beta = 0.5$, which weighs recall lower than precision.

$$F_\beta = \frac{(1 + \beta^2) * (precision * recall)}{(\beta^2 * precision + recall)} \tag{2.6}$$

When dealing with multi-class classification problems, as described in Section 2.2.1, the F1 measure should be adapted to involve all of the classes that are being evaluated. The *Macro F1-Score* is one commonly used metric [82]. The measurement provides the same weight to all classes so that there are no distinctions between classes with different population sizes. Therefore, it is particularly well suited to the evaluation of multi-class problems that have balanced classes [82]. To calculate the macro F1, we firstly need to obtain the macro-Precision and macro-Recall, which are themselves computed as the average precision and average recall for each predicted class for each actual class. Firstly, the precision and recall for each individual class κ are calculated using the formulas for binary classification problems as presented in equation (2.1) and equation (2.2), respectively. Subsequently, we thus calculate the macro average precision and recall as the arithmetic mean of the metrics for each class κ as follows:

$$MacroAveragePrecision = \frac{\sum^{K} \kappa_{=1} Precision_\kappa}{K} \tag{2.7}$$

$$MacroAverageRecall = \frac{\sum^{K} \kappa_{=1} Recall_{\kappa}}{K} \tag{2.8}$$

Hence, to produce the macro F1-score, we calculate the harmonic mean of the MacroAveragePrecision and MacroAverageRecall as thus:

$$MacroF1Score = 2 * \frac{MacroAveragePrecisionMacroAverageRecall}{MacroAveragePrecision^{-1} + MacroAverageRecall^{-1}} \tag{2.9}$$

We can examine how effective a classification model is at distinguishing between different classes of data by measuring the *Receiver Operator Characteristic* (ROC) and *Area Under Curve* (AUC) [1]. ROC is a probability curve that plots the TPR against FPR at various threshold values. Conversely, the AUC is used as a summary of the ROC curve and measures the ability of a classifier to distinguish between different classes of data, for example originating from different anomalies. Higher AUC scores indicate better performance of the model at discriminating between positive and negative classes. AUC is measured between 0 and 1.0 where 1.0 indicates that all points of data were correctly identified as either positive or negative classes, whereas a score of 0 represents a classifier where no data points were correctly identified. An AUC score of 0.5 indicates that the classifier cannot discriminate between the classes and hence is predicting a random class. In this case, the ROC curve is referred to as 'no skill'. We can denote high performance of a classifier when $AUC > 0.5$, indicating that there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. In other words, a greater proportion of TP and TN were correctly detected, than were FP and FN. Figure 2.6 depicts an example ROC-AUC graph illustrating the ROC for different classifier performance levels.

2.3 ICS Data Artefacts

Understanding the available sources of data and potential information that can be acquired from such sources, is critical in any anomaly detection algorithm or digital forensics investigation. *Artefacts* are objects of data that are typically used for investigative purposes, such as

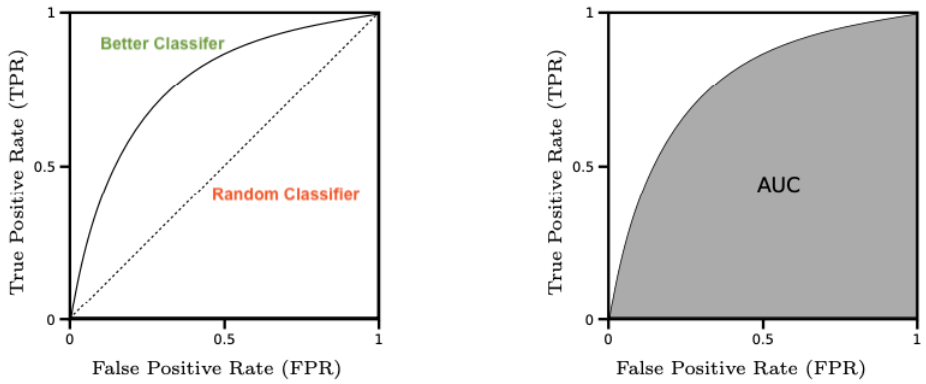


Figure 2.6: Example of an ROC curve and AUC graph depiction [1]

evidence within a forensics investigation. Generally however, an artefact describes a piece of datum that is dependent on or derived from a larger set of similar data [83].

2.3.1 Comparison with IT and IoT

There are some similarities between ICS, IoT and IT systems regarding the sources and data artefacts that can be used to build anomaly detection models, or that can provide digital evidence within a forensic investigation. One significant difference between the data sources found within IT and those used in ICS and IoT is that the latter two domains feature sensors, actuators and other IEDs that control or monitor data produced from physical processes. Regarding ICS, this does not only include the digital manifestations of sensor and actuator data, such as their relation to PLC variables within the controller memory, but also the measurable real-world physical quantities provided by the sensors and actuators, which often includes temperature, pressure and volume levels. These types of device generate a large amount of real-time data that is challenging to store and perform live analysis on, particularly in IoT systems, due to the extensive network interconnections and their distributed nature [84]. While IoT systems certainly consist of IT technologies, including cloud services, application servers, and smart devices (e.g. smart phones and tablets), there is a significant difference in the way that networks within IT and IoT are configured, with the latter relying heavily on distributed and mesh network technologies. Figure 2.7 summarises some of the key differences between IT, IoT and ICS digital artefacts and sources of data.

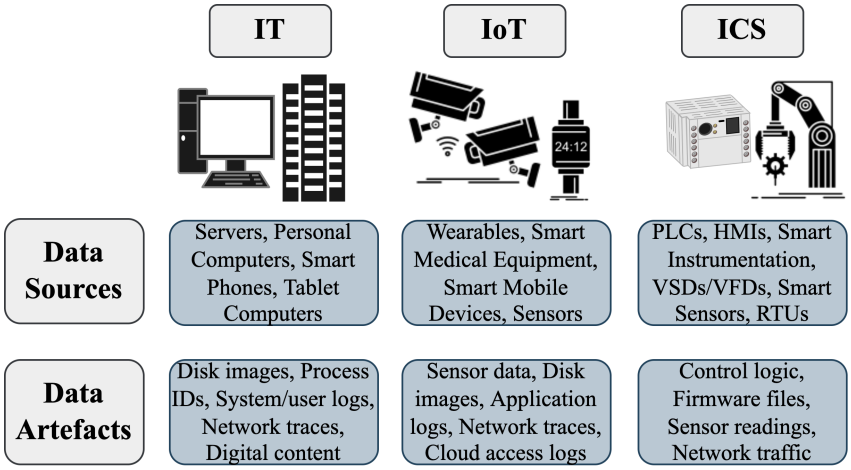


Figure 2.7: Comparison of Digital Forensic Evidence Sources and Data

2.3.2 Overview of ICS Data

Proprietary technologies are abundant within ICS. Many leading ICS vendors such as Siemens, Rockwell Allen-Bradley and Schneider Electric, implement custom file systems and firmware structures in their components, resulting in a wide range of data formats that are challenging to acquire and subsequently examine. An often sparse supply of publicly available documentation detailing these proprietary technologies impacts data acquisition and analysis security activities. Inconsistencies between vendor-openness is also clear, with some vendors providing detailed information outlining the structures of protocols, while others are reluctant to offer any information. There is often limited technical documentation for legacy equipment where the concept of 'security through obscurity' was applied in order to prevent attacks from potential adversaries. Reverse engineering can be used to address the aforementioned, and to better understand the operation and technical details of proprietary components, software or network protocols, as seen in previous studies [85], [86]. However there are overarching questions regarding the legality of using such methods, particularly concerning the intellectual property of vendors. The extensive use of vendor-specific technologies also impacts on training personnel in ICS incident response techniques, and on wider security controls for ICS, particularly regarding the analysis of binary and firmware files.

To date, limited research has examined what data can be acquired from ICS components, specifically PLCs and HMIs, that can then be used for anomaly detection and as evidence in digital forensic investigations. Overall, the existing literature focuses on establishing the distinctions between ICS artefacts and their IT counterparts [32, 87, 88]. For example, previous

research reported in [88] provides a list of data that can be acquired from different devices within three common network layers; control zone, data zone and corporate zone. In fact, the authors suggest examples of PLC data elements including but not limited to logs, timestamps and firmware versions. Although some of these data artefacts are specifically derived from ICS components, they are largely generic by definition. Furthermore, they do not provide any detailed information on the structure of these data artefacts or how they differ from their IT counterparts, for example timestamps from Linux workstations. An additional study which continued down a similar research path, also followed the same trend as previous studies by referring to generic memory storage types such as volatile and non-volatile data, rather than providing much new understanding or technical detail on how the artefacts contained within these storage media are different in ICS when compared with IT systems [32].

More recent work presented in [89], explores types of data artefact that can be found within an ICS and provides comparisons on how these differ with similar artefacts found in IT systems, such as network session data and user data. Although these again were, on the whole, limited to more generic categorisations of data, such as hardware data and raw data, the authors do highlight how these artefacts would be represented differently in an ICS context. Being able to understand the structure of ICS data artefacts and how they differ from IT artefacts is critical to the development of ICS anomaly detection capabilities and also digital forensic tools. Future work on highlighting the structural nuances of these artefacts and how they differ between different PLC vendors is also key to formulating greater data standardisation across ICS manufacturers.

To illustrate the current understanding of data artefacts found in ICS, table 2.3 presents a comprehensive list of device and network data artefacts that have been previously identified in the literature. Different artefacts are available from ICS devices at each layer of the Purdue Model, as outlined in figure 2.2. In some cases, the evidential importance and format of artefacts will vary, depending on which layer of the Purdue Model it is acquired from. For example, PLC firmware acquired at layer 1 will be a live version of the firmware extracted from the PLC itself, whereas if acquired at layer 3, it will be the raw firmware file. Consequently, the volatility of these two examples of firmware artefacts are likely to be different, with the live version being stored within both volatile and non-volatile PLC memory space. Moreover, some artefacts identified in table 2.3 are not ICS-specific and are found

in other computing domains such as IT. However, this does not mean that the data format of the artefact itself will be generic, as many device and network data types in ICS are in vendor-specific formats, resulting in limited artefacts having acquisition solutions. Control logic or application code, for example, has been acquired from PLCs in previous studies through contrasting approaches, however a generalisable and vendor neutral method for this is yet to be developed.

Table 2.3: Review of literature including data artefacts from devices found in OT/ICS components (For category (Cat) *G = Generic*, *I = ICS-specific*; We define acquisition (Acqu) as whether the capability to obtain an artefact has been seen in previous literature where ✓= *acquirable*, ✗= *not acquirable*)

Data Type	Data Artefact	Purdue Layers	Example Data Source	Cat.	Acqu.	Relevant Studies
Device Data	Disk images	3	EWS	G	✓	[90], [87], [91], [92], [93], [94], [95], [88]
	Firmware files	1 - 3	HMIs, PLCs	G	✗	[88]
	System logs	3	HMIs and EWS	G	✓	[90], [87], [96], [94], [88], [97]
	Control logic/application code	1	RTUs, PLCs, HMIs	I	✓	[32], [98], [99], [100], [90], [86], [101], [87], [96], [88], [26]
	Volatile memory (RAM)	0 - 3	PLCs, HMIs, EWS	G	✗	[32], [99], [90], [86], [87], [91], [93], [96], [94], [95], [88], [26]
	Engineering project files	1 & 3	EWS	I	✓	[98]
	Embedded device diagnostics and logs	0 - 2	PLCs, HMIs	I	✓	[98], [102], [90], [91], [88], [26]
	Program, processes and service management	2 & 3	HMIs, EWS	G	✓	[90], [87], [94], [88], [97]
	Non-volatile memory	0 - 3	PLCs, HMIs, EWS	G	✓	[90], [87], [91], [92], [93], [96], [94], [95], [88]
	User account logs	2 & 3	HMIs, EWS	G	✓	[90], [88], [97]
	Flash storage (memory cards)	1 & 2	PLCs	G	✗	[32], [90], [87], [91], [92], [96], [94], [88]
	PLC variables and tags	1 & 3	PLCs	I	✓	[103], [18]
Network Data	Proprietary ICS communication protocols	0 - 2	RTUs, PLCs, HMIs	I	✓	[32], [102], [100], [104], [86], [101]
	IP addresses and ARP tables	3	EWS	G	✓	[32], [102], [90], [87], [96], [94], [88], [97]
	Network logs	1 - 3	PLCs, HMIs, EWS	G	✓	[32], [102], [99], [90], [101], [87], [92], [105], [93], [94], [95], [88], [97], [106]
	Embedded device I/O traffic	0 - 3	RTUs, PLCs, HMIs	I	✗	[32], [102], [100], [90], [86], [91], [105], [96], [88], [26]

2.3.3 Detection Input Data Sources

ICS data artefacts have been used for ICS anomaly detection and digital fingerprinting in previous research, however primarily in a non-synthesised and independent manner. In other words, most of the research has based the feature sets for anomaly detection on one type of source of data, rather than using a combination of sources. Sensor and actuator data from industrial process equipment, particularly those that provide discrete values from digital signal components, are often used to model the baseline behaviour of an ICS. Each state that the underlying physical process, controlled by the ICS, can exist in, can typically be associated with a unique set of actuator and sensor values [103]. As PLCs are commonly the primary controller of these physical processes, many of the variables that the PLC uses within the application code are logically mapped to a sensor or actuator. Consequently, PLC variables, also known as registers, have been used as a source of data in proposed PLC anomaly detection models as they are often obtainable through network protocol commands, discussed later in Section 2.3.4.

Studies reported in [65], [103], and [107] all examine the use of PLC registers as feature sets for anomaly detection. The study described in [65] demonstrates an early use of PLC registers by converting the binary register values into time-series signal states and uses this as training data for supervised ML. The work in [103] extends on the concept of using PLC registers. An anomaly detection approach is proposed using convolutional neural networks (CNN) with deterministic finite automation as a process for modelling state changes in PLC registers and predicting future PLC register patterns. The model builds a Boolean logic matrix comprising discrete register values, including inputs and outputs that correlate with changes in the physical process. The study addresses the challenge of generalisability that previous work has uncovered by evaluating several widely used ICS network protocols with multiple PLC models, demonstrating that PLC registers can be generalised as an input data source for anomaly detection. The authors of [107] built on their previous work in [65] by using PLC input and output registers to generate finite state machines, representing the time-series behaviour of the registers, which is then used to generate Petri nets. The Petri nets are subsequently used as a baseline for PLC anomaly detection and are compared with test samples containing anomalies, to demonstrate changes to dynamic PLC behaviour. As the authors only utilise PLC input and output registers, it can be expected that this approach

would be unlikely to cover all types of PLC state that could be represented, as other register areas such as holding registers, that are often used in PLC programming, are omitted from the feature set. An additional challenge when using PLC registers is the extent of computational overhead that can result from large feature set dimensionality [69]. ICS facilities with a high number of sensors and actuators will result in high dimensionality, since each sensor and actuator typically corresponds to a PLC register. Therefore, feature dimensionality reduction techniques are clearly important in this case, for example, approaches used as part of the anomaly detection model might be the aggregation of numerical features, or the identification of the most powerful features [108].

Another data source that has been extensively evaluated for behavioural anomaly detection is data derived from the physical properties of ICS equipment. One area of research has used physical quantities from sensors and actuators, including analog measurements, such as pressures and temperatures, to generate baseline models. As ICS use an array of field-level devices to measure physical quantities, there is inevitably certain amounts of interference that can cause noise within datasets and inhibit accurate detection systems, particularly impacting on FNR and FPR metrics. A body of research conducted at the Singapore University of Technology and Design has demonstrated methodologies for ICS anomaly detection that exploit this noise generated by fluctuations in sensor and actuator measurements, to build accurate fingerprints of a given physical process [31, 109–111]. The work described in [110] utilises a Kalman Filter, an algorithm that uses time-series measurements to predict estimates of unknown variables through probability distribution, to provide predictions of sensor and actuator states [112]. The resulting state estimates are then subtracted from the real observed state values to provide a residual vector, comprising features related to sensor and actuator noise, which is then used as an input for an ML algorithm for attack detection. Related research described in [111] proposes a similar technique referred to as Process Skew, which uses subtle noise generated by measurement offsets within the physical processes. These process inaccuracies are generated and accumulated over time to produce the process skew, where a linear regression model is then used to provide a unique fingerprint for each process, along with a Cumulative Sum (CUSUM) detector that identifies subsequent abnormal deviations. One benefit of using physical measurement data as an input source is that the metrics and features extracted for anomaly detection can be generalised to different ICS use cases, such as water treatment, manufacturing or electricity distribution, since the unexpected behaviour

of sensors and actuators is not domain-specific. Therefore, there are particular advantages of using physical properties as input data sources instead of data that can be dependent on the process or component vendor, such as industrial network traffic. Work put forward in [113] offers a methodology that is component specific but that is still argued to be generalisable to different vendors. Specifically, the study uses side-channel data produced by the execution cycle of PLCs to provide a dynamic watermark that can be used to detect replay attacks, a type of network attack in which the transmission of valid data is maliciously repeated or delayed. The continuous operation of a PLC can be monitored to provide a scan cycle time, which is the duration of a single scan cycle iteration, consisting of reading input variables, executing the logic within the application code, and subsequently updating output variables accordingly [113]. Indeed, a further study presented in [75] uses the temporal properties of PLCs, mapped to the physical process, to perform change detection on PLC application code. The authors argue that by leveraging the resource constraint nature of PLC program execution, they can accurately detect local and remote changes to single instructions within the PLC control logic, which would be particularly beneficial in cases where complex control loop mechanisms are used, such as a proportional–integral–derivative (PID) controller.

Several previous studies have applied the characteristics of ICS network data to fingerprinting methods for anomaly detection [73,114–117]. Common network metrics have often been used such as packet size, inter-packet arrival time, and packet response time [114, 115]. As ICS network traffic is primarily deterministic, models of normal system behaviour can be accurately generated, in comparison to generic IT systems, where there would typically be a greater amount of noise in the network data due to less deterministic user behaviours [118]. However, network traffic is often limited to passive collection of visible data, which would result in an incomplete set of PLC registers and other device data [103]. Consequently, network data can often be too generic to identify subtle attacks targeting ICS components, particularly attack techniques that cause perturbations to PLC memory artefacts, which are discussed in greater detail in Chapter 3. Work presented in [119] used a combination of network traffic features extracted from the Modbus-TCP protocol, to train an OCSVM model and identify abnormal network behaviour. A data pre-processing method is proposed that establishes combinations of Modbus-TCP function codes and the register start address, embedded within the protocol message sequence. Through their evaluations, the authors demonstrate through low FPRs and high accuracy values that OCSVM is particularly suited to small

dataset sizes and ICS network traffic properties.

Logging mechanisms, particularly those provided by ICS component vendors, for equipment such as PLCs and HMIs, are designed to report on information related to process and operation faults within equipment/networks on a plant floor [27]. The reported logs and alarms are intended to assist engineers with fault troubleshooting rather than providing detailed information that would be useful to asses if a suspected cyber-incident has occurred. Indeed, many existing studies suggest that logged process and operational data from ICS is often inadequate in aiding forensic investigations to any great extent [120], [19], [92], [102]. Furthermore, the lack of memory capacity and storage within ICS components, particularly legacy devices, results in logs being overwritten and lost within hours or even minutes of initially being recorded. Consequently, this drastically reduces the chain of evidence and leads to additional challenges with respect to the quantity of data to collect and prioritising the significance of that data [95]. It is important to note, however, that existing logging mechanisms, while often seen as currently inadequate for security purposes, should not be completely dismissed as a potential source of evidence. For example, a PLC log file might contain data on the PLC's operation status mode changing from RUN mode to PROGRAM mode, potentially indicating an unauthorised upload of code to the PLC. Hence, this implies that indicators of compromise can be determined from PLC log files and that they may provide greater utility for attack detection rather than post-incident forensic analysis [120].

The use of different types of log data for ICS anomaly detection is considered by a series of studies [121], [122], [123], [124], [125]. For instance, the study described in [121] introduces the HAMIDS (HierArchical Monitoring Intrusion Detection System) framework. HAMIDS utilises deep packet inspection functionality provided by Bro[4], an open-source network monitoring and analysis framework, to parse and analyse ICS network traffic, and supports a number of industrial protocols such as Modbus. Furthermore, HAMIDS places network monitoring nodes at multiple positions in the ICS network, including at the supervisory and control layers, which enables the detection of anomalies that are distributed within the ICS, thus targeting multiple end-points. The work proposed in [123] extends the HAMIDS framework [121] and demonstrates an approach using state-dependent detection thresholds, which provide the estimation of a system state from network and physical pro-

[4]Bro was renamed to Zeek in 2018

cess data. This added further support for the value of recording logs of physical properties pertaining to kinetic changes within the underlying process. Sensor noise in particular was considered to be a significant parameter for system state detection.

Other studies have looked to use logs containing process data composed from state changes within the physical process [122], [124], [125]. However, the logs used in these studies are defined as messages that report values equivalent to using attributes of PLC registers, as reported by papers discussed earlier in this section, where a log event contains process data related to sensors and actuators in the physical process. In [124], a PLC anomaly detection approach is proposed that uses log messages and diagnostic buffer data from the controller itself. Furthermore, the authors generate custom events that represent events from the PLC diagnostic buffer, however there are limitations with generalising this data across PLC vendors and therefore the approach may only be applicable to Siemens PLC models, as evaluated in the paper. Critically, the approaches adopted in this thesis do not rely on the vendor-specifics of PLC logs such as the contents or structure of a generated log event, but rather utilise generalisable log-based metrics that are vendor-neutral. The work in [122] approaches the challenge of ICS anomaly detection using process mining techniques, specifically conformance checking analysis. The authors propose that Petri-Nets can be used to model the expected control flow behaviour of the physical process using attributes extracted from process data log events generated by an HMI. Petri-Nets are a form of graphical modelling that can be used to mathematically describe the behaviour and relationship between system states, components and dynamic changes using a set of places, transitions and arcs [126].

Giving consideration to the type of input data is an important step in the design of anomaly detection models, with some data sources benefiting generalisation and others enabling deeper analysis of anomalous behaviour in ICS, due to the increased granularity within the datasets. An additional factor to consider is the method through which the data is acquired from the target system or component. Within the field of anomaly detection, acquisition techniques can either be *passive* or *active* [116]. Passive techniques are not normally intrusive and involve using data flows that exist without any external influence or input to generate them. Conversely, active techniques require a user to introduce additional data requests to probe a system, often to acquire information that is not observable through passive approaches. Using active acquisition techniques can have adverse impacts on system availability and

Table 2.4: Comparison and summary of the different types of data used by existing ICS anomaly detection studies **Key:** ●= Coverage, ◗= Partial Coverage, ❍= No Coverage.

Approach	PLC Registers	ICS Log Data	PLC Application Code	Network Data	Side-channel Data
Caselli *et al* (2013) [114]	❍	❍	❍	●	❍
Peng *et al* (2015) [115]	❍	❍	❍	●	❍
Formby *et al* (2016) [73]	❍	❍	❍	●	●
Ghaeini & Tippenhauer (2016) [121]	❍	●	❍	❍	❍
Ahmed *et al* (2017) [31]	❍	❍	❍	❍	●
Yau & Chow (2017) [65]	●	❍	❍	❍	❍
Ahmed *et al* (2018) [109]	❍	❍	❍	❍	●
Ahmed *et al* (2018) [110]	❍	❍	❍	❍	●
Dong and Peng (2018) [119]	❍	❍	❍	●	❍
Ghaeini *et al* (2018) [123]	❍	●	❍	❍	●
Myers *et al* (2018) [122]	◗	●	❍	❍	❍
Settanni *et al* (2018) [124]	◗	●	❍	❍	❍
Shen *et al* (2018) [116]	❍	❍	❍	●	❍
Chan *et al* (2019) [67]	●	❍	❍	❍	❍
Ghazo & Kumar (2019) [117]	❍	❍	❍	●	❍
Hussain *et al* [125]	◗	●	❍	❍	❍
Stockman *et al* (2019) [63]	❍	❍	❍	❍	●
Ahmed *et al* (2020) [111]	❍	❍	❍	❍	●
Formby *et al* (2020) [75]	❍	❍	◗	❍	●
Yang *et al* (2020) [103]	●	❍	❍	❍	❍
Yimer *et al* (2020) [127]	❍	❍	❍	●	❍
Ahmed *et al* (2021) [113]	❍	❍	❍	❍	●
Yau *et al* (2021) [107]	●	❍	❍	❍	❍

real-time operations, which are fundamental requirements of ICS, resulting in some studies implementing passive techniques instead [113] [115] [117]. Both the studies by [115] and [117] use passive acquisition of TCP/IP communication streams within ICS networks to fingerprint the baseline network behaviour of a given system, that can, in turn, be used to detect network-based anomalies. Using TCP/IP steams facilitates a generalisable framework for detection since modern ICS network implementations have moved away from point-to-point (P2P) communications built on serial protocols and adopted more computationally optimised Ethernet-based protocols. Many ICS components often employTCP or Universal

Datagram Protocol (UDP) in conjunction with IP. The primary disadvantage of only using passive data acquisition approaches is that adversaries may have hidden data that cannot be observed by the anomaly detection system resulting in attacks being unnoticed. Instead, a hybrid model using passive and active techniques may be more suitable, as presented in [103]. However, uses of active data acquisition techniques for ICS anomaly detection should include an evaluation of the potential system impacts, resulting from the use of such methods, that many studies within the field fail to provide.

In summary, the majority of studies discussed in this section utilise network data metrics, and properties of physical quantities and side-channel data as the data input sources for anomaly detection and fingerprinting, as detailed by the comparison in table 2.4. Significantly, very few studies address anomaly detection or diagnosis through formulations of memory artefacts such as PLC registers and no studies were found to use PLC application code despite its critical importance in PLC operations. Moreover, data sources are primarily used independently, with only the work described in [73] offering a hybrid approach using a convergence of both side-channel and network data for establishing an ICS fingerprint. In contrast to existing literature, the approaches presented in this thesis primarily use synthesised combinations of different types of data to generate specialised feature sets for PLCs. Specifically, we will look to focus on device-related data that can be extracted from PLCs, including register and application code data. While previous literature has often dismissed the use of PLC logs specifically, we will also explore how the use of such data can be used within the context of anomaly profiling, and how the events generated in PLC logs are correlated with different types of anomalous behaviour. In general, the combinations of data that we use for anomaly diagnosis is particularly novel compared with existing research.

2.3.4 Artefact Acquisition Techniques

The extensive use of proprietary technologies in ICS results in a wide range of different protocols, file types and code structures, consequently making it challenging to use traditional data acquisition techniques that have been used in IT systems. At the time of writing, no standardised method or tool exists for the acquisition of potential evidence from ICS devices, including PLCs. A significant lack of conventional data access interfaces on ICS components, such as USB ports and graphical-user interfaces, means that software-based solutions to data

acquisition that are installed on the target device cannot be applied to components such as PLCs.

Commercial Tools

One area of considerable limitation is that of commercially available data acquisition tools that are specifically designed for ICS components and networks. The limitation in this area is largely driven by the heterogeneity of devices and the lack of standardisation across different ICS component vendors [32]. However, some ICS-specific monitoring and logging software capabilities could provide valuable information on a cyber incident and potentially yield data artefacts for detection and subsequent investigations. Some examples include Forescout [128], Nozomi Network [129] and Microsoft Defender for IoT [130]. Previous studies have also advocated that data historians, normally used to store operational data such as sensor measurements, could provide vital data artefacts for both detection and digital forensics where access to control system components such as PLCs or RTUs is more challenging, for example in high-intensity environments or geographically remote locations [95] [93] [131]. Moreover, a limited set of existing security tools have been modified or extended to include functionality for specific ICS technologies. An example of this is Wireshark, a free widely-used network traffic and packet analyser [132]. In addition to having dissectors for common network protocols such as TCP, IP and UDP, Wireshark has also been extended through the open-source developer community to include dissector plug-ins for ICS network protocols, including Siemens S7Comm, Common Industrial Protocol (CIP), Modbus-TCP and Profibus/Profinet. These extensions enable users to capture an analyse network traffic that is specific to the ICS.

Acquisition through Hardware Ports

Many embedded components, including those found in ICS, contain on-board chipset ports for various debugging and testing purposes. An example is an industry standard called Joint Test Action Group (JTAG), which is traditionally used for testing the circuitry on printed chipboards and debugging embedded software and firmware. JTAG can be used for data acquisition purposes by connecting a JTAG emulator to the Test Access Port (TAP) located on the embedded system chipboard. Since JTAG facilitates data acquisition for devices where

conventional approaches cannot be applied, it has also been assessed in the contaxt of ICS devices [87] [133] [134] [96]. However, the general consensus provided by the literature acknowledges the challenges in taking this approach for data acquisition. In addition to requiring specialist knowledge, it is clear that not all devices have TAP ports available, or activated by the manufacturer, and therefore would not support this as a method of data acquisition [133]. Work presented in [135] demonstrates the benefits that JTAG acquisition can offer to PLC memory acquisition by proposing the Kyros methodology, which generates JTAG profiles for target PLCs through a JTAG debugger and installing JTAG header ports on the contact pad of the PLC. The resulting acquired memory contents could provide evidence for stealthy attacks targeting PLC control logic. However, it is important to note that there would be challenges with implementing such an approach for real-time anomaly detection. Therefore, JTAG could be a solution for post-incident acquisition, assuming the JTAG ports are accessible, although the acquisition process itself would likely cause significant operational disruption to the ICS, since access to the PLC's chipset would be required.

An alternative approach described in [136] involves a method of volatile data acquisition by interfacing with the PLC bootloader system through a Universal Asynchronous Receiver Transmitter (UART) interface attached to the PLC's chipboard. The research specifically examined the Siemens S7-1200 PLC and identified that access to CPU handlers could be gained by issuing interrupt commands to the PLC bootloader during the initial bootup process and placing the PLC into a "special access mode". Specific commands were then able to be issued to the PLC such as retrieving firmware and bootloader versions, read bootloader values, dump internal RAM contents and perform various hardware tests. However, it is important to note that the evaluation of the approach proposed in [136] was limited to one specific PLC model. It is therefore unknown whether this could be applied to PLCs from other vendors, or even alternative PLC models from the same vendor. Furthermore, the methods discussed in this section present very limited practicality in terms of being transferred into live ICS environments for real-time anomaly detection. In general, it would be impractical to expect industry professionals to have the required specialist knowledge to perform such techniques or to have the willingness to access the chipset of a live PLC whilst in operation.

Network Communication Data Requests

One particular data acquisition approach that has previously been explored and reported in the literature, involves leveraging data transfer functions, that are built into many industrial communication protocols, to obtain artefacts [86]. ICS devices typically operate through an open request-response communication architecture where one node on the network will issue a request to another node, such as reading or writing a specific variable, and the second node will reply with the requested data in the response packets, as long as the data is available. An example of this could involve communications between a PLC and an HMI, where the HMI is continuously requesting the status of PLC memory registers related to particular output devices. Upon receiving the data from the PLC, the HMI can update its internal memory contents accordingly so that an operator can determine if the system is working as expected. The transparency of communications between nodes on the ICS network enables PLCs to easily communicate and distribute data to other ICS devices without interruptions.

Research has provided some initial insight into how the PLC built-in request-response data communication method could be used for ICS data acquisition, specifically for digital forensics and security [32, 86, 133]. Work reported in [86] assessed whether the request-response method could be used to acquire memory states of General Electric (GE) Fanuc PLCs using the General Electric Service Request Transfer Protocol (GE-SRTP), which is a TCP/IP application protocol, often referred to as SRTP. The GE-SRTP protocol uses service request codes to retrieve and exchange inter-network node information following similar steps to other ICS communication protocols such as Siemens S7Comm and Modbus. For example, service request code 0x25 in the GE-SRTP protocol will return the PLC's system data and internal clock time. Evidently, it is feasible to acquire artefacts from PLC memory through the protocol service or function codes, especially data stored in non-volatile memory space such as the PLC application program [86]. Although this study examines an individual model of GE PLCs, the Fanuc Series 90-30, the authors argue that their approach would be applicable to any PLC that uses this protocol, and that the overall methodology could be generalised to other communication protocols. Complementary to the aforementioned study, is work described in [133] which evaluates the challenges of performing live volatile memory acquisition on PLCs and other types of ICS device that operate within the typical layers zero and one of the Purdue Model. Specifically, the authors suggest that an approach similar to the

Table 2.5: Open-source tools for ICS data acquisition using network data transfer

Tool	ModbusTCP	ModbusRTU	S7Comm	EtherNet/IP	Other
LibModbus [137]	✓	✓	✗	✗	✗
LibPLCTag [138]	✓	✗	✗	✓	✗
Modbus-TCP Client [139]	✓	✗	✗	✗	✗
NodeS7 [140]	✗	✗	✓	✗	✗
PLCScan [141]	✓	✗	✓	✗	✗
PyComm [142]	✗	✗	✗	✓	✗
PyLogix [143]	✗	✗	✗	✓	✗
Snap7 [144]	✗	✗	✓	✗	✗
ICS Mem Collect [145]	✗	✗	✗	✗	✓

method presented in [86] could offer a possible generalisable data acquisition solution for Schneider Electric PLCs, however, no technical information or experimentation protocol is provided to support these claims on the presented Schneider Electric use case.

As we have already discussed, one fundamental limitation is the lack of commercial data acquisition tools and solutions for ICS networks and components. However, a number of open-source tools have been developed by third-parties and independent developers that utilise the data capturing functionality of ICS network protocols presented in the aforementioned studies. These tools were not designed specifically for security use cases or to be executed on live ICS equipment, however with no commercial solutions currently available, these open-source tools could provide some relief to the significant limitations in ICS data acquisition. Table 2.5 provides an overview of some of these tools and highlights which industrial protocols they are known to be compatible with. Many of the tools provide functions for device identification and network discovery by actively probing devices for meta-data, including but not limited to, device model, serial numbers, and IP addresses. One particular challenge that is emphasised babout the tools explored in table 2.5 is the lack of cross-vendor support. In many cases there exists a one-to-one pairing between vendor and protocol functionality, for example the S7Comm protocol and Siemens devices, or EtherNet/IP and Rockwell Allen-Bradley components. Hence, different tools are required to acquire data from different PLC models and vendors, which is a key challenge that is referred to throughout this thesis.

2.4 ICS Digital Forensics

Digital forensics can be defined as the branch of forensic science that involves using scientifically derived methods to perform acquisition, analysis and preservation of digital data used by computers and other digital sources [25]. More specifically, forensic science principles and methods are used to investigate criminal activities where the forensic process typically follows a formal investigation framework that often includes stages of data collection, containment, analysis and presentation [146]. As digital forensics became a fundamental part of criminal investigations, there was a growing need for the introduction of standardisation, to show transparency within the forensic process. Digital forensics was introduced into the ISO/IEC 27000 series of international standards on the security of information systems under ISO/IEC 27037, 27042 and 27043 [147] [148] [149]. These three standards describe the formal processes and terminology digital forensic investigations should conform to, including the handling of forensic evidence and the stages that should be followed. Despite the forensic investigation process involving identification, collection, acquisition and preservation as core stages, presented in one of the first digital forensics process models at the Digital Forensics Research Workshop (DFRWS) in 2001 and subsequently introduced into international standards, many studies have explored alternative investigation approaches by introducing additional stages to the forensic process [150]. However, the digital forensic investigation itself, only forms half of the broader forensic life-cycle, illustrated in figure 2.8. The initial proactive stages focus on forensic readiness, which relates to the preparedness of an organisation to respond and recover from a cyber attack [151]. Understandably, adequate forensic readiness is often regarded to be a core requirement of successfully performing the subsequent digital forensic investigation phase [151, 152].

2.4.1 ICS Forensics vs. IT Forensics

ICS forensics can be viewed as a subcategory of the wider digital forensics field, however it is important to highlight the two fundamental differences between IT and ICS forensics. The first difference regards the forensic evidence sources and data artefacts that would be collected and examined during and after an incident occurs. In IT digital forensics, tools are used to acquire the volatile memory space of a personal laptop, or recover deleted

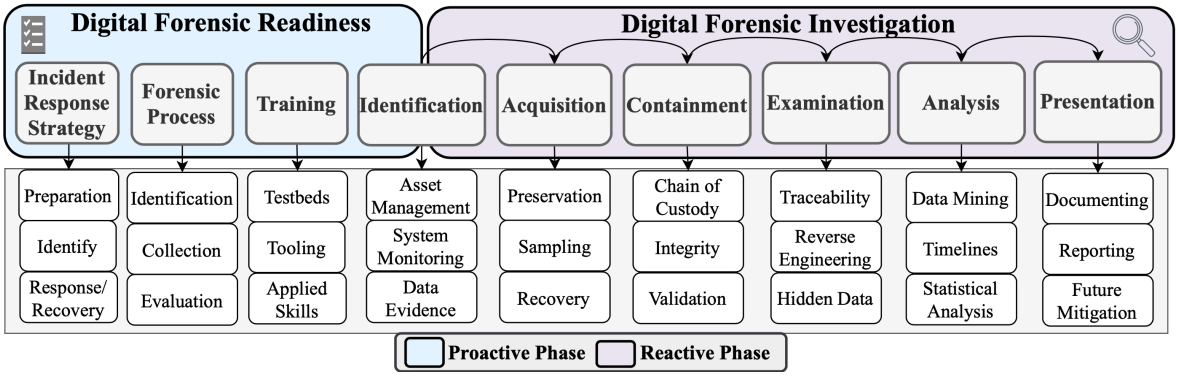


Figure 2.8: Digital Forensics Life-Cycle comprising forensic readiness and forensic investigation phases

emails from a corporate email account system. Conversely, digital forensics for ICS involves analysing data from industrial devices and networks that drive the physical operational processes specific to that industrial system. Unlike their IT equivalents, industrial technologies include many different types of device, for example PLCs and IEDs that often have vendor-specific or proprietary file formats and firmware layers that complicate the forensic investigation process, particularly those addressing data acquisition and analysis.

The second difference regards the deployment of forensic tools and techniques, and of cyber security controls more generally within ICS environments. Both ICS and IT digital forensic processes need to maintain the integrity of acquired data that could be used as evidence, however ICS forensics must also consider the mission and safety-critical nature of ICS by evaluating the risk to system's functionality caused by the application of forensic tools, and must mitigate interference with the industrial processes. For this reason, it is imperative that forensic methods, particularly those approaches that use live forensic analysis and data acquisition, must be strongly validated on representational testing environments before being applied to real industrial environments. This of course, is in addition to verifying the forensic integrity of the method itself.

2.4.2 Digital Forensic Challenges

The technologies used in ICS are very different to those found in IT systems, resulting in a wide range of unique challenges relating to the development and deployment of ICS forensics. Although several technological challenges impact aspects of forensic readiness, they generally place a greater burden on the forensic investigation stages, particularly data acqui-

sition and analysis processes. For instance, due to the proprietary nature of many ICS technologies, conventional IT forensic tools typically do not support interaction with the bespoke file systems and customised kernels incorporated within ICS components. Furthermore, is, or verges on, infeasible to conceptualise the development of an inter-operable forensic data acquisition tool compatible with all types of ICS components, or even all models of a particular component, due to the technical differences introduced by the vendors and often also the firmware-specific properties. Enabling the development of forensic data acquisition and analysis tools for ICS components therefore requires greater standardisation across vendors and models, of the technologies used by devices such as PLCs, particularly if there are to be generalisable frameworks and methodologies for ICS digital forensics, as should be the ambition. Although some common network analysis tools have utility with these protocols, it is impractical to assume that such tools will work with all existing ICS communications protocols [95]. Consequently, many asset discovery tools are not applicable to ICS environments creating barriers at the initial stages of the forensic investigation process.

Additional technology barriers challenge the ability to perform forensics in real-time. The requirement of continuous system availability that is often associated with safety and mission-critical ICS environments, significantly reduces the viability of a forensic investigator accessing an operating ICS to perform live forensic data acquisition and analysis. The risks posed to a system's safety and operational requirements through performing live forensics could include unexpected traffic, increase network latency, or reduce overall network bandwidth during run-time [27]. Moreover, logging mechanisms provided by ICS component vendors for equipment such as PLCs and HMIs, are typically designed to report on information related to process and operation faults within equipment/networks on a plant floor. Logs and alarms are intended to assist engineers with fault troubleshooting rather than providing detailed information that would be useful to asses if a suspected cyber-incident had occurred. Many existing studies argue that logged process and operational data from ICS are often inadequate in meaningfully aiding forensic investigations [120], [92], [102].

2.4.3 Conventional digital forensic tools

Early research in digital forensics for ICS largely examines the use of existing forensic tools to ICS environments. A lack of consistent terminology and definitions on what systems and

technologies an ICS comprises is a key limitation identified in early ICS forensic literature. Several early studies promote the applicability of existing digital forensic tools to ICS setups, specifically those originally developed for IT systems. However, when referring to ICS or OT systems, many studies are examining IT technologies used in OT environments [153] [154] [155] [156]. Examples of this include engineering workstations, and HMIs, both of which typically operate a COTS OS such as GNU/Linux or Windows. Regarding the use of IT systems within ICS settings, standard PC workstations can be utilised as HMIs through software, and engineering workstations often contains data artefacts that are directly related or used by ICS components, such as engineering project files. Therefore, the utility of IT digital forensic tools should not be completely dismissed when discussing ICS forensics. Nonetheless, when referring to ICS, more recent studies of research focus on the bespoke industrial components that make up the cyber-physical parts of the operational environment.

Research presented in [154] demonstrates the utility of remote ICS forensic acquisition and analysis using an existing IT forensic tool called EnCase. EnCase is a suite of IT digital forensics tools, though conventionally the software has been used for evidence recovery from seized hard drives as part of an investigation [157]. The authors examined how the functionality of EnCase could be applied to ICS environments, in particular to acquire the configurations of an HMI. Although the study shows that EnCase was able to acquire a copy of the HMI system's hard drive, the primary limitation was that standard IT hardware and software was used to emulate the HMI system, in this case a Windows XP workstation. Many HMI systems now are bespoke devices running on embedded OSs having the sole purpose of being an HMI rather than a PC-based system running a commodity OS such as Windows. By making HMIs more closed off, the attack surface could be reduced as less is known about the inherent vulnerabilities that are introduced with the bespoke OS, however this has a trade-off from a forensics perspective as it is likely that data is more challenging to acquire.

Other studies have taken similar approaches to addressing the limitations with ICS digital forensic [93] [96], [158]. For instance, research conducted in [93] brought existing digital forensic software and hardware into an ICS context by examining the applicability and relevance of components that could be contained in an ICS forensics toolkit. Specifically, the authors discussed utilising tools and equipment to perform stages of the digital forensic process, including acquisition (collection), preservation, examination and analysis. While this

study provides an initial understanding of what tools can be useful for performing forensics investigations in ICS environments, it is unlikely to be widely applicable to ICS components such as PLCs due to their use of non-conventional OS and file systems. Although analysis tools such as hex viewers and network traffic parsers are likely to be of use given their generic and scalable functionality, most existing data acquisition tools will not support the proprietary nature of PLCs and other components bespoke to ICS. The aforementioned is further justified by [96] where the authors discusses data within ICS that could be of evidential importance, as well as methods to acquire this data. Focus is given to volatile data, in particular RAM, and the significance of its acquisition in control systems. Specifically, this is conducted by using existing tools such as the open-source LiME loadable kernel module (LKM) tool [159] and the commercial FTK imager [160]. These are two examples of tools that can be used to extract the content of volatile memory from machines running on GNU/Linux and Windows OS, and therefore the tools are unlikely to have much applicability to ICS components running bespoke real-time OS environments or even those with firmware layers. While the authors highlight that important system and process data could be acquired from the ICS deployment through these tools, they provide little technical evidence of the tools being applied layers 0 and 1 of the Purdue model.

2.4.4 Post-mortem Forensics

Digital forensic methods are typically separated into two categories; 1) post-mortem (offline) forensics, and ii) Live forensics [161]. Post-morem forensics investigates the non-volatile memory artefacts of systems that have been shut-down prior to investigation. As a consequence of powering a system down, many volatile data artefacts are lost in the process. Live forensics proposes a solution to this problem, where an investigator can acquire and inspect the running processes, memory contents, registers and other artefacts that exist in volatile memory space, such as Random Access Memory (RAM) [161].

Post-mortem forensic analysis, therefore refers to the examination of acquired data at a later stage in the investigation, where analysis does not need to take place during live operation. Consequently, post-mortem digital forensic analysis focuses on the examination of non-volatile artefacts [162]. From a risk perspective, post-mortem analysis allows for evidence to be securely contained first and then analysed in environments unconnected to the

host system, reducing the risk of data integrity loss and anti-forensic techniques. Regarding post-mortem forensic analysis for ICS, previous research has largely focused on analysing data that can be acquired from PLCs. Two studies have examined the possibility of reconstructing acquired raw application code from the PLC and displaying it in the native format it was written in, such as ladder logic [163] [164]. The PLC application code is primarily a user-defined program built using a range of logic objects, timers, counters and input/output (I/O) addresses written to perform a specific task. Since application code is stored in non-volatile memory space on the PLC and does not change during the normal operation of the PLC, it is suitable for post-mortem analysis, typically by applying static analysis techniques.

As ICS devices often operate using bespoke file structures and memory systems that differ from IT components, standard methods to perform data acquisition do not have the same level of applicability when taken into an ICS environment. ICS devices that use proprietary and vendor-specific file systems, bootloaders, firmware and protocols, will unavoidably produce technical issues regarding the use of existing IT digital forensic tools. In particular, ICS devices such as PLCs do not have the same interfaces to access data as IT systems, for example Graphical User Interfaces (GUI) and desktop applications. The challenges presented by a lack of available interfaces affects both post-mortem and live data acquisition techniques since common tools including but not limited to EnCase, FTK and Belkasoft RAM Capture cannot be executed on the PLC system itself [157] [165] [166]. There is, however, scope to extend existing digital forensic tools, such as the examples just provided, to included specific functionality for ICS components. File system analysis tools, for example The Sleuth Kit (TSK) [167] which enables disk image examination, could be adapted to incorporate predefined structures of PLC control logic code to assist with ICS forensics, particularly due to the community of support for TSK and its open-source framework. Furthermore, as many raw data artefacts are in hexadecimal format, hex-editor tools, such as 010 Editor, could be used to build templates of common files used by PLCs and other ICS components [168]. The challenge here, however, is that if an ICS vendor is to change the file structure due to a firmware update, then the current templates would become obsolete and require updating. With the large number of different file structures provided by different ICS vendors, it would be very challenging to keep track of what changes have been made, which may result in digital forensic tools not operating as designed when they are required.

Although PLC application code is generally visually and syntactically similar across different vendors, the binary representation of the code is highly vendor-specific. To address this, an area of ICS digital forensics research has explored generalised approaches to translating PLC application code binaries into their high-level programming representations, such as ladder logic. The Laddis decompiler introduced in [163] is capable of taking raw binary ladder logic code in hexadecimal format and convert it to a graphical representation output so as to provide a visualisation of the implemented ladder logic. In more detail, Laddis analyses the individual ladder logic rungs, sometimes referred to as networks, and the instructions within each rung by identifying signature values that represent this data. The research evaluated the data accuracy of the decompiling process and found that 100% accuracy was achieved, showing that it is possible to analyse PLC application code without the need to use the PLC programming software environment to decompile it. Moreover, the forensic ability of decompiling was validated via three attack scenarios on a Rockwell Allen-Bradley PLC. Nonetheless, as the application code from only one PLC model was analysed and decompiled, the authors were not able to demonstrate the generalisations that could be extracted from this approach. The high-level graphical representation of ladder logic is similar across many different PLC vendors as it is a standardised language. However, as the structure of the raw hexadecimal binary data is determined by the PLC's internal decompiling process and firmware, we can assume that retrieved application code will not be returned in a standard format that is represented by all PLC models [32]. As there is a wide variety of PLC vendors, and even PLC models within those vendors, it would be challenging to scale this method to be applicable to the vast number of different PLCs. In addition, the decompiling process proposed in [163] assumes that the binary code will maintain the same format when vendors provide updates and patches to their PLCs. Consequently, while there is potential for data acquisition and analysis approaches to be generalised across PLC vendors, it is likely that there will still need to be significant tailoring towards vendor specifics.

A more recent study presented in [164] partially addresses these initial limitations by proposing Similo; an automated reconstruction process of the PLC application logic, expanding on the Laddis decompiler. Similo is a virtual-PLC framework, containing multiple binary decompilers for different PLCs, utilising previously captured ICS network traffic to perform the decompilation process into high-level PLC application source code. The functionality and network traffic enabling code retrieval from a PLC, commonly referred to as uploading

by PLC vendors, was monitored and used to design the functions of the decompiler. Similo was subjected to a training phase with the aim that it would learn dynamic header fields of network traffic packet capture (PCAP) files. Critically, this work clearly demonstrates that the decompiling of binary code can be achieved across different PLC vendors, in this case models by Rockwell and Schneider Electric. Additionally, the work highlights that the Similo decompiler was able to translate binary into multiple PLC programming languages as standardised by ISO/IEC 61131-3, rather than just ladder logic.

Generally, when performing live forensic acquisition on an IT system, the tool used for capturing the data typically needs to be executed on the system being interrogated in order to acquire data from volatile memory. Methods used to perform post-mortem forensic acquisition on non-volatile memory are not required to be executed on the system itself, as the storage medium can often be removed and imaged later in a lab environment. For instance, conventional Hard Disk Drives (HDD) or more modern Solid-State Drives (SSD), which are common storage technologies found through IT systems. However, PLCs do not utilise HDD or SSD technologies for non-volatile data storage, but instead typically use EEPROM that is embedded in the chipset and so cannot be removed from the device. Limitations with executing acquisition tools locally on the PLC, and the inability to extract any significant amount of removable storage media from the PLC, results in conventional post-mortem forensic acquisition tools being inapplicable to PLCs.

2.4.5 Support from ICS Vendors

Some digital forensic capabilities have been provided directly by IT vendors for their components and systems. An example of this is the Computer Online Forensic Evidence Extractor (COFEE) framework developed by Microsoft in 2006 as a toolkit for digital investigators examining Windows systems. COFEE offers a range of tools including volatile memory acquisition, and password decryption [169]. Notably, COFEE was developed and endorsed by Microsoft for Windows, suggesting that Microsoft acknowledged that successful forensics and cyber defence requires commitment and effort from vendors, third-party developers and end users. Conversely, there are no examples in the ICS domain to demonstrate similar understanding of, and commitment to, digital forensics. At the time of writing, no ICS vendor had supplied any form of digital forensics toolkit for their systems, despite the clear

requirement for such capability, as was the case in 2006 for IT cyber-crime.

However, some studies have examined the forensic capability and application of existing software functionality provided by ICS vendors, that is not specifically designed for forensic use [98, 99]. For instance, an investigation was conducted and reported in [99] that examined debugging tools that are provided by vendors for PLCs and how they might aid forensic investigations. In fact, the authors of this particular study examined a free tool called PLC Logger, which is able to capture, store and analyse data from numerous PLC vendors operating on different communication protocols, including the Siemens S7Comm and Modbus TCP protocols [170]. Despite the tool being designed for diagnostic applications, it was reported that it can also be used to acquire data from PLC memory such as the control logic of the PLC application program. The authors claim that such an output could subsequently be compared with an existing baseline of the code, to determine if alterations were present. Thus, the ability to acquire the application code from the PLC emphasises the potential forensic importance of this tool. Additional tools and capabilities are not required as this functionality is provided through the official PLC vendor programming software, for example TIA Portal for Siemens PLCs and Studio 5000 (RSLogix) for Rockwell Allen-Bradley controllers. However, previous ICS cyber-attacks, including Stuxnet, have highlighted vulnerabilities in the communications channels between the programming environment and the PLC [9]. Consequently, there are issues of trust in the integrity of the data, since it is possible that the network communication session between the engineering workstation and the PLC could be compromised, as was evident in Stuxnet.

An additional study explored the possible forensic functionality provided by the PLC vendor programming software, again focusing on Siemens PLCs and the TIA Portal engineering software [98]. The study identified that the inherent logging systems and diagnostics features can be used to identify unexpected PLC behaviour, such as changes in operation mode, and the uploading of new code to the PLC. As well as examining the functionality provided by the software itself, the authors provide an analysis of the PLC programming project files that are created by the programming software. These contain the application code, hardware and network configurations, and other meta data that is downloaded to the PLC when being commissioned or updated. Although it is reported that the timestamps of the project files can be used to indicate potential malicious activity, in particular data manipulation attacks, no ad-

ditional information was examined within these project files. This is likely because they are often in vendor-proprietary binary formats that cannot be analysed by standard hex-viewer tools. Furthermore, additional studies have shown that code injection can take place without the need for uploading project files to the PLC [163], [164]. In this case, the timestamps of the project files would not necessarily be updated.

2.5 Summary

Two separate but strongly interrelated fields have been reviewed in this chapter. As discussed in Section 2.2, anomaly detection research has enabled early identification of behaviours within the ICS that are outwith a baseline of normal operating conditions, primarily through the use of ML algorithms. Many recent proposals have focused on the use of ML methods to address challenges related to real-time detection and scalability to large interconnected ICS.

As described in Section 2.3.3, many studies have advocated the use of ICS-specific data sources for feature set generation in order to take advantage of the unique properties of the relationships between the cyber and physical domains. Many recent works have used measurable physical properties such as temperature and pressure to detect attacks on sensors and actuators. However, this data often introduces challenges where noise filtering is required to improve detection accuracy and increase the signal-to-noise ratio [171]. Due to its physical nature, different types of anomalies can occur, in particular cyber-attacks and system faults. Therefore, as discussed in Section 2.2.3, different anomaly detection approaches have been proposed to optimise the identification of these two categories of anomaly, while remaining independent of each other.

Within the context of ICS, Digital forensics research, as presented in Section 2.4, on the other hand, has typically focused on post-incident investigations to perform attribution and methods for post-hoc analysis, and therefore relies on evidence and data preservation. There are, however, challenges with performing live digital forensics in ICS environments, particularly due to the risk of impacting the real-time requirements of heavily resource-constrained devices.

Chapter 3

Experimental Methodology

3.1 Overview

The key focus and contribution of the research reported in this thesis addresses a relatively undeveloped area positioned between anomaly detection and digital forensics, referred to as *Anomaly Diagnosis*, and illustrated in figure 3.1. It is integrated with high-level security research fields and NIST CSF cyber defence paradigms, which were introduced at the start of Section 2.2. The core objectives of anomaly diagnosis are to better understand how different types of anomalous behaviour can be profiled, to assist with early incident triaging, and to inform subsequent digital forensic investigations.

A number of previous studies have proposed methodologies for network and host anomaly detection within the context of ICS, with many utilising ML algorithms to optimise the automation of the detection process. Furthermore, digital forensics research has examined post-

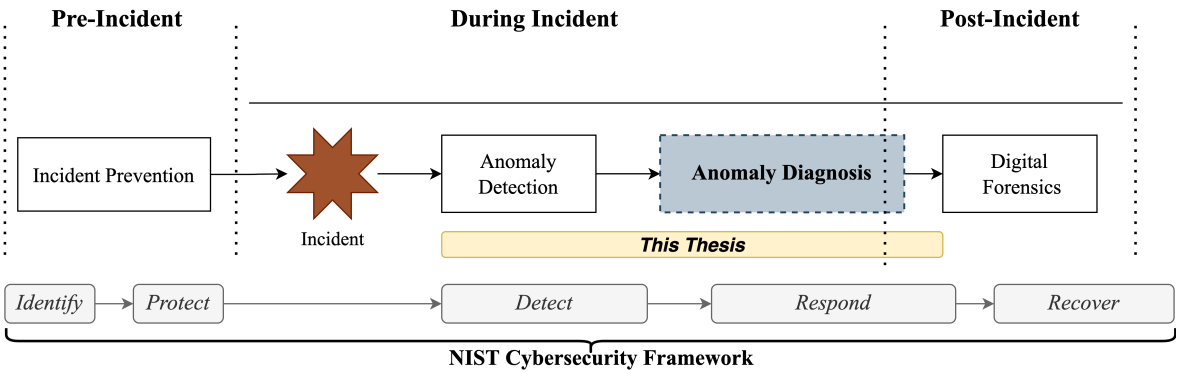


Figure 3.1: High-level Thesis Concept Placement

incident attribution of ICS attacks, often using intrusive methods of data acquisition. Consequently, there are significant challenges with real-time capture of anomalous behaviour through such methods. In this chapter, we firstly introduce an overarching PLC anomaly diagnosis framework and provide an overview of each individual component that the framework comprises that will address the limitations summarised previously.

In Chapter 2, we discussed evaluation techniques that are often employed by research to demonstrate the performance of novel anomaly detection tools or approaches. In particular, there is a growing agreement that physical ICS testbeds using real-world industrial components are critical in demonstrating that a proposed methodology does not induce adverse impacts on the system or physical process. Moreover, using industrial components, such as PLCs, that are found in real systems provides a more accurate use-case for manifesting the kinds of anomaly that have been seen in previous incidents, including ICS cyber campaigns. Hence, a core part of the research described in this thesis involves designing and implementing a physical ICS testbed that is representative of an industrial environment. The example of a water treatment facility was selected, which we refer to as the Glasgow University Liquid Purification (GULP) testbed.

In addition, a generalised ICS threat model is presented along with attack techniques, which are predicated on a review of real-world ICS cyber campaigns and existing literature. The resulting threat models are used to develop the PLC attack scenarios that are applied to the evaluation of the components of the anomaly diagnosis framework. The proposed attack scenarios are applied within the context of GULP, as an example use case, which underpins the evaluations conducted throughout this thesis.

3.2 Anomaly Diagnosis Methodology

As has been discussed in Chapter 2, a fundamental limitation of current ICS security research relates to challenges in how best to analyse and discriminate between different types of anomalous event using real-time behaviour, in a way that can assist with incident triaging. We refer to this area as anomaly diagnosis. The high-level PLC anomaly diagnosis framework that is central to this thesis is introduced in figure 3.2. The four related areas (in bold) that form the core components of the anomaly diagnosis framework are emphasised,

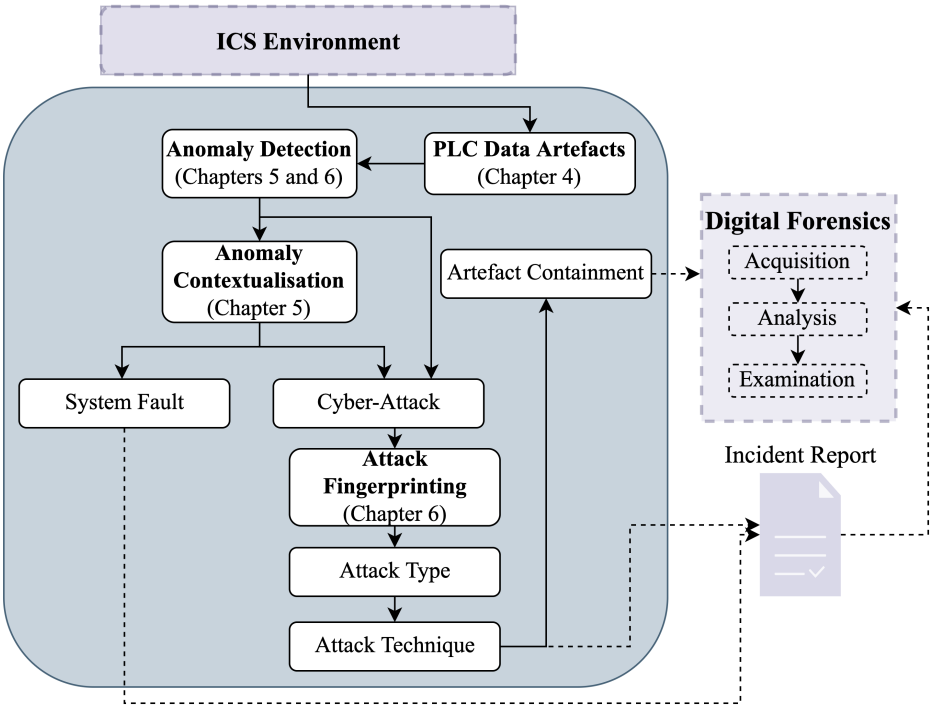


Figure 3.2: High-level PLC Anomaly Diagnosis Framework

and each of the corresponding chapters in which they are addressed is indicated. The starting point involves addressing limitations in the current understanding of *PLC data artefacts*. Many previous studies advocate the use of properties and metrics from either network data or physical quantities from field-devices to perform anomaly detection. However, this thesis proposes that the specific use of PLC data artefacts can provide greater benefits for anomaly diagnosis because of their intrinsic relationship with the underlying physical process and alterations in PLC memory manifested by different types of anomaly. Therefore, in order to proceed with the PLC anomaly diagnosis framework, a more informed understanding of the artefacts that can be acquired from PLCs, and used as feature sets in anomaly diagnosis, is required. In other words, the research described here focuses on evaluating and defining the most relevant device data artefacts is required before other components of the framework can be addressed. Chapter 4 presents this work.

In order to diagnose an event as an anomaly, anomalous behaviour firstly needs to be defined based on manifestations in system behaviour that are symptomatic of anomalies, when compared to a baseline. This thesis proposes that the use of features extracted from PLC-specific data artefacts related to the run-time operation, including PLC registers that provide discrete values, can be used to accurately model logical PLC states that are indicative of

changes represented in the physical process. Subsequently, deviations in logical states that are symptomatic of abnormal behaviour can be identified through the use of ML models that are ideally placed to detect anomalies within complex and noisy systems. In general, ML will be an integral part to enable the phases of PLC anomaly diagnosis, using properties and metrics derived from PLC data just discussed as input sources. To promote modularity of the anomaly diagnosis framework, two anomaly detection approaches will be proposed, which will involve the performance of independent anomaly diagnosis stages, if required. Moreover, we envisage that both detection algorithms will use state behaviour analysis, however, in each case, feature extraction will be performed on different PLC artefacts to generate the ML feature sets.

By further utilising PLC data artefacts that are local to the controller, *Anomaly Contextualisation* will inform the high-level categorisation of an identified anomaly, specifically as either a cyber-attack or a system fault. We argue that this is a critical stage in anomaly diagnosis as the type of anomaly that is identified will determine the subsequent incident triaging process required to restore system availability and ensure business continuity.

As mentioned, it is assumed that a PLC anomaly can result from either a cyber attack or a system fault. While system faults present an extensive challenge that justifies careful evaluation, exploring the diagnosis of the root-cause of a fault is outwith the scope of this thesis. Therefore, focusing on cyber-attacks, the final *Attack Fingerprinting* stage aims to provide an approach architecture that can use memory artefacts to establish how a PLC is being targeted in a given attack scenario. Through the real-time composition of memory fingerprints, derived by correlating the static and dynamic behaviour of PLC memory registers, the research proposes that a deeper layer of ML classification can be used to enable attack type and technique identification. Hence, attack fingerprinting can facilitate a subsequent forensic investigation by establishing a chain of evidence, and providing indicators for prioritised evidence collection from the PLC. Performing artefact containment on the acquired PLC data is imperative to maintaining the chain of evidence, however, exploring data provenance and forensic requirements for ensuring data integrity does not fall within the scope of this thesis.

3.3 Testbed Development

In this section, we introduce the GULP testbed, a water treatment facility that has been implemented as part of the work reported in this thesis. The design and architecture of GULP was guided by an assessment of the various barriers that prevent, or severely limit, evaluating the performance of active security frameworks within live ICS environments, primarily due to issues of safety, and the risks of impacting system availability and reliability.

A number of different CNI sectors and applications have been represented in ICS testbed facilities, and a recent survey identified that many testbeds replicate processes found within water treatment or electricity generation facilities [79]. Secure Water Treatment (SWaT), which was set up at the Singapour University of Technology and Design [172], is a large-scale cyber-security water treatment testbed consisting of several water treatment stages. The multiple stages of SWaT involve the use of real industrial sensors and actuators in order to mirror real world water treatment processes as closely as possible. Here, the development of GULP has taken a similar approach where real industrial components are also used to provide high levels of fidelity with real water treatment systems and processes. While SWaT produces realistic and detailed data, the publicly available datasets do not contain device artefacts related to PLC memory contents, such as control logic or buffer data, which are central to this thesis [173]. Implementing our own independent testbed enables greater experiment control over the definition of customised feature sets.

Other cyber-security research groups have taken a different approach by using simulation and virtualisation to either partly, or entirely, represent the real ICS components, networks and underlying physical processes. The HYDRA testbed is a low-cost hybrid emulation of a water treatment facility developed at the University of Roma Tre in Italy [174]. While some physical components and apparatuses are used, the PLCs and RTUs are simulated using Arduino controllers and open-source communication libraries to replicate the Modbus protocol for data transfer. However, using components found within real ICS setups is critical for performing accurate and representative evaluations, as it enables us to examine the intrinsic properties of industrial components. Therefore, GULP uses PLCs from commercial industrial vendors to monitor and control the underlying physical processes in the water treatment testbed. In addition, having real PLCs to operate GULP, provides us with information on data artefacts that could be used by PLCs in many different ICS use cases, since the PLC

models themselves, are agnostic to the ICS they support.

3.3.1 Glasgow University Liquid Purification (GULP) Testbed

To ensure that the research reported in this thesis is maximally representative, a physical testbed, comprising real industrial components and networks was developed and implemented. As discussed in Chapter 2, Section 2.2.4, physical testbeds provide particular benefits over their virtual counterparts, primarily as they use real industrial components and therefore produce data that is more representative of a real-world ICS environment. So, while there are several publicly available ICS dataset repositories, there are particular benefits in implementing our own testbed. Specifically, GULP provides control over the experiment process and specific variables that would not be possible solely through the use of existing datasets. Furthermore, using our own independent testbed enables greater scope when measuring different types of data sources. For example, public datasets primarily provide industrial network traffic flows and do not contain data regarding PLC memory artefacts. Therefore, using GULP allows us to build and implement novel feature sets that are not constrained to the specific artefacts that existing datasets comprise. Therefore, while there are several publicly available ICS dataset repositories, there are particular benefits in implementing our own testbed. Specifically, GULP provides control over the experiment process and specific variables that would not be possible solely through the use of existing datasets. Furthermore, using our own independent testbed enables greater scope when measuring different types of data sources. For example, public datasets primarily provide industrial network traffic flows and do not contain data regarding PLC memory artefacts. Therefore, using GULP allows us to build and implement novel feature sets that are not constrained to the specific artefacts that existing datasets contain. Some of the datasets that we generate from the GULP testbed that are used in parts of this thesis can be found here[1]. These contain snapshots of PLC memory registers recorded while the PLC is controlling the physical process.

[1]https://github.com/cookieprogram/PLCPrint_Datasets

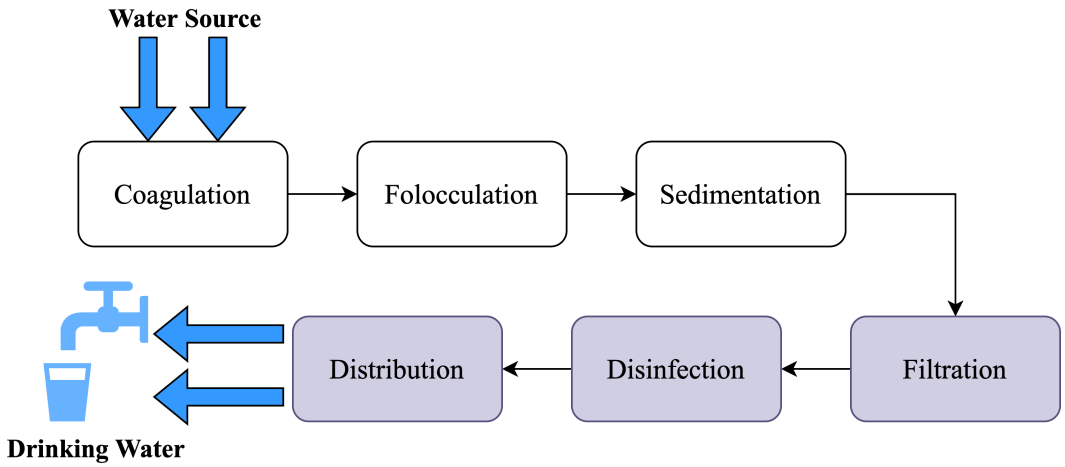


Figure 3.3: Water treatment process comprising 6 stages - highlighted stages reflect the three physical processes modelled within the GULP testbed

Water Treatment Process

The water treatment process is complex and involves several stages of physical process. Figure 3.3 illustrates the commonly used 6-stage process for water treatment adopted by government organisations such as United States Centers for Disease Control and Prevention (CDC) [175], and the United Kingdom regulator for water management, Ofwat [176]. GULP emulates a simplified version of this process, specifically the last three stages:

- **Filtration** is the first stage and represents the process of removing sediment from the water supply.
- **Disinfection** subsequently applies chemical solutions, typically chlorine, to the water supply in order to remove bacteria and virus, and then removes the chlorine before distributions.
- **Distribution** involves the processes of supplying the water to a local distribution facility, such as a water tower, and then on to domestic and commercial environments.

These three stages were selected due to the low overheads that would be incurred in their implementation, particularly regarding physical space within the laboratory environment. This selection does not compromise the research outcomes as the findings are independent of exactly which stages were modelled. During filtration, clear water that has been produced as a result of sedimentation passes through various different filters made of materials such

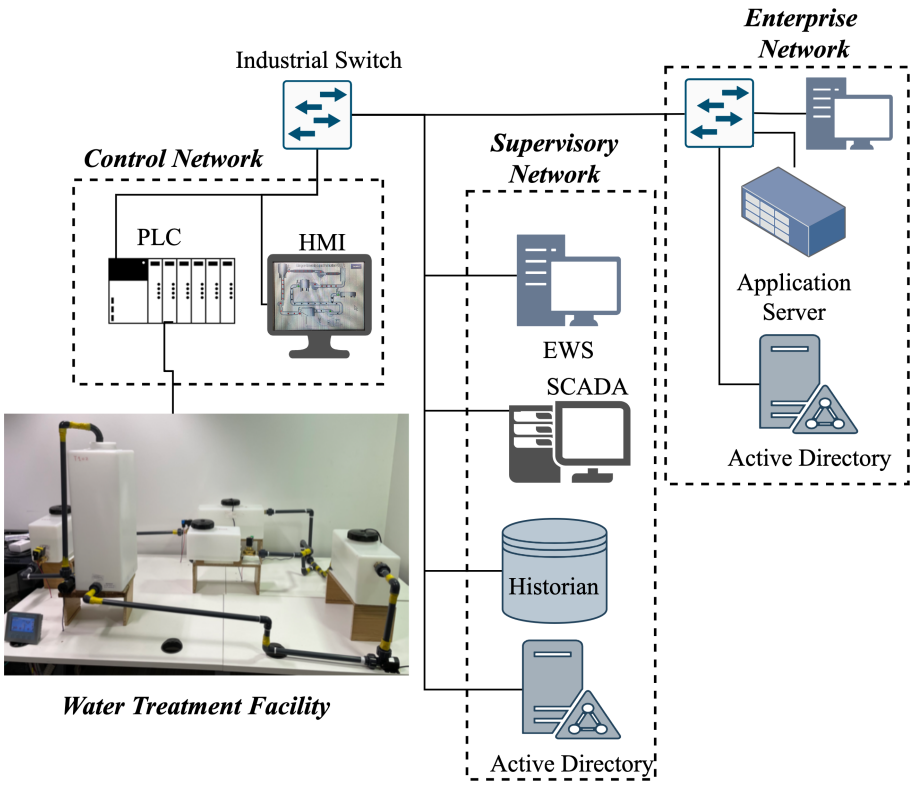


Figure 3.4: Glasgow University Liquid Purification (GULP) Testbed Architecture

as sand, gravel, and charcoal, thus providing different pore sizes for filtration. These filters remove dissolved particles and germs, such as dust, chemicals, parasites, bacteria, and viruses from the clear water. Subsequent to filtering, the water passes through a chemical disinfection process that typically uses substances such as chlorine to eliminate any parasites, bacteria, or viruses that have remained in the water at this stage. Water treatment facilities will often ensure low levels of the chemical disinfectant when the water leaves the treatment plant, in order to prevent chemical poisoning. However, small levels of chemical disinfectant will remain in the water to kills bacteria living in the pipes between the water treatment facility and the end user. Distribution is the final stage of the treatment process and involves sending clean water supplies to residential and commercial properties for people to use, usually via a pipe network and a local storage tank.

GULP Architecture

The architecture of the GULP Testbed includes three network zones; i) *Enterprise Network*, ii) *Supervisory Network*, and iii) *Control Network*, illustrated in figure 3.4. The enterprise

network represents layers four and five of the Purdue Model structure presented in Chapter 2, figure 2.2 and contains IT components used for daily corporate activities such as patch management, business intranet services, and productivity reporting. In the context of GULP, we utilise Windows 10 OS workstations to represent the enterprise service connected to a Windows Active Directory for user account management.

The supervisory network emulates layer three of the Purdue Model, primarily housing the engineering environments used to program and commission ICS components, including PLCs. The engineering workstations (EWS) contain PLC integrated development environments (IDE), where application code and network configurations can be written and uploaded to PLCs through a permanent physical Ethernet network connection into the control network. The EWS are configured running Windows 7 and Windows 10 OS, which were selected due to the system requirements of the PLC IDE software packages [177] [178]. PLC vendors often release annual upgrades for IDE software, and implementing these upgrades in real-world systems is dependent on the discretion of the organisation. Furthermore, a historian is present to record tag and variable changes from the PLC, as well as diagnostics and log data. Additionally, a SCADA system is used along with the HMI to enable control and monitoring.

Finally, layers 1 and 2 of the Purdue model are represented in the control network, containing the PLCs, HMIs, and other smart instrumentation used to control the underlying physical processes that the water treatment facility carries out. The default operation of GULP involves one PLC controlling the entire water treatment process, however a low-overhead cold-swap system has been implemented to enable the rapid interchanging of different PLC models when assessing generalisability. An HMI has been commissioned and can be used to monitor physical changes in the sensors and actuators.

ICS Vendor Selection

As vendor independence is a core aim of the proposed PLC anomaly diagnosis framework, PLC's from two vendors have been selected to monitor and control the sensors and actuators within the GULP water treatment facility. These are Siemens and Rockwell Automation (Allen-Bradley), which are the two leading PLC vendors, with a recent report claiming that Siemens have a market control of 31% while Rockwell Automation have 22%, with Mitsubishi and Schneider Electric following next at 13% and 8%, respectively [179]. The two

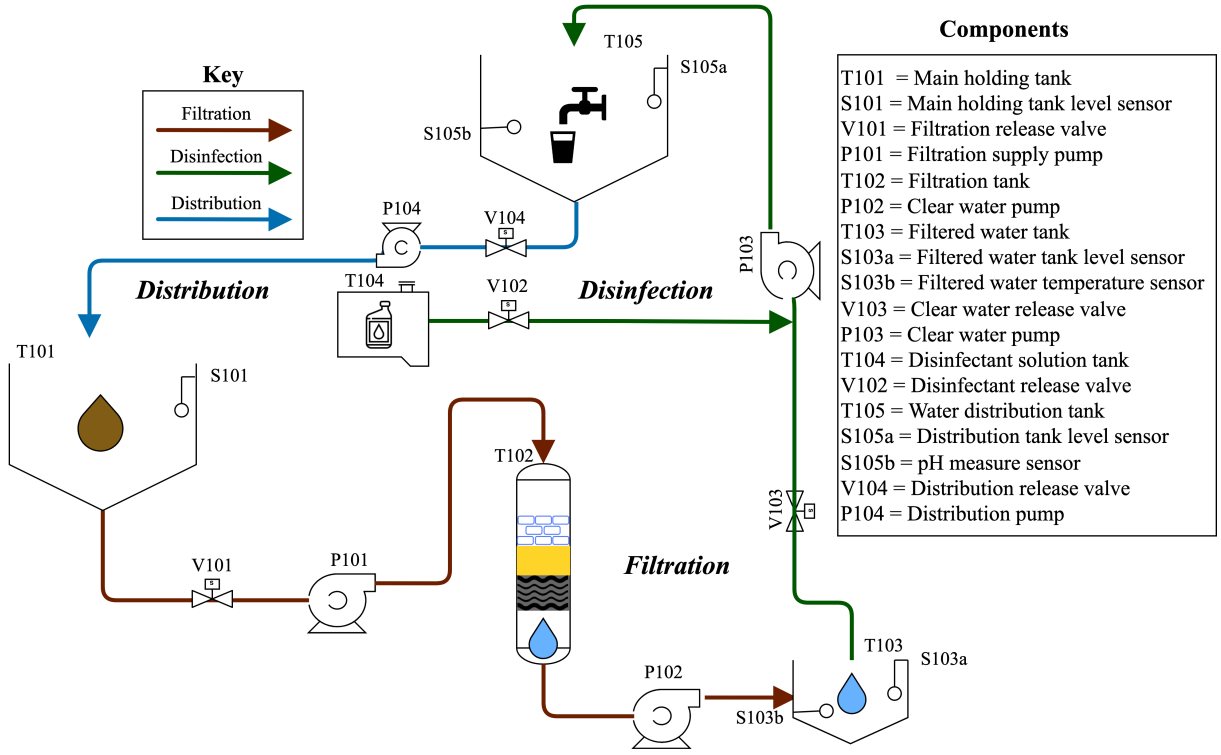


Figure 3.5: Water Treatment Facility of GULP Testbed

primary models of PLC used in the GULP control network are a Siemens S7-300 and an Allen-Bradley ControlLogix. These are part of the range of advanced controllers offered by their respective vendors, implying that they are highly representative of controllers that would be found in a range of real-world ICS, including water treatment, electrical generation, and manufacturing. In addition, two different HMIs are also used, one Siemens KTP 400 Series and one Allen-Bradley PanelView Plus 5 CE.

GULP Water Treatment Facility

The processes in the water treatment facility operate in a cyclic manner, starting with filtration, then moving to disinfection and finally completing with water distribution, before then repeating the cycle. Figure 3.5 demonstrates the process flow with reference to the individual sensors and actuators used at each stage. The type and purpose of each sensor and actuator that is used in the GULP water treatment facility, are also represented in figure 3.5. In total, 13 industrial sensors and actuators are used, together with five water tanks.

The filtration stage starts with the store tank (*T101*)providing the storage of clear water upon completion of the previous stage in the treatment process, which is not included in GULP.

Water from Tank T101 flows through a set of pipes into the main filtration tank (*T102*). The flow of water into tank T102 is controlled by a single solenoid valve (*V101*) and a water pump (*P101*). The state of valve V101 is dependent on a level sensor *S101*, in T101. When the upper bounds of S101 is triggered, the valve is opened, allowing water to flow from T101 to T102 through P101. Water then moves vertically down tank T102, which represents the filtration system, and into tank *T103* via pump *P102*, simulating the process of filtration. A thermocouple temperature sensor (*S103b*) is present in T103, which is used to generate additional physical data that can be used in subsequent datasets. When the level sensor in T103 (*S103a*) is triggered, *S103a*, V101, P101, and P102 are disengaged, ending the first stage in the water treatment facility.

Subsequently, the disinfection stage is triggered with valve *V103* and pump *P103* being activated. As water flows through into tank *T105*, an additional valve, *V102*, is opened for 15 seconds simulating the release of disinfectant chemicals from tank T104 which mix into the flowing water supply. After 15 seconds, V102 shuts preventing further disinfectant being added. A pH measure sensor (*S105b*) is contained in T105 to measure the acidity and alkalinity of the disinfected water in the tank. Tank T105 also has a level sensor (*S105a*), which when triggered closes V103 and deactivates pump P103. Thereafter, the final stage of distribution is engaged where valve *V104* is opened and water is pumped through pump *P104* from T105 back into tank T101. The water treatment process then repeats itself starting again from the filtration.

3.4 ICS Threat Models

There have been a number of high-profile cyber incidents targeting ICS that have taken place over the last 15 years ranging from cyber espionage attacks, to persistent cyber warfare campaigns; some of these were presented in Chapter 1 figure 1.1. Miller *et al.* provides a thorough review of ICS cyber attacks that are less widely-known and were reported before 2021 [13]. These attacks use a broad range of techniques that are often designed specifically to exploit the operation of ICS devices in order to maliciously impact the physical processes controlled by the ICS components, such as water valves and electrical generators.

Traditionally, PLCs and wider ICS environments were closed-off to outsiders through an

approach known as air-gaps, which physically segregated the industrial networks from the Internet-facing corporate and enterprise domains within the organisation [180]. Consequently, a security culture was created were it was believed ICS networks and underlying components could not be targeted by external threat actors. However, modern ICS architecture implementations have many more interfaces to external networks in order to improve automation, maximise productivity and enable remote control, consequently greatly reducing the applicability of the air-gap method in protecting ICS from cyber attacks [180]. Furthermore, an increasing number of ICS components are accessible through online search engines such as Shodan [2] and Censys [3], which enables attacks from lower skill-set threat groups often with limited resources [181] [182]. Moreover, many ICS-specific network protocols were designed without considering security as a critical requirement, focusing more on performance and real-time availability. Moreover, many of these insecure protocols have been connected to Internet-facing environments, consequently exposing the vulnerabilities resulting from a lack of security encryption, authentication and integrity checking [183].

In order to appeal to a wider market, an increasing number of ICS manufacturers have developed compact PLCs that often provide the core functionality of more advanced controller models but at a much lower cost. Using three leading PLC manufactures, Siemens Automation, Rockwell Allen-Bradley, and Schneider Electric, as examples, the advanced controller models offered by these vendors currently average approximately £2500 just for the PLC CPU module. In contrast, the average price for a compact model from the same three vendors is approximately £350, a significant decrease from the price of the advanced model. Of course, advanced PLC models do provide significant benefits to justify their cost, including safety controls and improved performance. However, from a security perspective, lower costing controllers easily enable threat actors to obtain real ICS equipment in order to learn and develop novel attack vectors for PLC technologies. Furthermore, an increasing number of low-cost Raspberry-Pi alternatives to real industrial components have become available, either through hardware architectures or software implementations [184]. Many of these are free, or much cheaper alternatives which can enable less advanced threat groups such as hacktivists to develop ICS malware.

Due to the vast breadth of ICS domains, attackers targeting such systems come with a range

[2] https://www.shodan.io
[3] https://censys.io

of motives, level of resources, and skill capabilities. Some adversary groups may have industrial espionage as the key objective, while others focus on compromising safety. In this section, the thesis will outline threat actor positions and attack vectors associated with ICS, and more specifically with PLCs. Threat actor models refer to the resources available and general motivations of an attack. Attack vectors are the specific techniques use by attackers to achieve certain objectives. Throughout this thesis, we build on these high-level models and attack vectors to ground the attack scenarios we use to perform realistic threats.

3.4.1 Threat Actor Positions

There are a number of threat actors that have previously been seen to target ICS and wider OT and CNI environments, and it is important to consider the specific actor to facilitate attribution and understand the threat landscape. While the capabilities of individual threat actors is not assessed within this thesis, there is an assumption that the attack scenarios we have implemented could be conducted by an adversary with varying levels of skill and resources, as long as they meet the key overarching assumptions, as well as the specific assumptions for each threat actor position, which are defined in the subsequent subsections. However, we assume that the attacker is not so highly sophisticated that they can accomplish any attack possible. For instance, one contemporary attack approach that this thesis does not explore is adversarial machine learning, since this would likely be conducted by a highly advanced adversary. Hence, we adopt the position of a given threat actor and consider two fundamental threat actor models, which are presented in figure 3.6; 1) *Insider Threat*, and, 2) *Outsider Threat*. For both threat actor position models, we make two overarching assumptions:

- The primary target device is always a PLC, although other components and equipment may be collaterally impacted;
- Other security technologies such as firewalls and intrusion prevention systems (IPS) have already circumvented or compromised though existing vulnerability exploits.

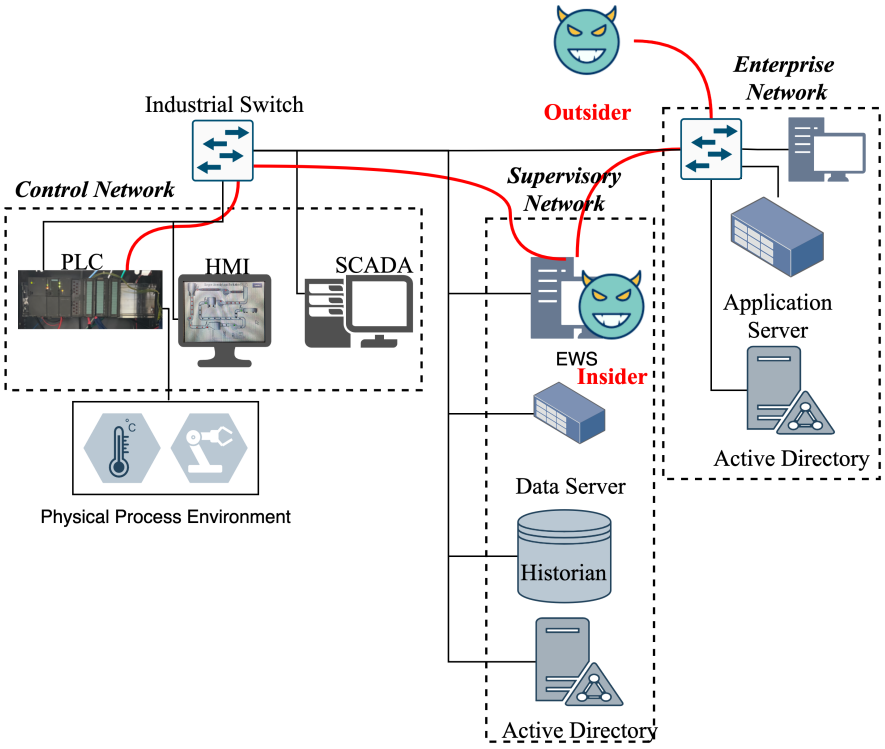


Figure 3.6: ICS threat actor positions mapped onto GULP testbed architecture

Insider Threat Model

An insider threat actor can be defined as a trusted entity, such as an employee who can either knowingly or unknowingly use their legitimate access to a system for malicious purposes [185]. Knowing insiders can take the form of a disgruntled employee who seeks revenge against a previous employer, as was the case in an early ICS cyber incident where an unhappy former employee used a two-way radio to target a sewage facility in Maroochy, Australia [186]. Unknowing insiders are often used as a component in larger adversarial campaigns, often referred to as Advanced Persistent Threats (APTs), by more developed threat actor groups. An example of this kind of insider was seen in the 2015 Ukrainian power grid attack, where attackers employed a spear-phishing campaign targeting multiple companies responsible for distributing electricity throughout Ukraine. The phishing campaign delivered emails to corporate staff and system administrators with a malicious Word document attached with important information pertaining the contemporary Geo-Political circumstances, which once opened asked victims to enable macros. Consequently, a program called BlackEnergy3 infected their machines and opened a backdoor for the hackers [187].

One additional type of insider threat that is not included within the scope of attack scenarios

that are considered in this thesis, is the act of physical sabotage. These types of attack can include, but are not limited to, cutting electrical and communication cables, tampering with power supplies, and damaging equipment. This type of attack warrants a different approach as detection and subsequent analysis of sabotage attacks is unlikely to be viable through the use of PLC data artefacts alone, such as network metrics and memory snapshots, which this thesis focuses on. Therefore, evidence collected through conventional forensic science approaches would be required to examine the physical damage, and this would be conducted from a post-incident point of view. Furthermore, such attacks are not specific to ICS and can occur within any part of an organisation's infrastructure, and therefore provide a broad range of different targets that would be too extensive to include. However, the thesis does consider the physical impacts resulting from the selected attack scenarios, with particular reference to the components included in the GULP testbed.

The key assumptions for the insider actor position can be summarised as follows:

- The attacker does have prior knowledge of the ICS physical process and specific PLC models that are being used;
- The attacker can physically access the engineering workstation, which has a permanent physical connection to the control network housing the PLC network.

Outsider Threat Model

Outsider threats are classed as adversaries who operate externally to the target organisation or system. They are likely to be highly skilled and resource-abundant threat actors, such as organised groups or nation states, as more proficient capabilities are required to circumvent external security controls. Outsider threat actors have a range of motivations including financial or political gain, physical sabotage, and even disruption to safety controls. The Triton/Trisis [12] and Florida Water Treatment [188] attacks in 2017 and 2021 respectively, serve as examples of outsider threats where attackers originated outside the target organisation's networks and infrastructure, and gained escalated remote access to the control network through the engineering workstations [13].

As the outsider threat actor starts from a position external to the control network where the PLCs are located, we define a methodology for gaining access through lateral movement

into the enterprise network and then to the supervisory network. The sub-threat model for this is demonstrated in figure 3.7. A position is first established by compromising a Windows OS system through employee credentials and unpatched system vulnerabilities, for example the *PrintNightmare* exploit that targets the Print-Spooler service used by Windows Operating Systems (OS) [189]. The attacker then pivots into the supervisory network using engineering credentials obtained through a spear-phishing campaign to then remotely access industrial equipment contained in the control network. Insecure code within services such as Print-Spooler provide mechanisms to install malicious files, such as Dynamic Link Libraries (DLLs), onto an engineering workstation via a remote server using common protocols such as File Transfer Protocol (FTP) Server Message Block (SMB). SMB is a general network protocol that enables shared access to file repositories and printers, for example.

We therefore summarise the key assumptions for the outsider actor position as follows:

- The attacker has no prior knowledge of the ICS physical process, however they have performed reconnaissance to target specific PLC models;
- The attacker can remotely access the engineering workstation, which has a permanent physical connection to the control network, through methods of lateral movement from higher layers in the network.

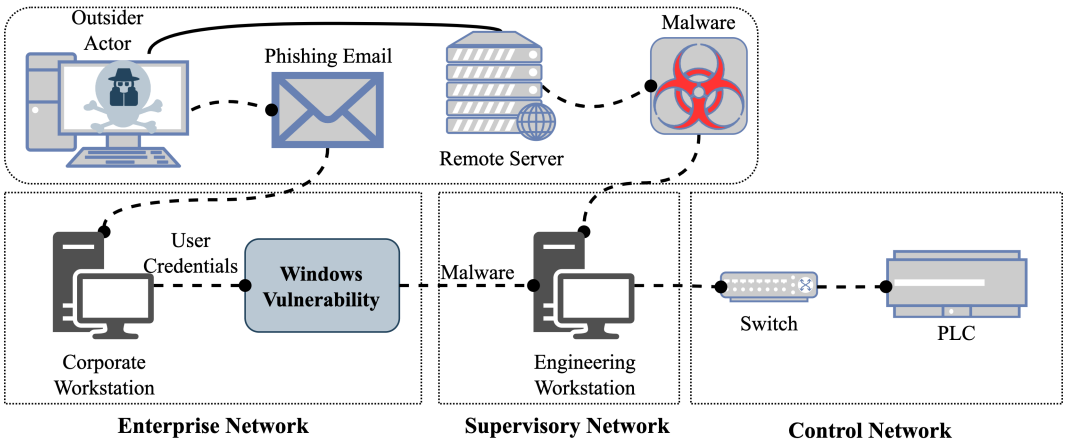


Figure 3.7: Outsider sub-threat model

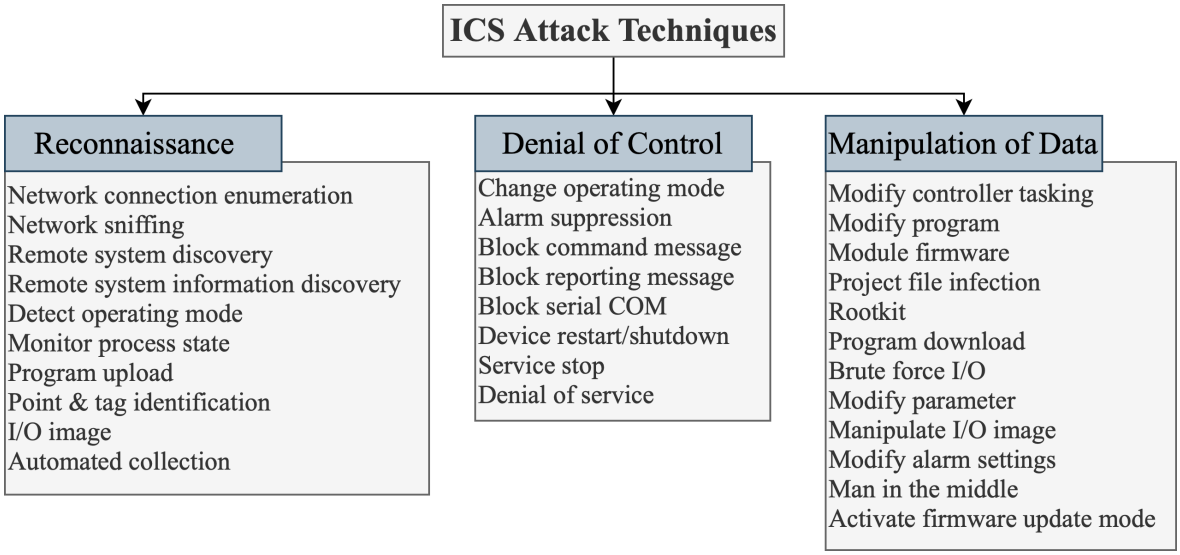


Figure 3.8: PLC attack taxonomy based on MITRE ATT&CK framework techniques

3.4.2 Attack Vectors

Attack vectors are techniques and methods used by an adversary to maliciously access or manipulate system vulnerabilities. Throughout this thesis, we use generalised attack techniques, proposed by the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework [190]. This is a curated knowledge base for cyber adversary behavior that considers attack methods at the various phases of an attack lifecycle. The MITRE ATT&CK framework has been used extensively in related research, and provides a validated foundation from which to base evaluation ICS attack scenarios on, particularly as it includes attack techniques that have been observed in real ICS cyber campaigns.

The individual tactics and techniques included in the MITRE ATT&CK framework provide a generalisable taxonomy of individual vectors as a common language for both offensive and defensive areas of cyber-security. There are three iterations of the MITRE ATT&CK framework, however within this thesis, we focus on using tactics and techniques from the ATT&CK for ICS version. As PLCs are the primary ICS component of interest within this thesis, an initial assessment was conducted to assess which techniques apply to such devices. The abbreviated taxonomy in figure 3.8, illustrates which MITRE ATT&CK techniques can be used against PLCs. We divide the techniques into three attack categories based on the adversary's objective and motivation.

Reconnaissance techniques are used to gather information on a potential target infrastruc-

ture or system, which then can be used to craft malware for specific component vendors or communication protocols [191]. These types of attack target the confidentiality of data within an ICS environment, either through collecting proprietary information on the meta-data of equipment being used, or details pertaining to the physical process. Specifically, reconnaissance attacks can involve passive or active forms of data collection, depending on what type of information the attacker is looking to acquire and whether stealth is a high objective. Many larger cyber campaigns against PLCs, and wider ICS environments, often employ initial reconnaissance attacks to determine whether the target system comprises the suitable device models, as was the case in Stuxnet [192]. In this example, the malware validated that the infected system was running Windows 7 operating system and was connected to a specific model of Siemens PLC.

Denial of Control (DoC) relates to conventional denial of service (DoS) attack models within the context of ICS. Other variants of this model have been proposed in previous work, such as Denial of Engineering attacks, which specifically relate to the disruption of the operation of PLCs [163]. DoS attack techniques targeting ICS often involve preventing components such as PLCs and Remote Terminal Units (RTUs), from performing the legitimate functions they are configured to execute, rather than altering what those functions are. DoC attack models, on the other hand, aim to disrupt the availability of ICS processes but can also target the integrity of data by intercepting command messaging or suppressing alarm notifications for ICS operators.

Manipulation of Data attack techniques are often more aggressive, in contrast to reconnaissance and DoC attacks, as they involve malware injections into a target controller to actively change functionality. Many of these techniques seek to alter the memory contents of a device, typically over network channels that are normally used for configuration and monitoring. For example, Stuxnet was primarily conducted through methods of application code injection, designed to infect the operating PLCs with malicious ladder logic, manipulating the speed of motors and ultimately altering the physical process being controlled [9]. Manipulation of data techniques have also been employed in ransomware attacks, which have emerged more recently in ICS, and which introduce an additional dimension to the impact of an attack. The initial phases of a ransomware attack involve exploiting, infecting and installing the malicious program onto a target system, all of which must circumvent any

security controls or cyber awareness that the user might have. The latter stages involving file encryption and user notification are designed to be more obvious as they require interaction from the victims of the attack, either an individual or an entire organisation. Attacks of this type have been identified in real-world scenarios, specifically the use of NotPetya malware against Ukrainian critical infrastructure [13] and LockerGoga, used in the attacks targeting Norsk Hydro in 2019 [14]. Furthermore, a proof-of-concept and vendor independent PLC ransomware was demonstrated in a study by Formby *et al.* called LogicLocker, which exploits weak authentication controls and injects a logic bomb comprising PLC ladder logic [193]. Attacks utilising techniques involving the manipulation of data are clearly focused on targeting the integrity of ICS data sources, primarily device-based data such as PLC control logic or smart instrumentation parameters. However, these types of attack also cause significant impact on the availability of an ICS by altering the kinetic processes taking place within the underlying physical and industrial equipment. Therefore, in the event of such attacks, it is likely that operators will be required to take part of, or all of, the system offline while the attack is remediated.

3.4.3 Attack Scenarios

As argued previously, there are many different attack vectors that can be used to target PLCs to achieve different goals. In this thesis, we focus on manipulation of data attacks that target PLC memory contents for two fundamental reasons. Firstly, attacks that manipulate PLC memory can be highly stealthy and result in similar impacts on the PLC. For example, an attack that injects code into the PLC application code would cause similar physical consequences yo what would be caused if individual register values were manipulated, assuming that the same part of the PLC program was targeted. Secondly, the impacts that manipulation of data attacks can have on the availability and integrity of an ICS and the physical process can be highly aggressive and cause significant damage. Therefore, such attacks present a high risk to ICS organisations. Four potential attack scenarios are defined that can be generalised to either of the threat actor positions presented in figure 3.6, across different ICS setups, and for contrasting PLC vendors. The attack scenarios are as follows:

- A_1 - *Malicious change to project file*: An attacker has access to the EWS within the supervisory network, which has a physical connection to the control network through

a switch. The attacker accesses the project files of the PLC containing the application code, which are often stored either locally or on a remote file server. Deviations to program objects, register mappings, or tag data are then achieved. The project is then compiled and downloaded to the target PLC with the malicious code embedded within.

- A_2 - *Random PLC register manipulation*: In this scenario, an attacker gains access to the supervisory network and performs brute force attacks on the PLC register states either from the EWS or a separate network node such as a laptop or Raspberry-Pi device, causing unexpected perturbations in the underlying physical process.

- A_3 - *Targeted PLC register manipulation*: A scenario similar to A_2, however, in this scenario, an attacker uses pre-existing knowledge regarding the models of PLC used and details of the underlying physical process to perform targeted attacks on PLC register values, such as inputs and outputs. Network-based commands are sent to the communication module that interfaces with the CPU of the PLC from the supervisory or control network. In turn, this forces specific register values to change causing anomalous behaviour.

- A_4 - *PLC application code injection*: An adversary uses access to an EWS to establish a connection with the PLC through its IP address and directly inject malicious control logic into the memory of the PLC. The specific code that is injected depends on the

Table 3.1: GULP Testbed Threat Use Cases on Water Treatment Facility

Attack No.	Treatment Stage	Attack Process	Physical Impacts
S_1	Filtration	*V101* and *P101* controlling filtration stage are continuously forced on	Overloaded filtration process resulting in unfiltered water
S_2	Disinfection	Activation timer for *V102* increased from 10 seconds to 60 seconds	Increased amount of disinfection product into water supply
S_3	Disinfection	Disinfection control valve *V102* continuously forced closed	No disinfection product added to water
S_4	Distribution	Water distribution valve *V104* is forced closed	Distribution valve forced shut
S_5	Distribution	Alarm thresholds for pH measure sensor *S105b* changed to higher values	Manipulating the alarm range indicated contamination leading to unsafe water being supplied

part of the physical process that the attacker is targeting.

3.4.4 Threat Model Use Cases

In this section, we will explore how the threat models and attack vectors discussed in the previous sections are applied to the GULP testbed presented in Section 3.3.1. To apply the attack techniques, described in the previous sections, to the use case of a water treatment facility, we generate five attack use cases that target the GULP testbed. These threat use cases, which are presented in Table 3.1, are informed by related research exploring the cyber risk within the water and utilities sector, and by previous real-world cyber attacks, such as the Florida water treatment incident, where attackers adjusted the levels of certain chemicals within the supply in an attempt to poison residents [188].

Moreover, the scenarios relate to the different physical process stages that are being emulated by the water treatment facility in GULP. They are generalisable between different PLC models, and can occur in isolation or as a set of combined scenarios (i.e. more than one scenario in a given attack). For each scenario, we consider the potential physical impacts resulting from the attack and how these might be represented in a real-world and large-scale water treatment environment. Furthermore, each threat use case targets different parts of the water treatment process, and therefore the sensors and actuators that are manipulated also differs. The specific components that are focused on within each use case is also outlined in table 3.1. Each threat use case can be achieved through any of the four attack scenarios (A_1 - A_4) and therefore realised from the position of either the outsider or insider threat actors, as described in Section 3.4.1.

3.5 Summary

In this chapter, the fundamental methodology and experiment setup that underpins the work in this thesis has been described. The PLC anomaly diagnosis framework introduced in Section 3.2, proposes a novel and important architecture for real-time incident response in ICS environments. Specifically, the framework comprises four related areas that will be addressed in subsequent chapters. The first area, which is explored in the next chapter,

regards data artefacts that are generated by a PLC, and specifically those that are stored in the volatile and non-volatile memory space of the PLC, and are central to generating feature sets using data that represents the relationship between the PLC and the underlying physical process being controlled. The remaining three areas of the anomaly diagnosis framework, namely Anomaly Detection, Anomaly Contextualisation, and Attack Fingerprinting, will all use metrics derived from the feature sets that focus on these PLC memory artefacts. The extent to which anomaly diagnosis can be achieved in real-time will be explored in the subsequent chapters presenting these three core areas.

Furthermore, the thesis has also introduced the Glasgow University Liquid Purification (GULP) testbed in Section 3.3.1, an ICS testbed emulating a water treatment facility use case. The GULP testbed has been designed and implemented as part of this thesis to enhance and optimise the evaluation of the central components of the anomaly diagnosis framework as described in the subsequent chapters. A physical setup was selected instead of virtual simulation to strengthen the evaluations by using industrial components, including PLCs that are used by real ICS, to improve the accuracy of the emulations. While the GULP testbed emulates physical processes related to water treatment, the overarching network architecture comprising of enterprise, supervisory and control networks, as well as the industrial equipment used, can be generalised to other ICS architectures and wider OT environments. This greatly improves the value and translational nature of the research findings presented in the subsequent chapters of this thesis. Moreover, a generalised threat model, which is influenced by the attack vectors of previous cyber incidents, was proposed in Section 3.4. In particular, the threat model depicts two threat actor positions, an insider and outsider, demonstrating how ICS attacks can be achieved from different network entry points. Furthermore, through a PLC attack technique taxonomy that is influenced by the MITRE ATT&CK framework, we propose four vendor independent PLC attack scenarios that focus on the manipulation of data artefacts that are used by the PLC to drive the underlying physical processes. The PLC attack scenarios will be employed within the evaluation of Chapter 5 and Chapter 6 which focus on anomaly contextualisation and attack fingerprinting, respectively.

Chapter 4

PLC Data Artefacts

4.1 Overview

In contrast with ICS, there are many known examples of artefact that can be acquired from IT systems. Moreover, these artefacts, and the methods of acquiring, them are are well-defined and their potential value has been explored in significant detail by previous literature. A highly detailed knowledge set from which to base the design of anomaly detection models exists for IT systems, with many studies focusing on generalisable properties relating to network traffic, for example metrics measured from TCP/IP flows [194] [195]. Other studies have used features derived from host-based artefacts for anomaly detection such as security logs, however there are challenges with generalising these across different operating systems [196], [197]. Conversely, a similar level of understanding for ICS components, such as PLCs, is missing from current expertise. Much of the existing literature that has explored PLC data artefacts use generic terminology and descriptions of data, rather than providing insights into what the specific artefacts are and how they can be used to build vendor-independent feature sets for anomaly detection and diagnosis.

The success of any anomaly detection methodology is fundamentally dependent on the data that is used as an input source. Furthermore, and from the perspective of digital forensics, understanding what data can be acquired from industrial components, including PLCs, is critical to enabling forensic experts to provide conclusions on a particular incident. In Chapter 2, we discussed the artefacts that previous studies have used to form input data sources for

anomaly detection models, with many using network communications and measurable physical properties as the primary source. In this chapter, through experimentation with real PLC devices controlling the GULP testbed, we examine the data artefacts that PLCs generate, use, and store during their operation. The key questions explored in this chapter are:

- What data artefacts can be acquired from PLCs using non-invasive acquisition techniques?
- To what extent can these artefacts be organised under artefact types?
- Which artefact types and specific artefacts are generalisable across PLC models and vendors?
- What are the features that can be extracted from these artefacts for anomaly detection and diagnosis purposes?

4.2 Artefact Acquisition

PLC data can be separated into two high-level descriptive categories: i) *Network data*, and ii) *Device (host) data*. The former includes artefacts that are generated as a product of a network communication relationship between a PLC and another device on a network, such as those discussed previously in Chapter 2 Section 2.1.2; for example, another PLC, HMI, or IED. On the other hand, device data, which is what this chapter focuses on, comprise artefacts that are generated by, or contained in the memory of, the the host device itself.

4.2.1 Experiment Setup

Three PLCs from two different manufacturers were selected for these experiments; a Siemens S7-300 315-2 DP with a 345-1 Industrial Ethernet communications processor, a Siemens S7-1500 1517-3 PN/DP CPU, and a Rockwell Allen-Bradley ControlLogix 1756-L71/B, hereon referred to as AB-CLX. Siemens and Rockwell were selected as vendors since they have the largest combined share of the global PLC market [198]. Selecting these three PLCs allowed the us to evaluate the differences in acquirable data on three dimensions:

1. PLC vendors (Siemens and Rockwell)
2. PLC models from the same vendor (Siemens S7-300 and S7-1500)
3. Different network protocols utilised by the PLCs

4.2.2 Data Acquisition Process

At the time of writing this thesis, there are no standardised acquisition methods for obtaining host data from PLCs. Several different techniques have been previously explored by the literature discussed earlier in Chapter 2, where invasive and non-invasive methods are considered. From this, we established that invasive approaches are less generalisable between different PLC vendors due to hardware specifications that are limited to only some models. Furthermore, invasive approaches involve greater overheads regarding both impacts on system performance and availability, consequently limiting their appropriateness for real-time acquisition. Therefore, it can be clearly argued that invasive techniques are less suitable for PLC anomaly detection and diagnosis than non-invasive techniques.

For the experiments in this chapter, the acquirable artefacts from PLCs are evaluated using two methods of data acquisition. Firstly, through industrial communication protocols, which is a non-invasive technique as it does not require any physical changes to the PLC, although it does involve active data capture. Therefore, such methods raise concerns regarding the computational impact that may be inflicted on PLC processing performance and quality of service (QoS) requirements, for example latency. However, there are two advantages of utilising the functionality of industrial protocol over other methods, that are particularly supportive for the anomaly diagnosis framework. Firstly, as the data request functions are typically used in ICS for data communication between different network nodes, for example PLC to PLC, or PLC to HMI communications, we can assume that utilising such functionality will not impair the reliability and availability of the system by introducing adverse affects into PLC operations. In other words, the workstation used to perform the data acquisition using industrial protocol commands can be seen as just another HMI or PLC requesting specific data artefacts, as would occur in real ICS environments. Secondly, in alternative approaches that were discussed in Chapter 2, for example hardware access through JTAG ports, industrial protocol commands enable the real-time acquisition of data during a live ICS environment,

Table 4.1: PLC Acquisition Methods.

PLC Model	Environment	Communication Library
Siemens S7-300	Totally Integrated Automation (TIA) Portal (Step 7)	Snap7 [144]
Siemens S7-1500		Snap7 and NodeS7 [140]
Allen-Bradley ControlLogix (AB-CLX)	Studio 5000 (Logix Designer)	PyLogix [143] and PyComm [142]

which is critical for automated anomaly detection and subsequent phases of the anomaly diagnosis framework, presented in the upcoming chapters of this thesis.

Different communication libraries are required depending on the PLC vendor. Based on the PLC models we use to control the water treatment facility in the GULP testbed, we use four libraries, which are outlined in table 4.1. Each communication library is then used to establish a connection to the PLC through its IP address over an Ethernet connection. The software scripts within the library are then called to issue commands over the network to the PLC by utilising the specific industrial communications protocol that the PLC is using. These commands are crafted to send data access requests using the specific PLC protocol, such as S7Comm, and parse the returned data payload. Figure 4.1 illustrates an abstracted version of this data acquisition process that can apply to all PLC models. A client such as an engineering workstation (EWS) establishes a connection session with the PLC acting as the server. The network packets are composed of PLC protocol layers that are specific to the communication protocol used by the PLC. With reference to the OSI model, these layers pro-

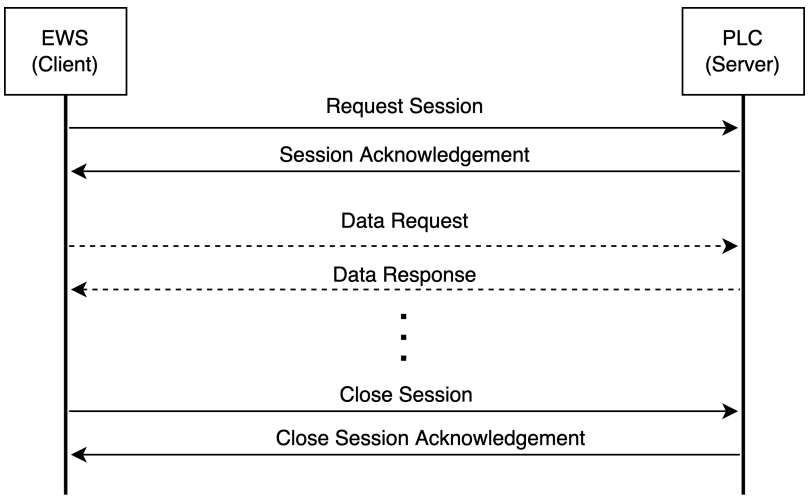


Figure 4.1: Simplified abstraction of PLC data acquisition using network communications for request and response sessions.

vide session, presentation and application functions, including specifying the type of request, such as data read or write. Assuming that the command is accepted by the PLC, a response will be provided back to the EWS containing either the requested data or a confirmation that new data has been written to the PLC. If no additional requests are made then the session will be closed. A more detailed technical description of how PLC communication protocols are used for data acquisition, specifically for Siemens and Allen-Bradley PLC models, can be found in Appendix A at the end of this thesis.

The second acquisition method we use is the PLC programming environment, which is very often vendor-specific. These environments are used for programming and configuring engineering tasks, and often contain functions for monitoring PLC behaviour and providing some basic diagnostic data. Furthermore, the programming environments typically enable PLC code debugging once the device has been deployed, and therefore provide functionality to examine PLC artefacts both in real-time with the PLC in operation, and offline. For both Siemens PLCs, we use Step7, a package of the Totally Integrated Automation (TIA) portal environment, and Studio 5000 Logix Designer for the Allen-Bradley model.

4.3 Establishing PLC Artefact Data Types

As a result of evaluating the four communication libraries and programming environments for each PLC, we define four artefact data types (ADTs) that group the identified artefacts based on their general function:

1. **Variable Content Data**: Dynamic data that is updated through the PLC's operation cycle and intrinsically linked to the underlying physical process.

2. **PLC Application Program**: The static PLC application, typically written by users/operators in one of the standardised PLC programming languages according to ISO/IEC 61131-3, such as ladder logic.

3. **PLC Meta-Data**: Data on the PLC hardware and technical specifications.

4. **Device Diagnostics and Logs**: Data and information reported and recorded through inbuilt PLC functionality or through the PLC vendor programming software environment.

The defined ADTs comprise a set of artefacts that correspond to specific artefacts of obtainable data from the PLC, illustrated in figure 4.2.

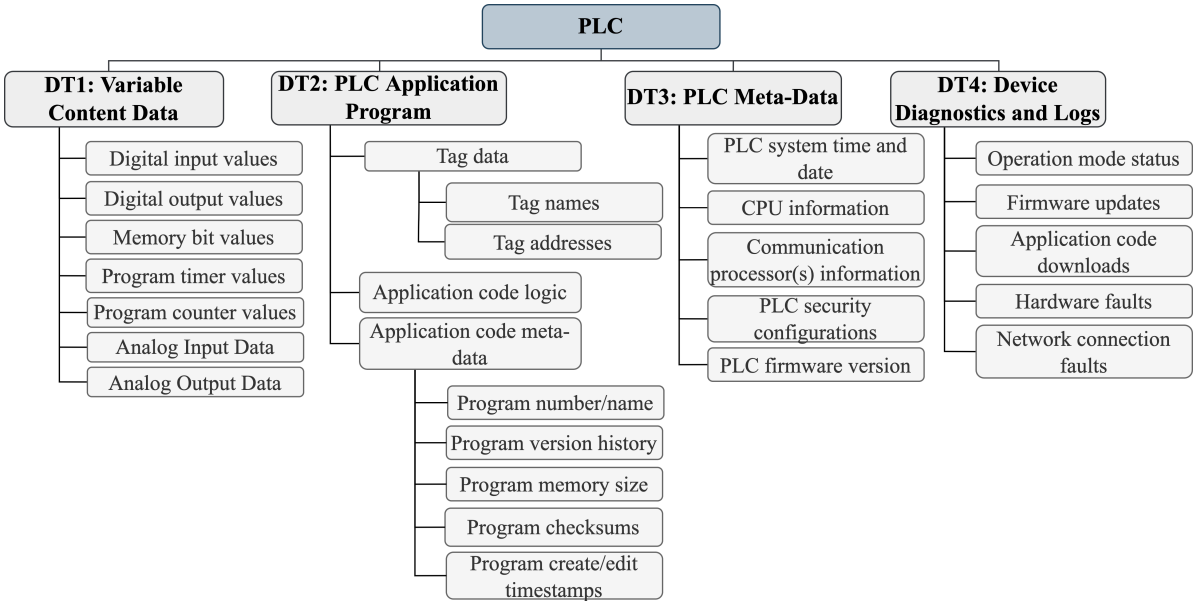


Figure 4.2: PLC Artefact Data Type (ADT) Taxonomy illustrating four core ADTs and related data artefacts.

4.3.1 ADT 1: Variable Content Data

Variable Content Data comprise artefacts that are typically volatile and dynamically aligned to the behaviour of the underlying physical process. In this case, volatility refers to the memory storage where these artefacts would be contained, and the nature of the variables that they represent. These variables represent individual registers that all PLCs use to interface with physical components, including inputs and outputs. Two types of data were observed from the acquirable artefacts. Digital registers are represented in binary as they can only be active or inactive. Conversely, continuous data was measured from analog sensors and actuators. Within this data type, both digital and analog artefacts logically represent the physical equipment and therefore can be used to model the behaviour of a given ICS using data from its PLCs. Binary vectors representing the the current states of inputs and outputs from the PLC's I/O modules, and bit memory values representing user-defined areas of memory that can be retentive or non-retentive, were obtained from the contents of PLC memory. An example binary register vector that was extracted from the acquisition of ADT 1 for the S7-300 PLC, is illustrated in figure 4.3, where individual registers relate to components within the physi-

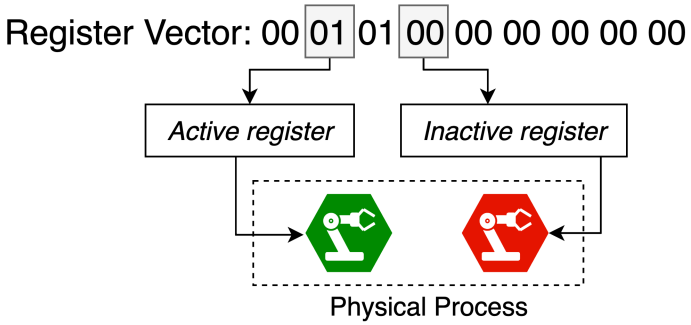


Figure 4.3: Example partial extraction of ADT 1 from S7-300 PLC demonstrating a binary register vector.

cal process. This is particularly the case for input and output variables. Similarly, input data and output data related to readings from analog components were also acquired. The former can often represent measurements of physical quantities such as temperature or pressure, whereas the latter reflects data issued by the PLC,to provide specific control parameters to analog components, for example, a speed frequency to a variable-speed drive (VSD).

Additional dynamic data was acquired in the form of counter and timer values. There are several different types of PLC timer and counter that can be used in a program, and the type used is normally dependent upon what the operational objective is of the program. However, all counters have a parameter that indicates the current count value, in real-time, that could be expected to either decrease or increase over time. Similarly, for timers, there is a time elapsed value that indicates how much time has passed since the timer was started.

4.3.2 ADT 2: PLC Application Program

The second established data type group, encompasses a variety of artefacts that relate to the PLC application program. Perhaps most importantly, is the inclusion of the application code originally written in one of the standardised PLC programming languages such as ladder logic [38]. It is important to note that the application program is not synonymous with the application code, where we consider the former to include the application code in addition to other data artefacts. Although the code could not be acquired in the original programming language format, the translated hexadecimal code was captured and examined. Developing methods to reconstruct the raw hexadecimal code into the high-level code format, such as ladder logic, is out of the scope of this thesis. However, previous research has already exam-

ined the feasibility of this, using research engineering approaches, particularly for forensic evidence reconstruction [163] [164] [199].

By extracting the hexadecimal payload contained within the response packets of each PLC, we were able to examine the structure and contents of the acquired PLC application program. As the application program could only be acquired as a copy from the memory of the PLC, rather than viewing the live code in run-time, static analysis was performed.

We start by examining the application code of the Siemens S7-300 PLC. Figure 4.4 illustrates the structure of the decompiled S7-300 PLC code structure, which we have established by performing multiple static analysis experiments involving small permutations to parts of the PLC program. Each hexadecimal payload represents a single block that forms part of the PLC application program. For each block, four areas of data can be separated out containing different artefacts. The block header contains four pieces of data with a total of 9 bytes primarily consisting of block identifiers and some meta-data. The first 4 bytes of data contains the Siemens Block ID, which is at the start of all Siemens PLC application code blocks, regardless of their type, for example a function block or data block. The ID is a constant set of four bytes with the value "70 70 01 01". The analysis of commonly used file formats such as PDFs (.pdf), JPGs (.jpg) and Windows executables (.exe), often involves the identification of file headers to confirm the file type being analysed, particularly to identify embedded malware hidden within seemingly normal files.

The block Type ID is a 1-byte value that identifies the type of program block defined, as per the naming conventions and Siemens program structure described in the previous section. The Block Load Size is the full size of the block contents in bytes, including data from block headers, timestamps, process code and block attribute sections, and represents the amount of memory that is required for the block to be loaded into the non-volatile memory of the PLC. The Timestamps section comprises two timestamps. The first identifies when the process code contained within the block itself was last changed. Changing one tag value or writing

Block Header				**Timestamps**	
Siemens Block ID 4 bytes	Block Type ID 1 byte	Block Number 2 bytes	Block Load Size 2 bytes	Code Modified TS 6 bytes	Code Modified TS 6 bytes

Process Code			**Block Attributes**				
Local Data Size 2 bytes	Process Code Size 2 bytes	Process Code n bytes	Block Author 8 bytes	Block Family 8 bytes	Block User ID 8 bytes	Block Version 1 byte	Block Checksum 2 bytes

Figure 4.4: Siemens S7-300 PLC application code structure.

a completely new control algorithm would result in changes to the timestamps, and allow an engineer or forensics investigator to understand whether a block had been altered, and then decide whether to examine further if this alteration was not expected. The second timestamp reflects the date and time at which the block was first created within the PLC programming environment. This could provide a key indicator that the application code of the PLC has been compromised and altered if the value was to be different to the known good value. The next set of data contains the process code contained within the program block, which is the code often written by the engineer, or operator, to instruct the PLC to perform the intended task within an operational process. The process code size portion indicates the size of the block's control logic in bytes. Different logical and programmatic objects have different sizes, where for example, a standard output coil is two bytes however a more complex object such as a timer is around 20 bytes. The final set of values contains some additional meta-data on the program block. All values with the exception of the checksum value, can be optionally set by the operator and are not necessarily found in all program blocks. A deconstructed example of application code extracted from the Siemens S7-300 PLC that demonstrates the aforementioned structure can be found in figure 4.5.

For the AB-CLX PLC, the structure and contents of the raw hexadecimal application program is simpler and only provides the application code, illustrated in figure 4.6 as it is seen in the CIP response packet. There are, however, several values of notable interest in understanding the structure of this packet and how the code is organised within it. The values $0x0778$ and $0x1f7a$ appear four times within the example packet in figure 4.6, which is also

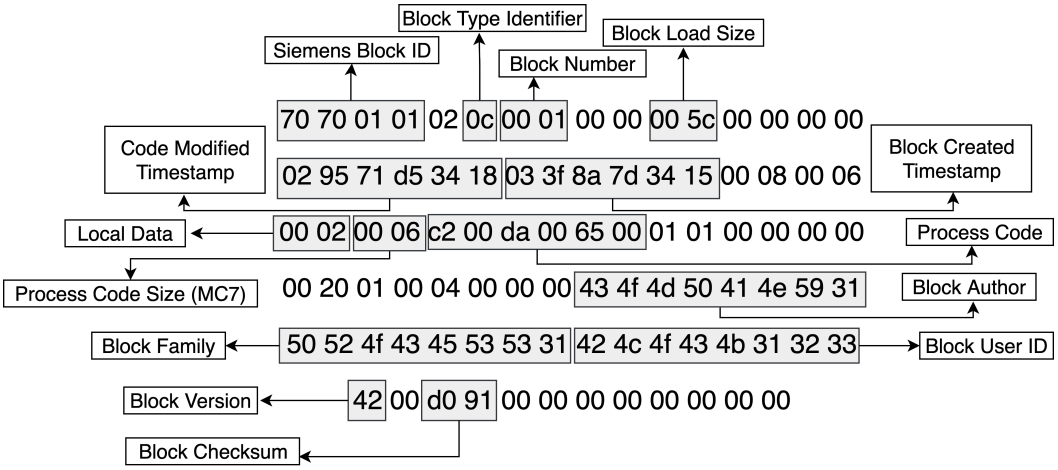


Figure 4.5: Siemens S7-300 PLC application code example of example function block extracted during data acquisition experiments.

the same number of ladder logic rungs in the application program on the PLC. Reducing or increasing the number of ladder logic rungs in the PLC application code was observed to be reflected by the number of times that these two values appear, with there always being equal numbers of rungs as values $0x0778$ and $0x1f7a$. Due to this match, it is likely that these two values represent the start and end of a rung in the ladder logic.

In addition to acquiring the raw application code logic, meta-data containing attributes of the PLC application program could also be acquired. Specifically, this included attributes such as program checksums, program version changes, and a set of timestamps that reflect the creation of, and revisions to, the underlying application code logic within the program. It was interesting to note that timestamp format did not reflect the standard structure of MAC (modified, accessed, and created) timestamps, which is commonly used by IT systems and operating systems. Instead, only timestamps reflecting edits to the code and its initial creation were present.

4.3.3 ADT 3: PLC Meta Data

In addition to collecting data on the PLC application code program, such as the raw code and tag data, a range of data regarding the PLC's hardware setup and configurations was also

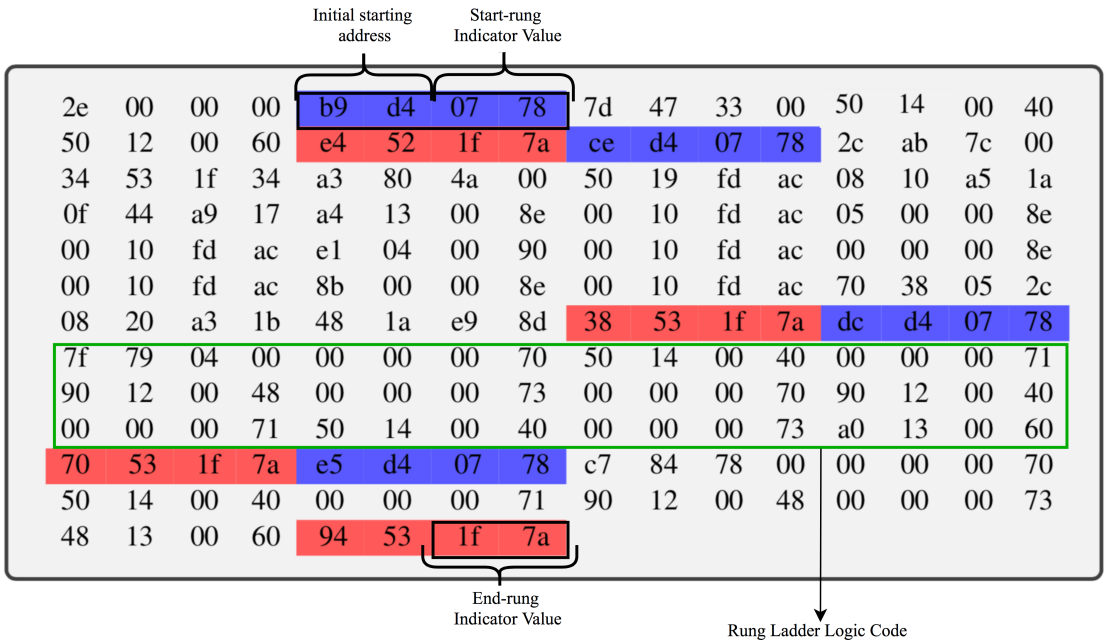


Figure 4.6: AB-CLX PLC hexadecimal application code.

acquired. This includes, but is not limited to, information on the PLC hardware modules, CPU firmware and the security configurations. The PLC CPU system data and time and information on the PLC processor model, could be acquired by using functionality provided by the Snap7 and PyLogix libraries. Establishing baselines of CPU firmware versions and security configurations may enable intrusion detection by acting as alerts should they change. Moreover, we believe that this artefact data type, and meta-data generally is suitable also for ICS network discovery solutions.

4.3.4 ADT 4: Device Diagnostics and Logs

The fourth data type that was established, encompasses the information provided by device logging and diagnostic capabilities. The acquisition of PLC device diagnostics and log data did not require the functionality of the network communications libraries. Instead, it was acquired through inherent functionality that is provided by the PLC programming environments that were installed on the EWS, namely TIA Portal and Studio 5000 for the Siemens and AB-CLX PLCs, respectively. Interestingly, none of the examined PLC logs record information relating to the underlying physical process under control, such as changes to sensors or actuators. Example log file entries from Siemens S7-300 and AB-CLX PLC models are illustrated in figure 4.7 and figure 4.8, respectively.

From the acquired log files that define ADT 4, it was concluded that generalising logs across PLC vendors is very challenging due a lack of standardised formatting and structure. Consequently, different vendors provide contrasting log information, including event types, event meta-details and event formats, as shown in table 4.2. Allen-Bradley PLC logs, for example, provide more information on interactions with the PLC's application code and project. Conversely, Siemens PLC logs record events associated with hardware changes.

The diagnostics information provided from the PLC programming environments specifies properties relating to the internal operation and memory overheads, including the PLC memory usage that is consumed by the PLC application program and related data, such as variables and tags. Furthermore, PLC cycle time measurements were able to be extracted that infer the regularity of PLC run-time behaviour, usually in the magnitude of milliseconds. The minimum and maximum cycle times are recorded primarily for engineering teams to measure the efficiency of the execution process since improved productivity can result from

```
Diagnostics buffer of the module PLC_1 [CPU 315-2 DP]

Article no. / name          Component                 Version
6ES7 315-2AH14-0AB0         Hardware                  7
- - -                      Firmware                  V 3.3.12
Firmware expansion          Firmware expansion        Boot Loader A 37.12.12

Rack:                       0
Slot:                       2

Serial number:              S C-J8K032232017



1 of 107; Event ID: 16# 4304
STOP caused by PG stop operation or by SFB 20 "STOP"
Previous operating mode: RUN
Requested operating mode: STOP (internal)
Incoming event
3/23/2021 13:33:17.443
(Coding:  16#  43 04 FF 84 00 00 00 00  00 00 00 00 21 03 23 13  33 17 44 33)
```

Figure 4.7: Siemens S7-300 Example Log.

```
remark "TSV-Controller-Log"
remark "Date = Jul-23-2018 13:21:09"
remark "Controller = 1756-L71/B"
remark "Serial-Number = 16#00C2_CE65"
remark "Revision = 24.11"
"2.0"
Record Number Time   Entry Description    User Name     Workstation Name     Factory Talk Login ID       Extended Information Change Detection Audit Value
       366    Jul-18-2018 12:38:28 Project download     KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA    Project Yiannis_test 16#773E_2876_3CD2_63B1
       367    Jul-18-2018 14:47:28 Project download     KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA    Project Yiannis_test 16#4497_580C_3D4B_532F
       368    Jul-18-2018 14:58:30 Project download     KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA    Project Yiannis_test 16#6C0A_6BCB_3D55_A898
       369    Jul-18-2018 15:36:00 Online edits modified controller program  KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA           16#DCD6_EF4A_3D73_3FE0
       370    Jul-18-2018 15:36:39 Online edits modified controller program  KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA           16#DCD6_EF4A_3D73_40D4
       371    Jul-18-2018 15:36:39 Online edits modified controller program  KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA           16#DCD6_EF4A_3D73_42B0
       372    Jul-23-2018 10:56:44 Program properties modified KITHNOS\scadalabA [ scadalabA ]    KITHNOS       KITHNOS\SCADALABA    Program nopower      16#9E8C_9CA5_56D0_9353
```

Figure 4.8: AB-CLX Example Log.

Table 4.2: Comparison of PLC Log Features.

Feature	S7-300	S7-1500	AB-CLX
PLC Meta Data	✓	✓	✓
Timestamps	✓	✓	✓
Project Download	✗	✗	✓
Project Upload	✗	✗	✗
Mode Change	✓	✓	✓
I/O Faults	✓	✓	✗
Power Failure	✓	✓	✗
Acquisition Method	IDE	IDE	SD Card

lower cycle times. From an anomaly detection perspective, as measurements including internal memory usage and cycle time are continuous variables, they would be particularly suited to regression-based approaches that model fluctuations in dependent data.

4.3.5 Other Device Artefacts

It is important to note that the previous sections discussing PLC network and device data do not comprise a superset of PLC artefacts. In fact, there are other artefacts that could potentially be manipulated in different attack scenarios and therefore would hold evidential value for both live detection, anomaly diagnosis, and post-incident forensic investigations. Perhaps most significantly is the acquisition of the PLC's firmware and bootloader code. At the time of writing, there are no publicly known methods to extract the live firmware from a PLC. PLC firmware code has been a central attack vector to inject malware into device memory [12] [10] to previous real-life ICS attacks and therefore has significance in the detection of attacks. However, the complexities regarding live firmware acquisition from a running PLC augments the development of potential solutions [85]. Therefore, it may be more pertinent to consider the meta-data of PLC firmware, such as checksums, timestamps and version values instead, which is more easily acquirable from raw firmware files.

4.3.6 ADT Generalisations

To ascertain the generalisation the of ADTs defined in table B.1, we examined the percentage of acquirable artefacts associated with each ADT, illustrated in figure 4.9 that presents average acquisition coverage across all three PLC models for each artefact data type. Both ADT 1 and ADT 4 provided high percentages demonstrating that the ability to acquire artefacts related to these ADTs through the methods of acquisition we used in our experiments, is generalisable. Thus, the use of PLC protocols for data acquisition, and to establish detection feature sets, is further supported by the range of artefacts that can be obtained.

Although a high percentage of artefacts could be obtained from all three PLC models, we observed a noticeable difference in the structure of the individually acquired data artefacts. Therefore, while it is possible to provide a degree of formalisation of the data types, different PLC vendors and models will provide this data in different formats. It should be noted though, that while certain data could not be acquired from certain PLCs, for example the PLC application code could not be acquired from the S7-1500, the methods for acquiring this data were not exhausted.

Furthermore, the coverage of artefacts under each ADT is examined to provide a more ac-

curate description of the generalisations that can be drawn from figure 4.9. The comparison provided by table 4.3, presents which artefacts within the four ADTs could be identified and obtained from the three PLC models that were evaluated. For ADT 1, all the identified artefacts could be acquired from each of the evaluated PLC models, demonstrating that this data type is highly generalisable across different PLC vendors and models.

ADT 2, which notably includes the application code itself, could only be acquired from the S7-300 and AB-CLX as the necessary functionality required to retrieve this from the S7-1500 PLC was not identified. We explored implementing a similar approach to the AB-CLX PLC where the PyLogix library was extended to include application code acquisition, however vendor proprietary encryption controls prevented extensive reverse engineering of the S7Comm-Plus protocol needed to craft the required request packets that could be sent to the PLC over the network. The limitations presented by the S7-1500 evaluations highlight the challenges that proprietary technologies pose to generalisable solutions for security. However, we still consider this artefact to be generalisable as it can be assumed every commercial PLC in operation will contain application code because it is an internationally recognised standard of PLC programming. Functionality provided by the Snap7 library enabled the acquisition of individual application code structures, such as function blocks from the S7-300 and routines from the AB-CLX PLC.

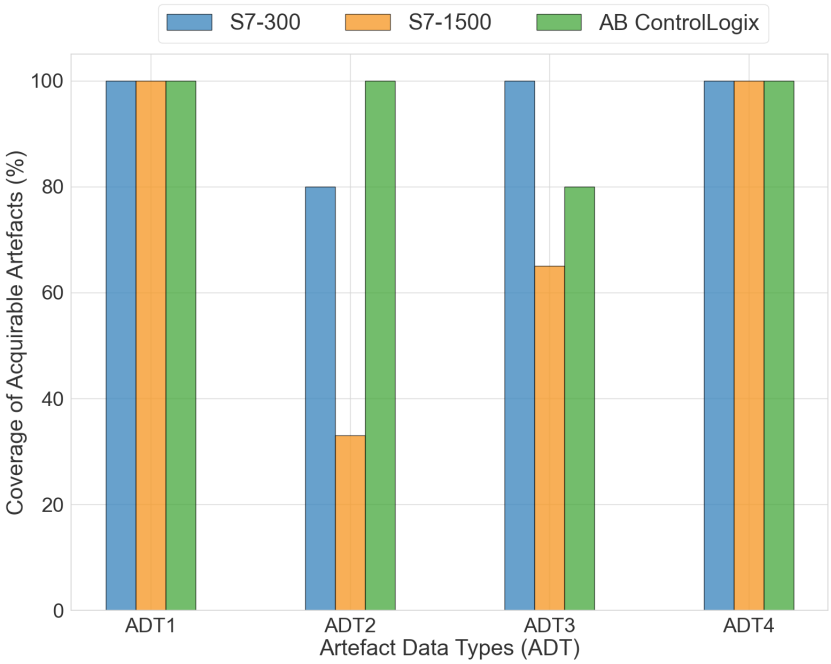


Figure 4.9: Coverage in percent of PLC artefacts between examined PLCs

Table 4.3: Comparison of ADT artefact coverage between PLC models

ADT	Artefact	S7-300	S7-1500	AB-CLX
ADT 1	Digital input values	✓	✓	✓
	Digital output values	✓	✓	✓
	Bit memory values	✓	✓	✓
	Program timer values	✓	✓	✓
	Program counter values	✓	✓	✓
	Analog input data	✓	✓	✓
	Analog output data	✓	✓	✓
ADT 2	Tag Data	✗	✗	✓
	Application logic code	✓	✗	✓
	Application program meta-data	✓	✓	✓
ADT 3	PLC system data and time	✓	✓	✓
	CPU module information	✓	✓	✓
	Communications processor information	✓	✗	✓
	PLC security configurations	✓	✗	✗
	PLC firmware version	✓	✓	✓
ADT 4	Operation mode status	✓	✓	✓
	Firmware updates	✓	✓	✓
	Application program downloads	✓	✓	✓
	Hardware/network faults	✓	✓	✓

Regarding ADT 3, which encompasses artefacts related to PLC meta-data, not all artefacts were available from all of the PLCs. Most notably for the S7-300 PLC perhaps, is that a full list of application program blocks and set security configurations could be acquired for analysis, including details about read/write access to the PLC's CPU and application code represented by a numerical value indicating the enabled security controls. These specific artefacts in ADT 3 could not be acquired from the S7-1500 or AB-CLX PLCs.

4.3.7 Time Consumption and System Impact

Requirements mandated by the safety and mission-critical processes that ICS control, partly rely on continuous availability of system resources including, but not limited to, network bandwidth and computational processing resources. It is therefore crucial that any data acquisition or monitoring processes minimise the use of these critical resources. Hence, we examine the system impact of acquiring ADTs, specifically exploring the potential impacts induced by the acquisition of ADT 1, ADT 2, and ADT 3 as these are acquired during runtime while the PLC is controlling the physical process, and therefore any adverse effect is

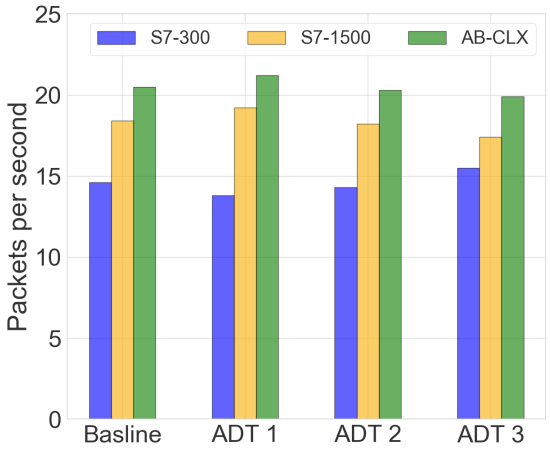
likely to have greater consequences. We consider the impacts on variables defined as follows:

1. *PLC Scan Cycle Time* – The PLC scan, often referred to as cyclic program execution, is the method by which PLCs read and write variables, and execute the application program. The scan cycle time is the time taken in milliseconds (ms) for the PLC to complete one cycle including any interrupts induced by system functions within the application code or the environmental factors such as hardware errors. The scan cycle time is measured automatically by the PLC programming software environment while connected to the PLC through a network session.

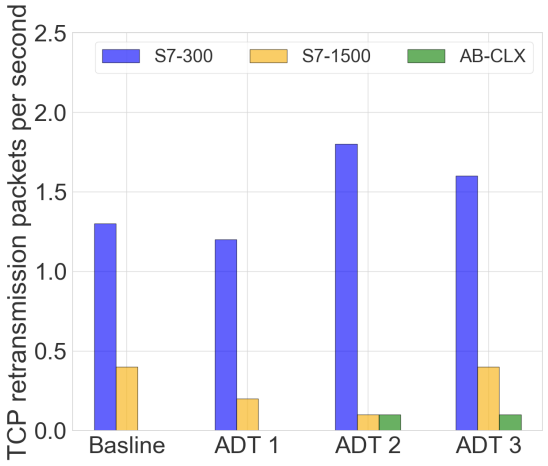
2. *TCP Retransmissions* – As the industrial protocols of the PLC models that are being evaluated utilise TCP, we measured TCP retransmission to examine whether the acquisition of ADTs increases rate of packet loss between the PLC and HMI devices, which is measured in average packets per second. Specifically, when an outbound TCP segment is handed down to an IP device with no response acknowledgment for the data before the TCP automatic timer expires, the segment is retransmitted. The communication between the PLC and HMI includes the continuous transfer of variable data from the PLC so that interface elements on the HMI screens, such as sensor readings, can be updated for the ICS operator. It is important that these communications remain uninterrupted to ensure that real-time data is provided for system diagnostics purposes.

The PLC scan cycle time was measured through the programming environment for each PLC. Specifically, this was TIA Portal for the two Siemens PLCs, and Studio 5000 for the AB-CLX model. From the PLC scan cycle times obtained, we noted that the acquisition of each ADT does not have any observable impact on the scan cycle time, with all PLCs remaining at their baseline time consumption of 1ms for the S7-300 and AB-CLX, and 2ms for the S7-1500. As there was no deviation from the baseline, we have not included these metrics in a figure.

From network impact analysis conducted through Wireshark for all three PLC models, the acquisition of all three ADTs had limited induced effects on the network, as illustrated by the average TCP retransmission between the PLC and the HMI in figure 4.10. The total average packets per second between the PLC and HMI remained consistent with the baseline



(a) PLC - HMI network communication in packets per second.



(b) PLC - HMI TCP retransmission measurements in packets per second.

Figure 4.10: Network impacts induced by ADT acquisition for all PLCs, specifically using industrial communications between PLC and HMI as a use case.

operation, which involved no ADT acquisition processes, for all three PLC models. Furthermore, as indicated in figure 4.10b, minimal fluctuations in TCP retransmission levels also occurred, demonstrating further, that ADT acquisition is unlikely to impact the communication between PLCs and other ICS devices on the network.

Finally, the time consumption required for ADT acquisition was evaluated. Generally, this is dependent on the amount of data being acquired from the PLC, which can vary between applications. We observed that ADT 2 induces the highest acqusition time consumption, where for the S7-300 PLC, a maximum time of 4 seconds was identified. As ADT 2 includes the highest volumes of data to be transferred from the PLC to the EWS, this is to be expected. Moreover, the acquisition of ADT 1 was shown to have the shortest time consumption of 0.4 seconds.

4.4 Summary

In this chapter, we have investigated what data artefacts are acquirable from PLCs, and how these artefacts can be categorised under high-level artefact data types (ADTs). From the acquired artefacts, we defined four ADTs which are summarised as follows:

1. Variable Content Data: Comprises artefacts that are dynamic, volatile, and aligned with the changing of the underlying physical process.

2. PLC Application Program: Artefacts that are static and unchanging during normal continuous operation of the PLC, and which are related to the control logic.

3. PLC Meta Data: This ADT comprises artefacts that are descriptive of the PLC model and hardware modules.

4. Device Diagnostics and Logs: Established as artefacts that are generated by the inherent diagnostics capabilities of the PLC, and typically relate to the PLC's operational behaviour.

The categorisation of artefacts within ADTs depends on two key properties. Firstly, the information that is provided by an artefact is considered, and specifically how the data relates to the operation of the PLC. We have shown that generally ADT 1 and ADT 2 artefacts have the strongest relationships with the dynamics of the underlying physical process. Secondly, the volatility of artefacts is also considered. For example, dynamic data that is mapped onto PLC variables through the memory registers, as defined in ADT 1, changes over time as the physical process is controlled. Furthermore, this research has demonstrated that the data access functionality inherent within industrial communication protocols enables ADT acquisition from multiple PLC vendors, highlighting that the approach is generalisable. The PLC ADT taxonomy and the artefacts that have been presented in this chapter, are used to implement the methodological approaches applied to the anomaly diagnosis framework components, proposed and explored in the subsequent chapters of this thesis. Therefore, references to ADTs will often be made.

Chapter 5

PLC Anomaly Contextualisation

5.1 Overview

Having examined the data artefacts that can be obtained specifically from PLCs in the previous chapter, we now move to examine how such artefacts can be used to enable the first two stages of the PLC anomaly diagnosis framework. In contrast with conventional IT systems, the operational and safety-critical importance of PLCs introduces challenges for ICS operators when empirically determining whether an anomalous event is a cyber-attack caused by malicious intent, or a system fault resulting from a hardware failure or software error. Fundamentally, both occurrences can present themselves similarly and cause indistinguishable outputs on the underlying physical process the PLC is controlling. Moreover, existing anomaly detection techniques specific to PLCs, that were discussed in Chapter 2, primarily give indication of an incident rather than categorising what the incident is. In this chapter, we introduce an approach that uses the novel synthesis of PLC artefacts for the diagnosis of anomalies in order to classify them as either a cyber-attack or a system fault. Specifically, a PLC anomaly diagnosis framework defined through a two-stage state-based detection and contextualisation approach based on novelty detection, is introduced. A combination of semi-supervised and supervised learning techniques are employed to automate the detection and contextualisation processes, respectively.

The key contributions of this chapter can be summarised as follows:

- Presenting a register state-based anomaly detection approach to model PLC operation

behaviour represented by physical and logical manifestations

- Proposing a synthesised feature set comprising PLC run-time data that is vendor-independent, giving particular focus to PLC device logs metrics
- Introducing a novel and vendor-independent contextualisation model for PLCs anomalies that distinguishes malicious intent from a system fault through the integration of multiple PLC run-time data sources.
- Evaluating key performance metrics, including F1, accuracy, and feature importance, for both the detection and contextualisation of PLC anomalies through emulated scenarios conducted over the GULP testbed, as presented in Chapter 3.

Table 5.1: Key notation used in Chapter 5

Notation	Meaning
a	An anomalous state that occurs in the PLC operation
a^{atk}	An anomaly that is caused by the realisation of a given cyber attack
a^{flt}	An anomaly that is caused by the realisation of a given system fault
$a(t_i)$	Timestamp of detected anomaly event
s	A PLC state that is represented by a unique set of discrete variables
S^{ph}	A physical state represented by a unique set of active and inactive sensors and actuators
S^{lo}	A logical state represented by a unique set of active and inactive PLC registers
X^{ac}	A set of active actuators
X^{in}	A set of inactive actuators
Y^{ac}	A set of active sensors
Y^{in}	A set of inactive sensors
R	A complete set of registers used by a PLC
R^{ac}	A set of active PLC registers
R^{in}	A set of inactive PLC registers
I	PLC input registers
Q	PLC output registers
H	PLC holding registers
P	PLC timer registers
C	PLC counter registers
U	A set of network packets from the ICS
U^{len}	Average packet length of a given dataset Z_k
U^{rak}	Amount of read acknowledgement packets from PLC in a given dataset Z_k
U^{wak}	Amount of write acknowledgement packets from PLC in a given dataset Z_k
L	A set of PLC device log events
L^{tot}	Total amount of PLC device log events in a given dataset Z_k
V^{bin}	Binary vector of PLC registers
V^{dec}	Decimal representation of binary vector V^{bin}

In table 5.1, we provide the key notation and abbreviations used in this chapter. Some of these notations are used in Chapter 6 as well.

5.2 PLC State Formulation

The operation of an ICS, and PLCs more specifically, can be described using Deterministic Finite State Machines (DFSM) which model the evolution of states of a system. A PLC has a finite set of defined states S, which are dependent upon how the PLC has been programmed and the physical process it controls. We define a set of states as:

$$S\{s_1, s_2, \ldots, s_n\} n \in \mathbb{N} \tag{5.1}$$

The number of states a given PLC can have is dependent on the physical process it is controlling. For example, a PLC controlling only one light will have a smaller number of states than one controlling a water treatment process, where each state involves the manipulation of physical sensors and actuators, such as pressure sensors, valves, and pumps. At a given point in time, the physical process under control can be in a particular state (s_n). A state has a physical manifestation S^{ph} comprising a set of active (ac) and inactive (in) sensors (X) and actuators (Y) connected to the PLC:

$$S^{ph}\{\frac{X^{ac}}{X^{in}}\} + \{\frac{Y^{ac}}{Y^{in}}\}. \tag{5.2}$$

In addition, each PLC state also has a logical manifestation S^{lo} that represents a unique set of registers (*R*) within the PLC's memory; $S^{lo} = \{r_1, r_2, \ldots, r_k\}$. Therefore, we define this logical manifestation as:

$$S^{lo}\{r_1, r_2, r_k\} \subset S^{lo}\frac{x^{ac}}{x^{in}} + \frac{y^{ac}}{y^{in}} \tag{5.3}$$

Since we fundamentally focus on the data and logical manifestation of PLC states, a given logical state $s_n^{l\,o}$ is herein abbreviated to just s_n.

5.2.1 PLC Registers

We show that PLCs use different register areas to generate variables that can change during the operation of the physical process under control. Building on from the Artefact Data Types (ADT) defined in Chapter 4, PLC register artefacts exist under ADT1 (Variable Content Data), meaning that they are volatile and dynamic. Motivated by the idea that PLC register data is used to influence the physical process and the physical process can influence register values, a snapshot of current PLC register values at a given time can be constructed. We define R as the super-set of registers that a specific PLC uses from five common register areas that are used for basic PLC operations: Inputs I, Outputs Q, Holdings H, Timers P and Counters C; $R\{I, Q, H, P, C\}$. Each register area has a predefined finite number of registers assigned to it, which is determined primarily by the PLC model. We define each register area as:

$$I\{i_1, i_2, \ldots, i_v\}, Q\{q_1, q_2, \ldots, q_w\}, H\{h_1, h_2, \ldots, h_x\}, P\{p_1, p_2, \ldots, p_y\}, C\{c_1, c_2, \ldots, c_z\} \tag{5.4}$$

All register sets I, Q, H, P, C have a set of active (ac) and inactive (in) registers, where the full set of registers for each register area is a concatenation ($\frown$) of ac and in, for example:

$$I\{I^{ac\frown}I^{in}\}, Q\{Q^{ac\frown}Q^{in}\} \ldots \tag{5.5}$$

Therefore, we define the logical manifestation of each PLC state (s_n) at time point t as:

$$s_n^{l\ o} = \frac{I\frown Q\frown H\frown P\frown C}{t} \tag{5.6}$$

which represents the full set of of ac and in for a given PLC at state s_n.

5.2.2 Anomalous PLC Behaviour

Within Chapter 3, we discussed different cyber attack vectors and techniques that have previously been seen to target PLC, and more generally ICS. However, where cyber attacks are

purposefully malevolent and malicious, unintentional system faults that occur can also result in the ICS producing similar anomalous behaviour. PLC anomalies can be defined as one or more states that are unexpected or do not exist within a set of baseline states S. We assume that a given PLC anomaly α can be either a cyber attack α^{atk} or a system fault α^{flt} and that the anomaly is reflected within the composition of s_n. We denote A as a set of anomalies that can contain A_j anomalous states:

$$A\{a_1, a_2, \ldots, a_j\}j \in \mathbb{N} \tag{5.7}$$

The time at which an anomaly is detected is denoted as $a(t_i)$, providing a timestamp of anomaly identification. In this thesis, we define a fault that occurs within an ICS as one or more non-malicious anomalous PLC state(s) resulting from either a software error or hardware failure. From existing literature, we determine that there is no comprehensive list of faults that can be used, however there are commonly reported ones discussed in several PLC technical sources, which are summarised below [200], [201], [202].

- **I/O System Fault** which can occur as a result of a coding error within the PLC's application code, or from a hardware fault in one or more of the I/O modules of the PLC.
- **Power Supply Failure** result from a loss of power to the PLC itself and, if available, an additional failure of a redundancy power source.
- **Communication Disruption or Memory Corruption** can be induced from physical phenomena. For example, the memory of a PLC can become corrupted by frequency interference and disruptions of power, which would typically place the PLC into STOP mode and indicate one or more error(s) on the PLC modules.

ICS engineers and operators who are often the front-line personnel with PLCs, are likely to treat anomalous behaviour within the ICS as indicators of a fault rather than warnings of an evolving cyber incident, due to their training being primarily in engineering concepts rather than cyber-security [92]. However, it has been argued previously that this attitude reflects misplaced complacency [203]. Due to the increased cyber threat against PLCs due to modern ICS architectures often being networked with Internet-facing domains, anomalies may be

induced by malicious cyber attacks instead of system faults. Furthermore, ICS components such as PLCs that appear to be malfunctioning due to a hardware fault could be physically replaced in order to maintain continuous operation and system availability. Consequently, any volatile data would be lost, and it is highly likely that any other acquirable data or meta-data would be forensically inadmissible. The ambiguity of event data generated from logging capabilities can further augment the challenge of anomaly analysis.

5.3 PLC Anomaly Contextualisation

To investigate how PLC run-time data can be used to not only identify anomalous behaviour but also to diagnose the event, we propose the PLC Anomaly Contextualisation model. The approach first involves the acquisition and generation of a PLC run-time state dataset, which performs the synthesis of different data artefacts produced by a PLC at run-time. Subsequently, the run-time dataset is used to detect and contextualise PLC behavioural anomalies based on the assumption of PLC states, as defined in Section 5.2. Firstly, a novelty detection algorithm is implemented to detect a potential anomaly in real-time by identifying perturbations in PLC register values. Detected anomalies are then examined in greater detail to contextualise the anomaly as either a system fault or a cyber attack based on deviations between baseline and test PLC run-time state datasets.

5.3.1 PLC Runtime Data Formulation

When detecting attacks targeting PLCs, it follows that data that correlates with changes in the physical process should be utilised for proactive detection. We can determine if a run-time data source correlates with the physical process if: (i) its value influences the physical process, or (ii) the physical process influences its value. Where data does not correlate with the physical process, we show how it can be used to provide a categorisation of identified anomalies and facilitate a rapid mitigation response.

The generation of a PLC run-time state dataset is presented in algorithm 1, where a register snapshot R comprising a set of PLC register values, a collection of network communication packets C, and a PLC device log file L, that contains a total of x log events, are formulated.

Algorithm 1 Generation of PLC Run-time State

Input: Register Snapshot *R* = $\{I_v, Q_w, H_x, P_y, C_z\}$, Log File Events $L = \{log_1, ..., log_m\}$, Network Packets $U = \{pkt_1, ..., pkt_j\}$
Output: PLC Run-time State

```
for vector ∈ R do
    if vector instance of T then
        for timer in vector do
            if timer is not equal to previous timer then
                timer state = True;
                Store timer value as previous
            end if
        end for
    end if
    if vector instance of C then
        for counter in vector do
            if counter is not equal to previous counter then
                counter state = True
                Store counter value as previous
            end if
        end for
    end if
    vector value = encode vector as decimal
end for
for packet ∈ U do
    if packet instance of write acknowledgment then
        write acknowledgment count increment by 1
    end if
    if packet instance of read acknowledgment then
        read acknowledgment count increment by 1
    end if
    accumulate packet length
end for
mean packet length = total packet length / U
total logs = L
```

Three linked PLC data sources were identified under our definition of PLC run-time data. As these data sources are updated during PLC run-time, a set of PLC register values (*R*), network communications traffic properties (U) and device log data (L) can be defined collectively as (Z), linked through a common timestamp (t). Figure 5.1 illustrates the process of generating a PLC run-time dataset comprising the three artefact types R, U and L:

$$Z = t_i\{R, U, L\} \tag{5.8}$$

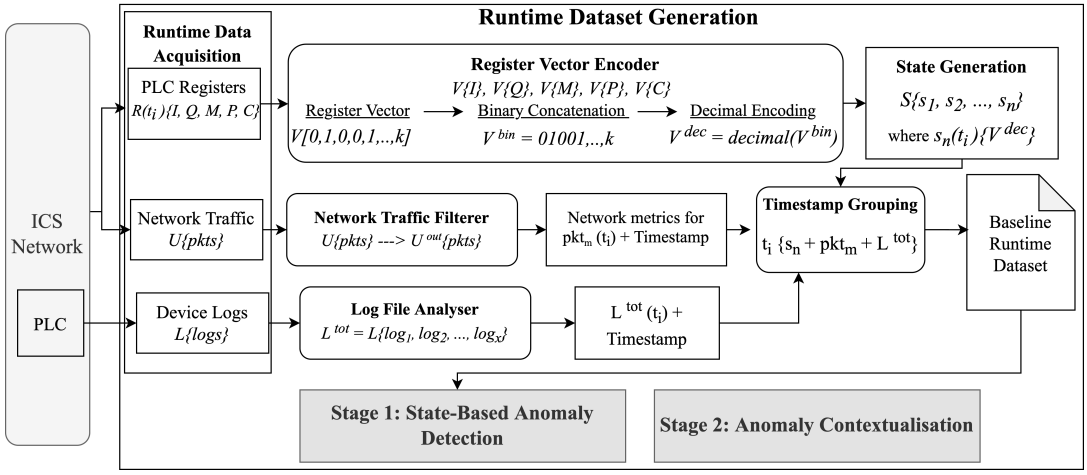


Figure 5.1: PLC run-time dataset generation within Anomaly Contextualisation Model

PLC Register Snapshot

An active approach is taken to acquiring PLC register values by querying the register areas of the PLC through crafted network packets, that request this type of data by using the functionality provided by open-source communication libraries. Table 5.2 summarises the libraries used and the different network communication properties between the two PLC models explored in our evaluation, which is presented later in Section 5.4. Specifically, we utilise the same functionality used to acquire ADT1 presented in Chapter 4. ICS historian logging systems, which are a common component in ICS, use a similar approach for retrieving and monitoring PLC process data, and therefore an active acquisition method was deemed appropriate. The data returned by querying the dynamic register states is in a raw non-unified form due to the different types of programming construct used in ladder logic. The acquired PLC register values are pre-processed by converting each value into a binary representation given some condition related to its behaviour, as described in Section 5.2 and defined by lines 1 to 15 in algorithm 1. Each register is assigned 0 if inactive and 1 if active. Converting the register values to a binary representation enables analogous analysis of both discrete and continuous data types. Counter and timer registers are not naturally represented as binary values as they represent continuous data. Therefore, in order to process a timer or counter register, its current value is compared to its value in the previous run-time state. If the value has changed, then the timer, or counter object, is set to be *True* for that specific PLC state. The pre-processing conditions for the PLC registers are outlined as follows:

- **Inputs, Outputs and Holding Registers**: active = 1, inactive = 0
- **Timers**: incrementing/decrementing (active) = 1, stationary (inactive) = 0
- **Counters**: incremented/decremented (active) = 1, no change (inactive) = 0

As a result, five binary vectors are returned representing a snapshot of PLC registers, with a vector for each register area, such as I or Q. Let's take I as an example where $I\{i_1, i_2, \ldots, i_v\}$ and $i_v = i^{ac} \vee i^{in}$ depending on whether the value of i_v is 1 or 0, respectively. Subsequently, a binary vector is generated; in the case of I this is of length v where $v \geq 0$ and is equal to the number of input registers used by the PLC. A PLC using eight input registers, for example, could provide a vector of: $t_i[1, 0, 0, 1, 0, 0, 0, 0]$, which represents $i_0 \wedge i_3 \in i^{ac}$ and $i_1 \wedge i_2 \wedge i_4 \wedge i_5 \wedge i_6 \wedge i_7 \in i^{in}$. In other words, registers i_1 and i_3 are active and the remaining six i registers are inactive at time point t_i.

In the example above, it is noted that the size of the resulting vector is not typical of a real PLC where large scale industrial systems may have upwards of 50 sensors and actuators connected to a single PLC [172]. Larger vector sizes would significantly increase the number of features that would be used in a given dataset. Therefore, we apply a decimal conversion encoder to the binary vectors in order to improve the scalability of the anomaly detection approach by reducing the dimentionality of the feature set without removing the fundamental properties of the registers. Each vector element is concatenated to form a k-bit binary value where $k =$ number of registers in a vector V. The binary value is then converted to decimal using a standard conversion equation. Let V^{bin} be a binary value of k, and v be an element of V^{bin} which can be either 0 or 1:

$$V\{v_k - 1, \ldots v_3, v_2 v_1, v_0\} \tag{5.9}$$

The decimal representation of V (V^{dec}) is calculated as:

$$V^{dec} = (v_0 2^0) + (v_1 2^1) + (v_2 2^2) + \ldots + (v_k 2^k) \tag{5.10}$$

Therefore, a decimal value for each V^{bin} from the five PLC register areas (I, Q, M, P, C) is generated.

PLC Vendor	PLC Model	Library	Protocol	Read Ack ID	Write Ack ID
Siemens	S7-300	Snap7	S7 Comm	$0x04$	$0x05$
Rockwell (Allen-Bradley)	ControlLogix-1756 (AB-CLX)	Pylogix	CIP	$0xc1$	$0xcd$

Table 5.2: Comparison of PLC network communication properties

Unidirectional Network Communications

We define a PLC's run-time network communications C as *responses* to other devices on the network during the PLC execution cycle. In contrast to other studies, for example, work presented in [204] and [115], we use a unidirectional flow of traffic, resulting in significantly reduced data processing overheads. Moreover, protocol-specific acknowledgement packets are returned, which are typically small in size in comparison to the original request packets sent to the PLC.

In general, network data generated by the PLC during execution does not influence the underlying physical process being controlled by the PLC, and similarly state changes in the physical process do not directly generate network data. Therefore, we identify an absolute deviation in network traffic metrics while a controller is under attack, and use these properties to enrich the anomaly diagnosis stage of the proposed methodology. Specifically, the *average packet length* (U^{len}), *number of read acknowledgment packets* (U^{rak}), and *number of write acknowledgement packets* (U^{wak}) are used as network features within a given dataset Z_k at time point t_i:

$$U_m(t_i)\{U^{len} \frown U^{rak} \frown U^{wak}\} \tag{5.11}$$

A network packet sniffer module is implemented to capture the ICS network traffic. The packet sniffer is built using the Scapy Python library[1], which provides a framework for real-time packet acquisition and crafting. Furthermore, a Berkeley Packet Filter (BPF) rule is also implemented to reduce the dataset to only include PLC network responses in order to significantly reduce the overheads of data pre-processing. The selection of these specific network metrics is motivated by the idea that cyber attacks targeting PLCs will use the network as a method of payload delivery, due to the lack of alternative interfaces on the PLC. Further-

[1] https://scapy.net

more, PLCs are often physically secure in locked cabinets for additional security. Therefore, we assume that given a set of incoming network packets to the PLC, an associated set of outgoing packets will also be observed. Subsequently, we calculate the defined network metrics that are included within the feature set. Moreover, the number of data read and write access acknowledgement packets are obtained through deep packet inspection using byte identifiers in the application layer, as defined in table 5.2. Read and write acknowledgement packets are generated as a result of a network node issuing data access requests to the PLC, for example an HMI requesting to read a range of output registers to update the values presented to the ICS operator on the HMI display. Hence, the mean packet size in bytes is calculated for all the captured packets since this can be indicative of the type of acknowledgement returned from the PLC.

PLC Device Logs

Querying the PLC logs can be used to analyse system events that have occurred during the PLC execution cycle, typically associated with the hardware of the PLC CPU and related modules that are connected on the PLC communication backplane. As identified in Chapter 4, artefacts derived from PLC device logs and diagnostics[2] are grouped under ADT4. These are typically accessed either through the PLC's diagnostic buffer or saved onto a form of removable non-volatile storage such as a memory card. Generally, PLC device logs are designed to monitor engineering interactions with the PLC such as downloading a project, changing operation mode and noting module errors. We identified that these logs do not map to the physical process, and consequently have significant limitations in aiding the initial detection of PLC anomalies. Furthermore, as customised log events are not consistent across different PLC models, as explored in the analysis of DT4 in Chapter 4, we utilise default log configurations within the anomaly diagnosis model. Therefore, we calculate the *number of PLC device logs* (L^{tot}) generated at time period t_i as the metric for device log data (L) in a given dataset Z_k.

The use of PLC device logs for the anomaly detection stage of the model is limited due to both their detachment from the physical process and the lack of scalable format between PLC vendors, as identified in Chapter 4. Furthermore, they are not suitable for real-time

[2]Unless otherwise stated, the terms device logs and diagnostic logs are considered to be synonymous

detection since the acquisition process for PLCs, that use an SD card as a storage medium for logs, can only be conducted while the PLC is in a Stop state. Despite these limitations of using PLC device logs, it was noticed that the diagnostic buffer is typically populated during events that could be associated with anomalous behaviour resulting from a system fault. PLC device logs are acquired either through the PLC IDE, which communicates with the on-board diagnostics buffer of the PLC, or by a accessing a memory card connected to the PLC CPU module. PLC system events are parsed and accumulated post-acquisition, and represent the total number of device logs generated by the PLC.

5.3.2 Stage One: Register State-Based Anomaly Detection

Before we can attempt to contextualise an anomalous event, the anomaly first needs to be identified. Using novelty detection algorithms, subtle anomalies in PLC states represented through register values, can be identified. As PLC states also have a physical manifestation (see Section 5.2), perturbations in PLC registers are often also reflected as a result of changes within the underlying physical process. A baseline PLC run-time state dataset is first produced comprising the artefacts as defined in Figure 5.1. Test datasets (Z'') consisting of a partial PLC run-time state are generated, specifically only containing a PLC register snapshot and network traffic capture, which are both pre-processed in the same manner as the baseline dataset. As logs can only be generalised as an offline artefact, since the acquisition approaches require the PLC to be in an inactive state, we omit them from the datasets at this

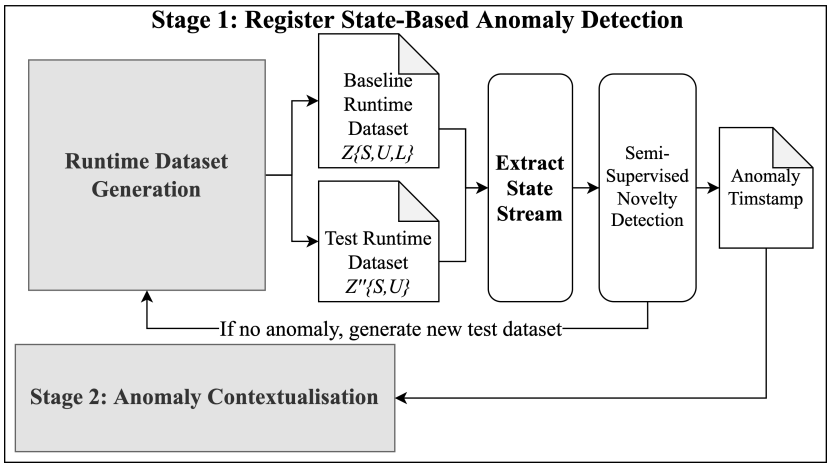


Figure 5.2: Register State-based Anomaly Detection - Stage 1 of the Anomaly Contextualisation Model

stage and utilise the log metrics only in the anomaly contextualisation process.

A semi-supervised approach is adopted to enable automated near real-time detection, which has multiple benefits when implemented for an ICS. Firstly, the approach does not require labelled anomalous data, which is particularly beneficial due to the high volume of volatile process data generated by a PLC, and the potentially safety-critical environments they operate in [18]. Furthermore, since PLCs in general drive and operate physical processes with consistent states, they generally have a deterministic operating behaviour. Evidently, this makes ICS well-suited to baseline dataset generation under normal operation, which is a fundamental aspect of novelty detection and semi-supervised ML algorithms.

While the underlying physical processes in ICS are unlikely to change often as they are based on control theory principles, the way in which a PLC is programmed to control such processes could change. For example, the application code of the PLC could be updated to use a reduced set of registers and hence be more efficient. In such cases, the detection baseline would need to be updated to reflect this new approach. One suitable solution for this would be to employ *online learning* techniques. In online learning, the ML model is continuously updated with new data [205]. However, there are certain challenges with using online learning approaches in ICS contexts. For instance the requirement for continuous data streams could potentially impact the real-time operational requirements of ICS networks, particularly where resource-constrained and legacy components are used such as PLCs. The use of online learning for ICS anomaly detection, however, is severely limited in existing literature, and hence future research is required to better understand the potential ICS applications. We revisit this topic again later in Chapter 7 of this thesis.

At any point in time t_i, an anomalous state (a_j) could exist. If a_j is identified by the novelty detection approach, the timestamp of the point at which the anomaly was first identified is extracted, defined as $t_i(a_j)$, and is used in the subsequent anomaly contextualisation stage. Figure 5.2 illustrates the high-level process of PLC register state-based anomaly detection using the baseline and test datasets and a semi-supervised learning approach.

5.3.3 Stage Two: Anomaly Contextualisation

Although the anomaly detection stage is conducted in near real-time, contextualising the anomalous event is ultimately conducted offline due to the collection methods required for PLC logs, as discussed in Chapter 4. The acquired log file is combined with the partial test run-time state dataset to form a complete dataset. As deviations in PLC registers, and therefore PLC states, can occur from both cyber attacks and system faults, we omit these from the contextualisation process and focus on the network and log metrics within the run-time state datasets. Figure 5.3 demonstrates how these datasets are combined within the second stage of the Anomaly Contextualisation model.

Specifically, to contextualise the corresponding deviation of each $a \in A$, we first calculate averages from network and log metrics in the baseline PLC run-time data set. From this, and utilising the baseline metrics, a sum of the absolute standard deviation is calculated for both network and device logs. Based on the number of metrics, a weighting is applied, resulting in an incident indication for each anomalous sample identified in stage one. The network or log deviation at time t, for m total metrics in subsets of size n, where x_i is a specific subset metric, defined as:

$$Dev_t = n(\frac{1}{m}) \sum_{i}^{n} |x_i - Baseline\_Mean(x)| \textbf{\textit{Dev}} \in \{U, L\} \tag{5.12}$$

A given a is diagnosed by contextualising the incident as *cyber attack* (a^{atk}) or *system fault* (a^{flt}) based on the deviations calculated by equation (5.12). Our approach adopts the idea that PLC device logs are typically generated during anomalies where $a = a^{flt}$, as device

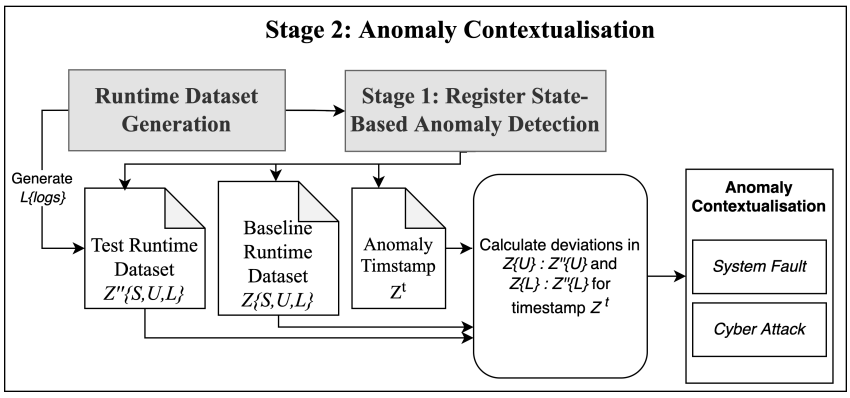


Figure 5.3: Anomaly Contextualisation stage (Stage 2) of Anomaly Contextualisation Model

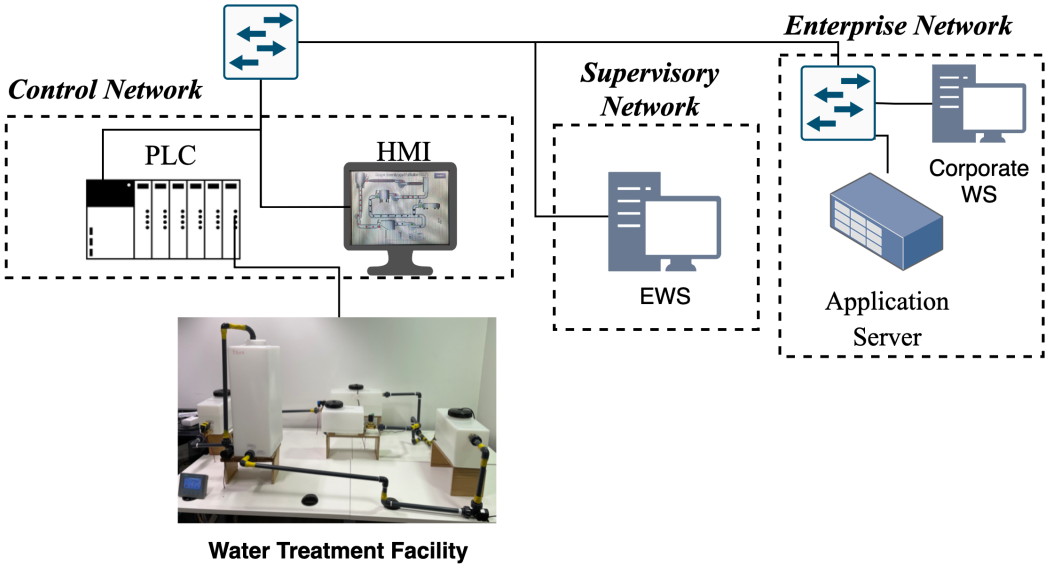


Figure 5.4: Simplified evaluation architecture with GULP testbed (abstracted from figure 3.6)

logs have been implemented by vendors for this type of event [206]. Conversely, an anomaly denoted as $a = a^{atk}$ will typically utilise network channels as attack vectors, thus network metrics can be used to identify these incidents. One reason for this is that PLCs lack conventional physical interfaces that are commonly exploited in attacks, such as USB ports.

5.4 Evaluation

All of the datasets used within the evaluation of the anomaly contextualisation model were generated through the GULP testbed, presented in Chapter 3, Section 3.3.1. It is important to note that while the GULP testbed emulates a water treatment facility, the data collected for our methodology, and the devices we use, can be generalised to other ICS architectures and wider OT environments. Figure 5.4 illustrates an abstracted version of the overarching threat model and evaluation setup presented in Section 3.3.1.

5.4.1 Evaluation Anomaly Scenarios

To evaluate the performance of the Anomaly Contextualisation model, we develop and assess three anomaly scenarios that target the PLC, each of which is generalised and launched across two PLC vendors.

Scenario one involves a cyber attack denoted as a^{atk} on one of the PLCs controlling the GULP water treatment facility. For this, we assume the position of an escalated outsider threat actor defined in Chapter 3 where the adversary would originate in an external position to the supervisory network and infiltrate through the enterprise network, as depicted in figure 3.6. The attacker firstly performs a reconnaissance stage to identify the manufacturer and the PLC model under attack, which is used to select vendor-specific malware scripts that target the memory contents of the PLC. It is important to note that, while specific implementation for the attacks is designed for the GULP testbed and the PLCs models that we have included within the evaluation, the methodology and overarching threat model can be applied to multiple PLC vendors because of the generalisations that can be made regarding how PLC use registers.

The attacks implemented in this scenario are derived from a subset of manipulation of data techniques presented in Figure 3.8. Specifically, the attacker's objective is to introduce malicious perturbations into the PLC registers by injecting data into memory through crafted network packets, ultimately manipulating the underlying PLC states. As these registers are typically not encrypted so that other devices on the network can perform read and write actions on them, they can be manipulated and are therefore a common attack vector for adversaries. Building on the threat model presented in Chapter 3, we implement an attack module which initially selects a random subset of generalised PLC memory register areas, for example inputs and outputs. A random set of register addresses are then subsequently generated for each selected memory register area. Finally, a random binary value is injected at each register address to execute the attack, altering their contents. Two single-register injections are performed, followed by a multi-register attack to evaluate the detection performance on both subtle and more obvious register fluctuations.

Scenario two focuses on system faults as an anomalous event, which we denote as a^{flt}. As discussed in Section 5.2.2, there are many different types of fault that can occur within an ICS. As no definitive set exists comprising all possible faults, we extracted three of the the most commonly identified fault types, which were also selected based on the lower overheads required for their implementation in a lab environment. These are defined as:

1. *Stop/Start operation fault*: This changes the PLC operating mode from START to STOP, and was the primary fault adopted for evaluation as if the PLC encounters a

system fault, it will often default into STOP mode.

2. *Key-switch fault*: Using the on-device switch, we change the mode of the PLC from RUN to program or reset.

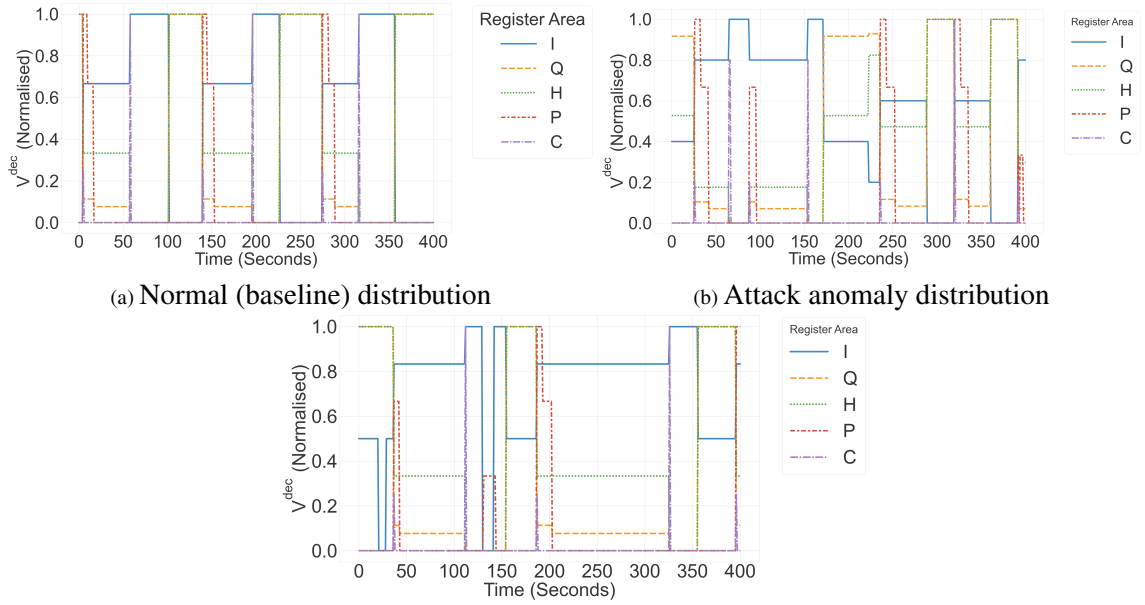
3. *I/O module fault*: This simulates a corrupt or removed hardware module from the PLC.

All three of these faults are intended to be generalisable to other PLC models and are all implemented through physical interactions with the PLC. For each time, scenario two is enacted, one of these three faults is introduced into the operation of the PLC. We perform an equal number of evaluations on each fault within scenario two.

Scenario three aims to assesses the performance of detecting and contextualising events that include both a^{atk} (cyber attacks) and a^{flt} (system faults). The attack module manipulating PLC registers is executed twice, performing one single-register and one multi-register attack, and a fault from scenario two is also introduced. Therefore, a combination of events from scenario one and two are generated and included within the test datasets comprising anomalous data.

5.4.2 Anomaly Detection Performance

To examine how different anomalous events are represented in the PLC run-time state datasets, the time-series distribution PLC register values is explored. Figure 5.5 presents PLC register distributions for scenario one, scenario two, and a baseline with no identified anomalous data. Considering the baseline distribution representing normal operation shown in figure 5.5a, there is a consistent flow of states with a deterministic pattern, seen as the physical process cycles through the PLC states. However, during the anomalous event scenarios, there are fluctuations and variance in the register states, as demonstrated in figure 5.5b for scenario one and figure 5.5c for scenario two. Therefore, the resulting perturbations demonstrate that PLC register states can be used to identify anomalous PLC behaviour within both test conditions. A distribution plot for scenario three is not included as it includes data from both scenario one and two. However, the distributions and deviations in PLC register states that are illustrated in Figure 5.5 are not evidential of the different types of anomaly in isolation of other data features.



(a) Normal (baseline) distribution

(b) Attack anomaly distribution

(c) Fault anomaly distribution

Figure 5.5: Normal and Anomalous Distribution of PLC Register States

Model Section

Based on the generated feature set size within a run-time state dataset Z^k described in Section 5.2 and challenges with generating large amounts of training data within real ICS environments, we select semi-supervised ML algorithms for the anomaly detection stage where we compare two algorithms; One Class Support Vector Machines (OCSVM) and Local Outlier Factor (LOF), which were both defined in Chapter 2 (Section 2.2.2). The efficacy of using OCSVM for ICS anomaly detection has been demonstrated through previous literature [18]. Furthermore, both OCSVM and LOF are particularly suited to training data comprising fully of normal operating behaviour, which is advantageous within the context of ICS since obtaining abnormal data, for instance attack samples, is more challenging. Additionally, previous literature has signified that OCSVM and LOF can be more efficient regarding time consumption compared with unsupervised clustering approaches [207].

We note that while semi-supervised learning require less labeled data than supervised, providing sufficient amounts of labeled anomalous data to optimise training performance can be challenging within ICS. To address this concern, we also utilise an unsupervised algorithm by adapting the Isolation Forest (IF) formulation with a typical outlier detection approach. IF adopts principles of decision tree algorithms, isolating outliers by randomly selecting a feature and subsequently partitioning random split values on said feature. The partitioning can

be represented in a tree structure, with anomalies producing smaller paths than others. IF is known to be particularly computationally efficient with low time complexity compared with other unsupervised approaches, and thus suitable for rapid anomaly detection in ICS [61]. Moreover, IF was selected for evaluation because unsupervised approaches are particularly suited for the detection of unpredictable attacks, such as those posed by advanced persistent threats (APTs) [208].

The implementation of all classifiers includes an automated hyper-parameter tuning using a grid search approach as presented in table 5.3. For OCSVM, a radial base function (*rbf*) kernel was selected because the distribution of normal PLC states will likely cluster in vector space, and therefore multiple decision regions may need to be established. In addition, a low *nu* is essential as the baseline data should comprise purely positive samples. Furthermore, the *gamma* hyper-parameter is scaled so that decision regions are calculated to suit the distribution of features within the training set. In regards to LOF, the novelty hyper-parameter was set as *true* which allows the algorithm to operate in a semi-supervised manner. Additionally, the algorithm parameter was set as *auto* to allow the nearest neighbor algorithm to adapt and generalise for PLC registers which may differ in distribution between controllers. Regarding IF, maximum samples is set to *auto* so that the maximum depth of decision trees used for classifying anomalies can adapt to the number of samples in a given testing set. In addition, a total of one hundred n estimators was selected to allow reliable performance, while keeping the computational cost relatively low. Moreover, the contamination rate is set to reflect a low number of anomalous samples in the threat scenarios.

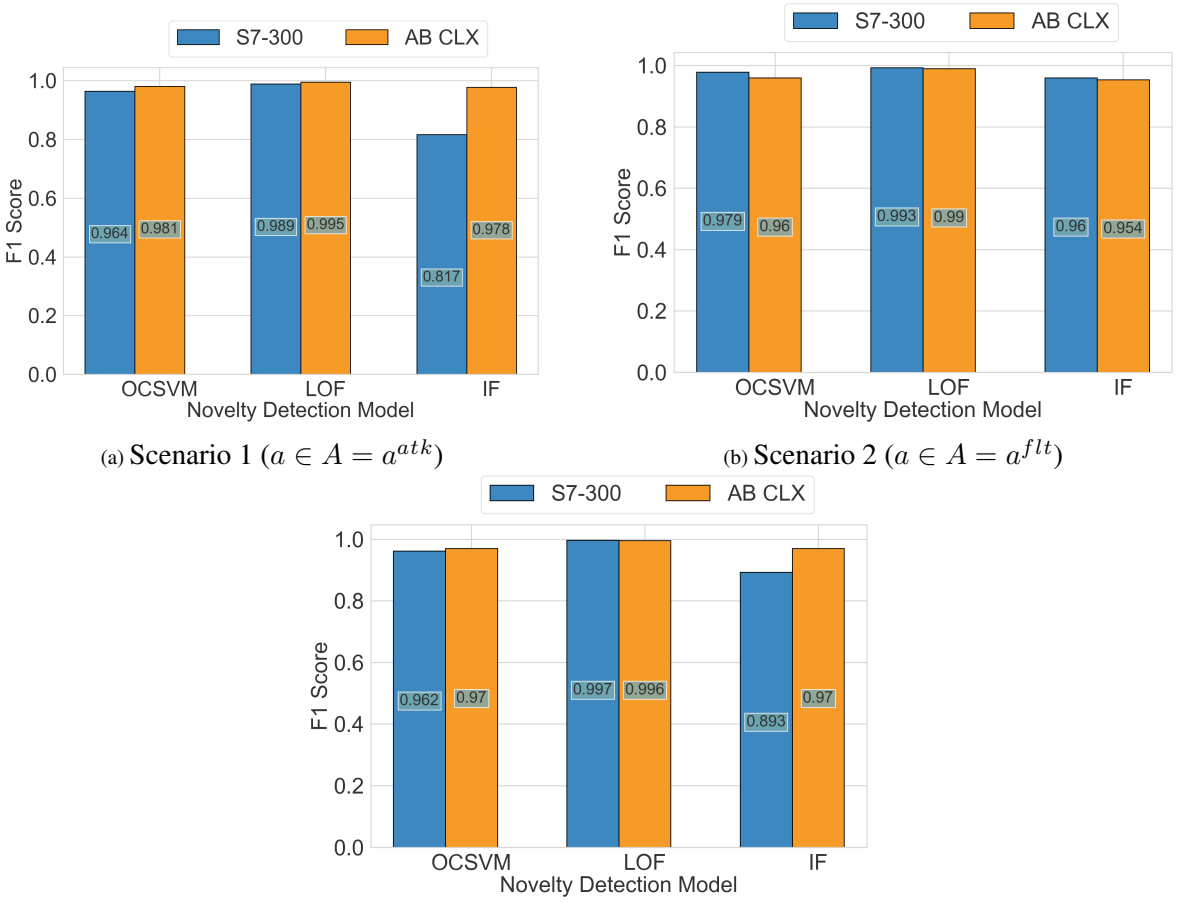
Table 5.3: Model Hyper-Parameters from Grid Search where: *OCSVM = One-Class Suport Vector Machine; LOF = Local Outlier Factor; IF = Isolation Forest*. All other parameters were kept at default values.

Model	Parameters
OCSVM	**Kernel**: RBF, **nu**: 0.001, **Gamma**: Scale
LOF	**Novelty**: True, **Algorithm**: Auto
IF	n **estimators**: 100, **Maximum Samples**: Auto, **Contamination**: 0.1

Anomaly Scenario Detection Results

To evaluate the PLC register state-based approach for anomaly detection, we utilise a sample size of 3000 normal data points to train the detection models, while each test scenario comprised 400 samples. Across all three ML algorithms, LOF was the most effective at detecting anomalous behaviour on both PLC models with an average F1 score of 0.993% for the S7-300, and 0.994% for the AB-CLX models, shown in figure 5.6 where the *precision* and *recall* results are also included. The second best performing model was OCSVM, with an average F1 score of 96.8% for the S7-300, and 0.970% for the AB-CLX PLC models.

Conversely, the unsupervised approach, IF, did not perform as well, resulting in mean F1 scores of 0.890% for the S7-300 and 0.967% for the AB-CLX, although these scores are still comparable with state-of-the-art anomaly detection models for PLCs. One reason for the decrease in score with the IF approach is that when detecting outliers, data points with low



(a) Scenario 1 ($a \in A = a^{atk}$)

(b) Scenario 2 ($a \in A = a^{flt}$)

(c) Scenario 3 ($a \in A = a^{atk}$ & a^{flt})

Figure 5.6: F1 scores for anomalous PLC state detection using different ML algorithms under each test scenario

Table 5.4: Performance Evaluation of Anomaly Detection with OCSVM, LOF and IF algorithms for each PLC model under all three anomaly test scenarios (α^{atk}, α^{flt} and α^{atk} & α^{flt}

Model	PLC Model	Test Scenario	Precision	Recall	F1	Time
OCSVM	S7-300	α^{atk}	0.972%	0.955%	0.964%	19ms
		α^{flt}	0.992%	0.966%	0.979%	23ms
		α^{atk} and α^{flt}	0.970%	0.954%	0.962%	28ms
	AB-CLX	α^{atk}	0.967%	0.995%	0.981%	29ms
		α^{flt}	0.930%	0.991%	0.960%	18ms
		α^{atk} and α^{flt}	0.948%	0.991%	0.970%	14ms
LOF	S7-300	α^{atk}	0.997%	0.978%	0.989%	38ms
		α^{flt}	0.998%	0.987%	0.993%	41ms
		α^{atk} and α^{flt}	0.997%	0.993%	0.997%	42ms
	AB-CLX	α^{atk}	0.993%	0.989%	0.995%	38ms
		α^{flt}	0.993%	0.981%	0.990%	33ms
		α^{atk} and α^{flt}	0.994%	0.992%	0.996%	31ms
IF	S7-300	α^{atk}	0.763%	0.879%	0.817%	88ms
		α^{flt}	0.993%	0.923%	0.960%	87ms
		α^{atk} and α^{flt}	0.884%	0.900%	0.893%	84ms
	AB-CLX	α^{atk}	0.994%	0.957%	0.978%	85ms
		α^{flt}	0.992%	0.912%	0.954%	84ms
		α^{atk} and α^{flt}	0.988%	0.940%	0.970%	81ms

frequency of occurrence are identified as abnormal. However, these points have already been identified as normal points when training the ML detection models. In particular, due to the different firmware/hardware between the PLCs, the S7-300 ladder logic contained more of these sporadic system states. It is evident that both OCSVM and LOF are the more effective classifiers used for the detection model when identifying anomalous behaviour. Given that an ICS can contain many sporadic system states, it follows that a classification method which can cope with this variance is the most suited. Hence, this strengthens the case for using novelty detection ML approaches for PLC-specific anomalies. Regarding computational performance, the OCSVM model was able to classify anomalies the quickest, as shown in table 5.4, with an average time of 21.8 milliseconds (ms) across both PLCs. In contrast, IF was the most computationally expensive algorithm with a cross-controller average time of 84.8 ms. Although the LOF model took considerably longer than OCSVM on average, it is clear that both semi-supervised methods significantly outperformed IF.

While we have evaluated IF as a potential unsupervised approach due to the benefits of requiring no data labels, other unsupervised approaches could have been explored. For instance, clustering methods such as K-means or Affinity Propagation can be used to distin-

Table 5.5: F1 scores for test scenarios using LOF model showing percentage coverage of included PLC states

Test Scenario	PLC State Coverage		
	70%	*80%*	*90%*
α^{atk}	0.72	0.89	0.97
α^{flt}	0.76	0.91	0.96
α^{atk} and α^{flt}	0.76	0.91	0.96

guish outliers. Distributions of register values, such as those demonstrated earlier for attacks and faults in figure 5.5, could be used to generate clusters for multi-state PLC processes where an unknown number of states could exist. Such cases would make Affinity Propagation particularly suited as an approach since it does not require a pre-determined or estimated number of clusters before running the algorithm, unlike K-means. However, the main challenge with Affinity Propagation is the computational overheads and time complexity, which is particularly high for larger datasets. Hence, there are likely to be barriers to using such an approach for real-time detection, particularly when compared with semi-supervised methods.

In real ICS deployments, we can assume that some PLC states will be less frequent than others, particularly those states that provide back-up or safety-critical functions [51]. Therefore, as the proposed approach uses the formulation of logical PLC states derived from unique combinations of registers, an additional test was performed to determine how the inclusion coverage of PLC states within the training data impacts the detection performance for all three test scenarios. The test was performed using the LOF model since it had achieved the highest F1 scores, and we provide an average F1 score for both PLCs, demonstrated in table 5.5. Interestingly, at 90% coverage of states, detection performance is still highly comparable with the full coverage (represented by the F1 scores in table 5.4). However, as the quantity of states omitted from the training dataset increases, the performance steadily decreases, where at 70% PLC state coverage we identify a loss in performance of 26%, specifically for α^{atk} (cyber-attack) scenarios. It is important to note, however, that the total number of PLC states used within our datasets is likely to be smaller compared with real systems, and therefore, this particular analysis is more sensitive to state coverage than if we evaluated the approach within an actual ICS environment.

In table 5.6, the detection performance measurements of several existing ICS anomaly detection studies have been collated to provide a comparison with the PLC register state-based

Table 5.6: Comparison between anomaly detection performance, measured with accuracy and F1-score (where available), for algorithms proposed in this thesis and those reported in the existing literature. Where more than one accuracy or F1 score is provided by the study, an average was used.

Approach	**Accuracy**	F1 **Score**
Yau & Chow (2017) [18]	0.95	0.83
Lin *et al* (2018) [66]	-	0.82
Chan *et al* (2019) [67]	-	0.94
Yang *et al* (2020) [103]	0.75	0.97
Ahmed *et al* (2021) [113]	0.96	-
Huang *et al* (2021) [209]	0.89	0.85
PLC Register State-based Anomaly Detection	0.98	0.99

detection algorithm presented in this chapter. It is evident that the approach is highly competitive with existing algorithms that have been previously proposed in the literature. While our approach demonstrates the highest accuracy and F1-scores compared with recent work, it is particularly significant since the PLC register state-based algorithm can detect occurrences of cyber-attack and system fault anomalies.

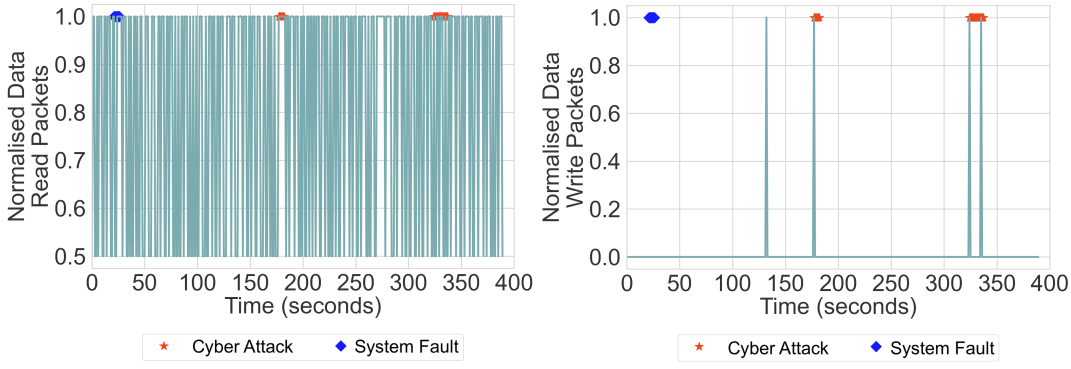
5.4.3 Anomaly Contextualisation Performance

The second stage in the proposed model aimed to contextualise anomalous data points that were identified in the anomaly detection phase of the methodology by using deviations in network and log features. During the experiments, it was observed that a higher deviation in the PLC network data metrics, indicates a probable cause of a^{atk} (cyber attack). Conversely, a higher deviation in device log metrics dictates a probable cause of a a^{flt} (system fault).

Figure 5.7 and figure 5.8 illustrate the normalized PLC network traffic metrics and device logs for the S7-300 and AB-CLX PLCs, respectively. Diagnosed anomalies are also displayed on these graphs as either a system fault (blue marker) or cyber-attack (red marker). From these graphs, we can see that an increase network write acknowledgement packets shown in figure 5.7b and figure 5.8b greatly correlates with the diagnosis of a cyber-attack anomaly (red marker), which is understandable as acknowledgments would be responses to manipulations of data on the PLC. Furthermore, a cyber-attack is also indicated by the drop in average network packet size illustrated in figure 5.7c and figure 5.8c, however this

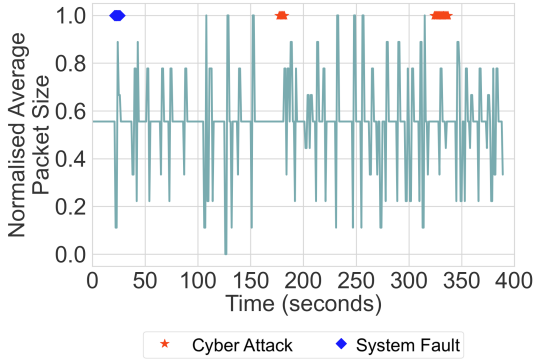
is clearer for the AB-CLX PLC as network packet size fluctuates less during normal operation. A drop in average packet length is most likely due to the fact that write operations sent to a PLC do not return any data, and therefore their response size is much lower when compared to the response of a read. Hence, during an attack which is writing to the PLC, the average response packet length will slightly decrease. Conversely, deviations in the amount of PLC device logs illustrated in figure 5.7d and figure 5.8d show significant increase at the time where an incident is classified as a system fault. In general, this further supports the argument that PLC logs are strongly suited to identifying faults rather than attacks.

When analysing deviation in the number of read acknowledgements it can be seen that the results differ greatly between devices. Although neither correlate greatly towards indication of cyber-attack or fault, the S7-300 PLC has significantly less consistency in the number of read requests. This serves to show that when generalising across PLC vendors, the difference in manufacturing and fundamental PLC design can have significant impact in developing general security solutions for ICS. In addition, this statement is reinforced by the isolation forest classifier struggling with a more sporadic dynamic I/O trace, which was necessary due

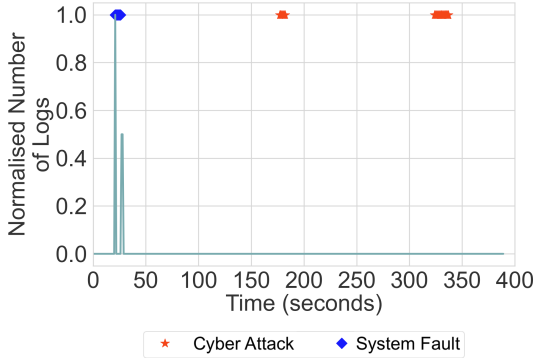


(a) S7-300 Packet Reads

(b) S7-300 Packet Writes

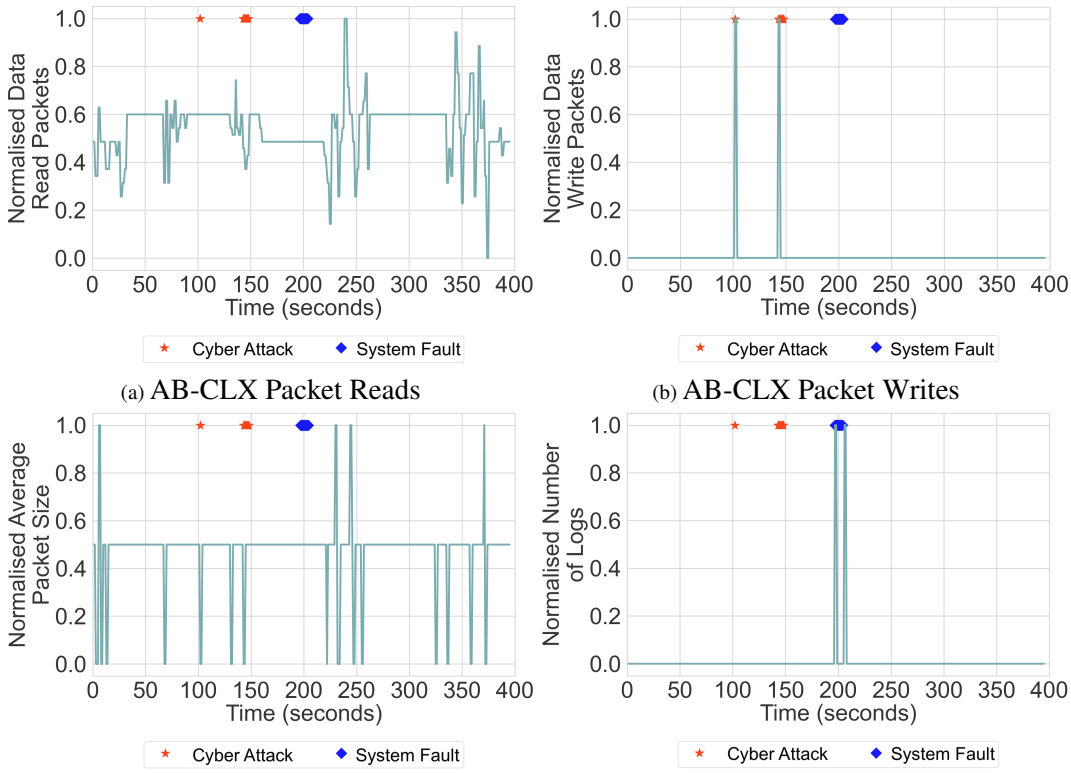


(c) S7-300 Packet Size



(d) S7-300 Total Logs

Figure 5.7: Deviations in anomaly contextualisation features for scenario 3 with Siemens S7-300 PLC



(a) AB-CLX Packet Reads

(b) AB-CLX Packet Writes

(c) AB-CLX Packet Size

(d) AB-CLX Total Logs

Figure 5.8: Deviations in anomaly contextualisation features for scenario 3 with Siemens AB-CLX PLC

Table 5.7: Classifier Hyper-Parameters for supervised learning in anomaly contextualisation

Classifier	Hyper-Parameters
RF	**random_state**: 0
GNB	**var_smoothing**: $1e^{-9}$ (default)
LR	**penalty**: l2 (default), **solver**: newton-cg: **C**=1.1

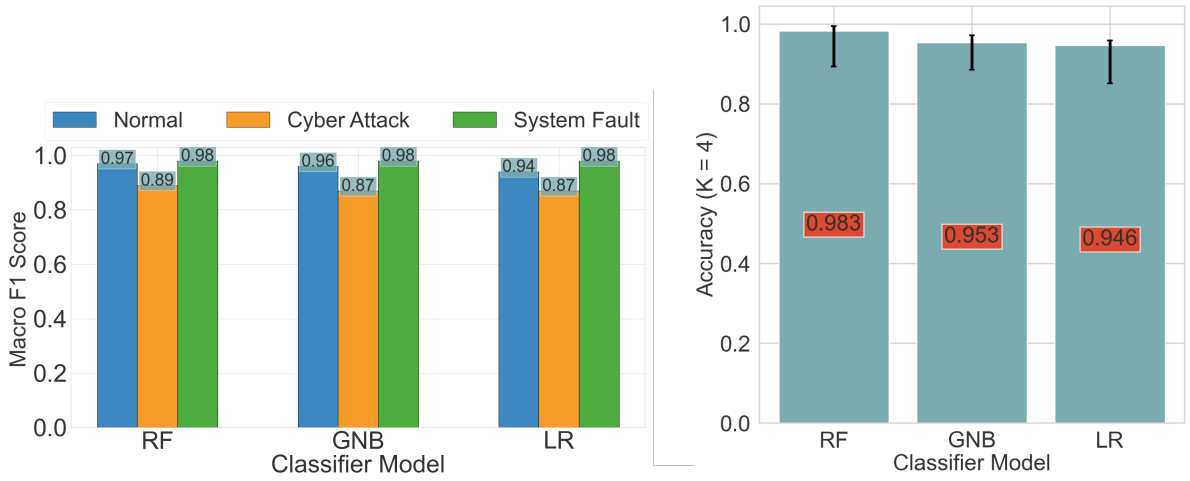
to the way in which the S7-300 PLC needed to be programmed.

From the graphs illustrated in figure 5.7 and figure 5.8, there are clear distinctions between how the features of the run-time dataset deviations relate to different anomaly scenarios. However, these represent one experimental instance with limited data points that represent feature deviations subsequent to PLC cyber attacks and system faults. Taking this into account, we extend the contextualisation stage to include supervised classification of the different scenario types in order to further validate the deviation algorithm defined in equation (5.12), and examine whether subtle changes in feature set metrics are able to accurately contextualise an anomaly.

We firstly examine the performance of supervised classifiers at discriminating between the

anomalies that are classed as cyber attack and those that are cyber faults based on the scenarios defined in Section 5.4), as well as normal behaviour. Hence, this inevitably becomes a multi-class classification problem where we have three classes, namely *normal*, *attack*, and *fault*. The data points associated with each scenario from datasets of both models of PLC, were manually labeled and combined into a single training dataset. The final dataset comprised data for 50 attack scenarios and 50 fault scenarios, 25 for each PLC model, along with 140 data points of normal PLC behaviour (240 data points in total). The use of unbalanced classes within the dataset, which contains a higher proportion of normal data compared with the anomalous data, is characteristic of real-world scenarios where anomalous data is more challenging to produce [210]. Three classification algorithms, Random Forests (RF), Gaussian Naive Bayes (GNB), and Logistic Regression (LR) are evaluated and selected due to their suitability for multi-class classification problems [211] and also towards smaller datasets [212], which is particularly advantageous for ICS where it is challenging to generate anomaly data points. In addition, Naive Bayes and LR classifiers have previously demonstrated high computational efficiency, particularly when smaller datasets with low dimentionality are used [213]. A grid search hyper-parameter tuning process was selected to optimise the performance of anomaly contextualisation, and the selected parameters are provided in table 5.7. The grid search approach was selected primarily as a low number of features are used and therefore the performance is less impacted by the curse of dimentionality [55]. Unlike the hyper-parameter tuning with the approaches selected for register state-based anomaly detection in table 5.3, there were minimal adjustments required from the default parameters. Specifically, for LR, we select the *newton-cg* and the *l2* penalty for ridge regression, as we are performing multi-class classification. As this is a multi-class classification problem, we utilise the macro F1-Score, which uses average precision and recall measures calculated from all three classes.

We identified that all three classes achieved high macro F1 scores calculated from the resulting confusion matrices. However, as indicated in figure 5.9a, system fault anomalies were able to be more easily distinguished than cyber attacks achieving average macro F1 scores across the three classifiers of 0.98 and 0.88, respectively. Comparable macro F1 scores were produced by all three classifiers, with RF performing marginally better on discriminating normal and cyber attack classes compared with GNB and LR. To validate the anomaly contextualisation approach further, we employ a *K-Fold* cross-validation test to calculate the



(a) Macro F1 Scores

(b) K-Fold Accuracy Scores

Figure 5.9: Anomaly contextualisation scores for multi-class classification

overall accuracy of each classifier algorithm. Initial tests demonstrated that the optimal value of K if $K = 4$, although discrepancies between mean accuracy measurements with alternative K values are low within the range of ± 0.05. Similarly, the RF classifier demonstrates very high accuracy of contextualising PLC anomalies with a score of 0.983. Logistic regression in this case is the least effective for contextualisation, which is likely due to the high non-linearity of the features within the PLC run-time dataset consequently impacting on the ability of the LR coefficients to correctly predict discrepancies within individual features. Furthermore, as the dataset is composed of data points from both the Siemens and Allen-Bradley PLC models, the resulting high accuracy and macro F1 scores further support the generalisability of the feature set and deviation algorithm.

It is evident from the analysis provided by the K-Fold cross-validation that generally the RF classifier outperforms the GNB and LR models, however this does not substantiate the performance of individual test scenario since the dataset comprises both cyber-attack and system fault anomaly occurrences. Hence, we also demonstrate classifier suitability for the specific test scenarios involving two cases: 1) System fault Vs. Normal and Cyber-Attack, and 2) Cyber-attack Vs. Normal and System fault. The resulting Area Under The Curve (AUC) Receiver Operating Characteristic (ROC) (AUC-ROC) metrics provided in table 5.8a indicate that the all three classifiers are particularly effective at reliably predicting system fault anomalies, and hence provide highly optimised measures of separability between normal and anomalous behaviour, with RF achieving 0.97. Conversely, the AUC-ROC scores when predicting cyber-attacks, particularly for the GNB and LR classifiers, are not as ef-

Table 5.8: ROC-AUC scores and training and testing computation times for anomaly contextualisation classification

	Classifier		
	RF	**GNB**	**LR**
Fault α^{flt}	0.97	0.94	0.94
Attack α^{atk}	0.94	0.9	0.89

(a) ROC-ACU scores for fault and attack classification tasks for RF, GNB and LR classifiers

Classifier	**Train**	**Test**	**Samples**
RF	1.56s	0.59s	240
GNB	0.73s	0.05s	240
LR	0.85s	0.19s	240

(b) Time consumption for training and testing with RF, GNB and LR classifiers (measured in seconds)

fective reaching 0.9 and 0.89, respectively. From this, we can deduce that RF is the most effective classifier at performing anomaly contextualisation.

However, an additional dimension of performance evaluation regards the computational performance, in particular the time taken to perform the anomaly contextualisation process using classification. The setup specifications we use to train and test each classifier are a Windows 10 workstation with an Intel Core i5-5257U CPU (2.70 GHz) and 16 GB DDR3 RAM. The support of a Graphics Processing Unit (GPU) was not employed for these experiments. The resulting training and testing time for each classifier is represented in table 5.8b, and it is evident that all classifiers produce similar low time consumption, although classification with RF takes noticeably longer, especially when we conduct training ($1.56s$). The time consumption for GNB in comparison to RF represents a 53% and a substantial 91% decrease when performing training and testing, respectively. Hence, when we consider the ROC-AUC performance in unison with the percentage decreased in time consumption, GNB provides a more balanced approach. Moreover, it is likely that the use of larger datasets comprising an increased amount of normal data points would cause RF computational performance to extend into the magnitude of several minutes. However, as we have demonstrated that high classification performance metrics can be achieved through small dataset sizes, we deduce that the favourable classification performance of RF compared with the results produced by GNB and LR outweighs the increase in time consumption, particularly for testing.

Taking the performance of each classifier into account, we select RF to perform feature importance validation for each of the four feature metrics that compose the run-time dataset to determine their significance in PLC anomaly contextualisation. In order to evaluate feature significance, we firstly perform a permutation feature importance test using the same dataset comprising three classes, which provides a global examination of how each feature impacts

the performance of the classification model, and hence the RF classifier algorithm we utilise. Specifically, permutation feature importance measures the strength a particular feature by calculating the increase in prediction error of the proposed model after feature values are permuted, which breaks the relationship between the feature and the ground truth. The results demonstrated in figure 5.10 reveal that both diagnostic logs and data write network packets are the most significant features particularly in comparison to the average packet size and data read network packet metrics, where the resulting permutations decrease the accuracy by 0.315 (31.5%) and 0.221 (22.1%), respectively.

Although performing the permutation feature importance does provide a good indication of feature importance for anomaly contextualisation, it examines this through a global lens without indicating how features relate to the separate contextualisation classes. To address this limitation, we perform a Shapley Additive Explanations (SHAP) examination to explain how critical PLC diagnostic logs and network write packet features are in contextualising cyber attacks and system faults. SHAP provides Shapley Values for given features, which describe the contribution magnitude of each feature to the prediction of a particular class, compared to the average prediction for the entire dataset. A TreeSHAP algorithm, used for tree-based models, is implemented since we are using the RF classifier for classification. Demonstrated in figure 5.11, we can further see that both diagnostic logs and network write packet features are strong parameters for contextualisation accuracy. Specifically, the cor-

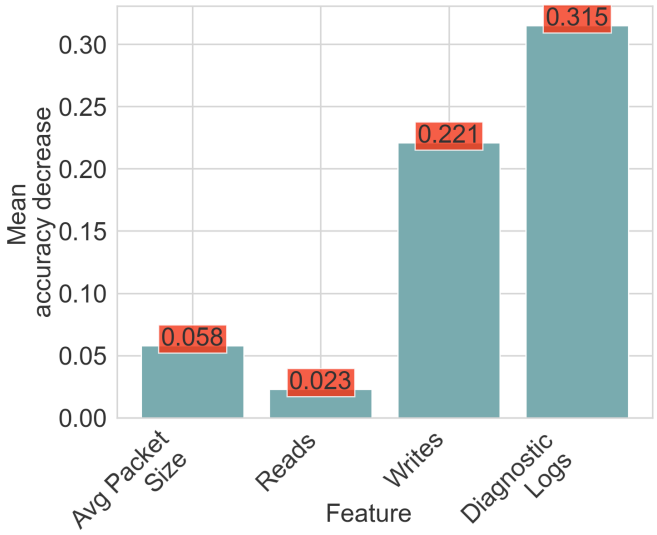


Figure 5.10: Permutation feature importance scores for each evaluation scenario

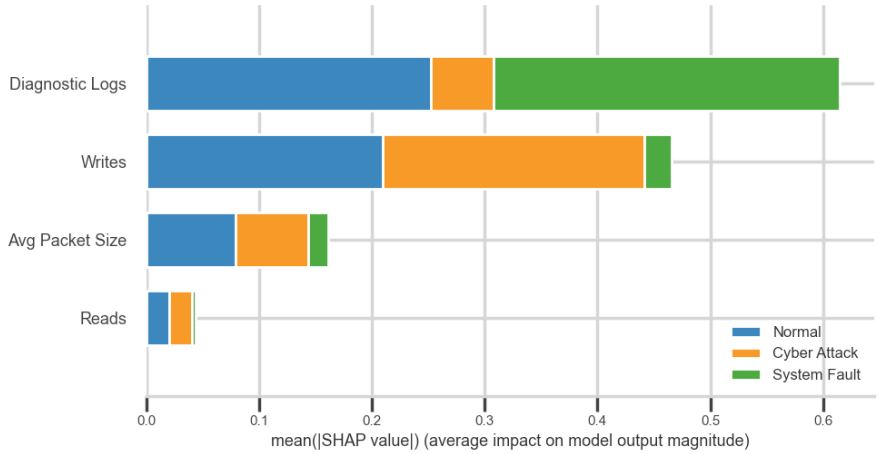
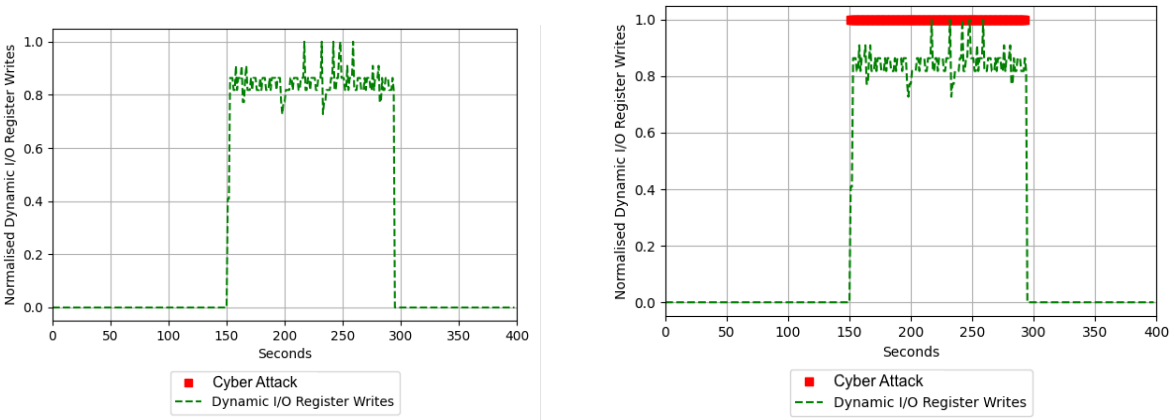


Figure 5.11: Mean SHAP values for localised feature importance based on multi-class prediction

rect contextualisation of cyber fault anomalies strongly requires diagnostic logs substantially more than other features present, scoring a mean SHAP value of 0.3, which is considerable compared with other metrics where SHAP values of less than 0.05 are demonstrated. Conversely, cyber-attack scenarios produce a high SHAP value greater than 0.2 for network write packets. In general, all selected network metrics produced higher mean SHAP values for the classification of cyber-attacks.

5.4.4 Good State, Bad Outcome

In reference to figure 5.7b, it can be seen that the PLC responded to a write request that was not identified by the register state-based anomaly detection approach. When examining the S7-300 PLC dynamic I/O trace logs, it was apparent that the attack module had forced a set of values to the PLC which were already active in the current physical process state. To test this, an attack module was implemented that utilised known-good states of the physical process. The module first reads the currently active dynamic values from PLC memory. Then, a random subset is selected, and their initially read values are continually injected for some pre-defined amount of time. This effectively freezes the state of the physical process. For example in the context of water treatment, this module could activate all the pumps during the transfer of water from filtration into a holding tank. Hence, by continually forcing this "good" state to the PLC, the holding tank may reach capacity and overflow since the valves to release the water will not be opened. It was found that in solely using dynamic I/O values



(a) Before Data Source Synthesis

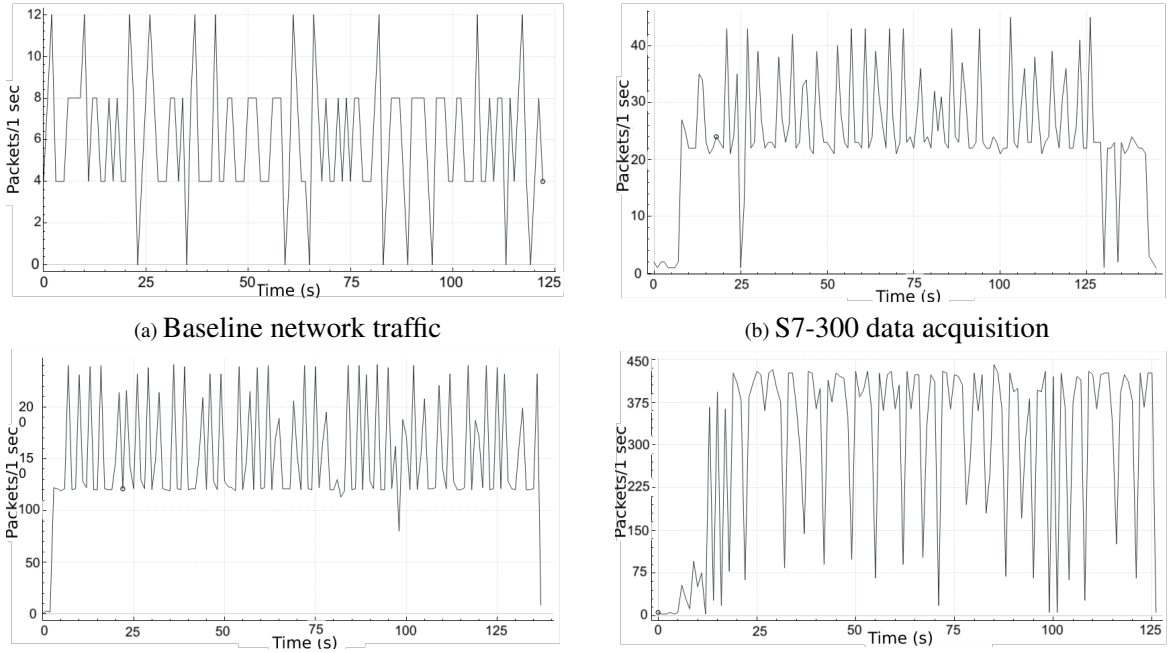
(b) After Data Source Synthesis

Figure 5.12: Known Good State Attack Detection

for anomaly detection, this type of attack was not identified, illustrated by the figure 5.12a. However, when we include write requests as a feature in the initial anomaly detection of our methodology, the detection module was able to identify and subsequently classify this attack vector, as demonstrated in figure 5.12b. Attacks of this type further highlight the significant need for synthesis across multiple data sources for optimal anomaly detection.

5.4.5 Network Performance Impact

To evaluate the network impact of the implemented acquisition method, Wireshark [132] is used to generate the network traffic flow graphs measuring the packets per second and examine the potential packet loss resulting from runtime dataset generation, as illustrated in figure 5.13. It is evident in figure 5.13c that the AB-CLX PLC acquisition introduces significantly more traffic on the network than the S7-300. The increased traffic is due to the way in which the CIP and EtherNet/IP protocols used by the AB-CLX establish a connection for querying data from PLC. CIP first registers an Ethernet/IP session with a PLC and creates a forward open request before issuing specific data access request packets to acquire memory register states. Furthermore, the mechanics of the interaction services that operate within CIP result in a *0x55* service packet being sent to return a register instance before executing a separate *0x4c* read request to return register's current value. Despite this, no packet loss was seen while acquiring data from either of the controllers. Additionally, although packet rate does decrease, there was no packet loss seen when acquiring data from both controllers in parallel. However, it is clear that when acquiring data from PLCs, caution must be taken to



(a) Baseline network traffic

(b) S7-300 data acquisition

(c) AB-CLX data acquisition

(d) Parallel data acquisition

Figure 5.13: Packet rate per second during run-time dataset generation

ensure that aspects of vendor-specific protocols do not impede network performance.

5.5 Summary

In this chapter, the first two stages of the PLC Anomaly Diagnosis framework, introduced in Section 3.2, have been addressed through a multi-stage model for the detection and contextualisation of PLC anomalies using specific data features, characterised as run-time artefacts. Earlier in this chapter, we define a method for logically representing changes in the underlying physical process, and the behaviour of the PLC in general, by using changes in discrete PLC register states, as defined in Section 5.2. The accuracy achieved through this PLC behaviour modelling is represented by the performance evaluation of stage one anomaly detection, which resulted in high F1 scores realised through OCSVM and LOF semi-supervised learning approaches. LOF in particular, provided notably high scores compared with existing state-of-the-art literature. An IF unsupervised approach, was also evaluated due to the challenges of data labelling for ICS datasets, that would be encountered through semi-supervised methods. While the achieved F1 scores were comparable for most cases, the detection time consumption for IF was higher compared with OCSVM and LOF. Although

the performance difference was in the order of magnitude of tens of milliseconds, the utilisation of larger datasets in real-world use cases would likely produce increased temporal overheads, impacting the detection time. In the evaluation of stage two anomaly contextualisation, initial results demonstrate the effectiveness of using deviations in network and device log metrics to inform occurrences of cyber attacks or system faults. Through further evaluation using supervised learning for multi-class classification, high F1 scores are achieved, with the RF classifier providing a more optimised approach given the lack of linearity between features. Feature importance examinations were performed to determine which features provide greater predictive power for PLC anomaly contextualisation. The results conform with the initial hypothesis that the generation of device logs is more aligned with cases of system faults. Conversely, network features and particularly write packets are stronger in contextualising anomalies that are cyber attacks. The anomaly contextualisation model enables the subsequent stage of attack fingerprinting, as defined within the anomaly diagnosis framework, and explored in the next chapter.

Chapter 6

PLCPrint - Attack Fingerprinting

6.1 Overview

The previous chapter explores how different types of PLC anomaly, specifically cyber attacks and system faults, can be discriminated from each other to provide anomaly contextualisation and ultimately improve incident triaging. However, on its own, the contextualisation of anomalous PLC behaviour does not directly facilitate ICS digital forensics, primarily because the chain of evidence and provenance of PLC data artefacts will not yet have been established. Hence, in this chapter, we build on the research established in Chapter 4 and Chapter 5 to address the final part of the PLC anomaly diagnosis framework by presenting a real-time fingerprinting approach to assist with the anomaly detection, and critically also, the forensic analysis of PLCs. The proposed methodology, referred to as *PLCPrint*, utilises the data artefacts and signatures stored in the memory of the PLC, as presented in Chapter 4 and is evaluated using the GULP testbed and attack scenarios developed in Chapter 3.

Performing attack fingerprinting offers a number of important advantages that go beyond the identification of attacks themselves. Firstly, using machine learning (ML) to classify attacks into different types and techniques, can enable a greater level of granularity in attack detection itself, significantly increasing the rate of detection. Secondly, identifying the attack vectors can enable rapid incident triaging by informing cyber-security professionals what the likely part of a system is under attack in a specific incident. Furthermore, the fingerprinting of PLC data artefacts, and in particular, those generated at memory level, demonstrates the

chain of evidence that is needed for a subsequent digital forensics investigation. More specifically, using fingerprinting to demonstrate the data provenance can provide key insights for forensic investigators regarding the prioritisation and targeted collection of digital evidence from the PLC and ICS components in general. Finally, understanding the types of attack that are targeting real-world ICS systems can encourage organisations to proactively implement cyber defence controls and policies. Consequently, this can help reduce the likelihood of similar attacks occurring again in the future.

The key contributions of this chapter are summarised as follows:

1. **PLC Memory Register Mapping (PMRM):** We present an entirely novel vendor-independent method that fingerprints the relationship between different properties of PLC registers, specifically the dependencies of registers in relation to PLC memory artefacts.

2. **Synthesised PLC Memory Artefacts:** We propose a synthesised dataset comprising different PLC memory artefacts, including the PLC application code, which has not been explored in previous studies within the context of PLC anomaly detection and fingerprinting.

3. **Memory Fingerprint Attack Detection:** A novel attack detection approach is proposed using PLC memory fingerprints generated through the PMRM technique and enabled using semi-supervised learning

4. **Memory Attack Type Classification:** Through the memory register mapping approach, we demonstrate high performance for PLC attack type classifications by comparing multiple supervised ML algorithms to rapidly enable incident triaging.

5. **Attack Technique Multi-Class Classification:** We demonstrate the ability of memory register mapping to identify specific attack techniques used to target a PLC by applying multi-class classification.

6.2 PLCPrint Architecture

As presented in Chapter 2, there are a number of research challenges and limitations with existing approaches to ICS fingerprinting regarding their utility in anomaly diagnosis and incident triaging applications. In this section, we present PLCPrint, an attack fingerprinting methodology for PLCs that enables the detection and classification of cyber intrusions. The high-level architecture of PLCPrint comprising five modules is presented in figure 6.1. The data acquisition module uses network commands to retrieve data from PLC memory. The acquired artefacts are then passed onto the fingerprint generator, which performs the *PLC memory register mapping* technique by analysing how memory registers are related to the artefacts. The output of this module is a PLC memory fingerprint that is used as a baseline of normal operation for a particular PLC. The memory fingerprinting attack detector component continuously acquires new temporary fingerprints from a PLC, and compares these with the baseline to detect anomalous behaviour. Subsequently, the fingerprint analyser and forensic examiner modules examine the temporary fingerprints to determine the type of attack that has occurred based on a predefined subset of memory attack vectors.

6.2.1 PLC Memory Artefacts

As defined in Chapter 5, PLCs use different register areas to generate dynamic variables that can change during the operation of the physical process under control, and we denote R as the super-set of registers that a PLC uses. The PLC state formulation methodology proposed in Section 5.2 is used in this chapter to define R.

PLC Application Code

As presented in Chapter 4 under Artefact Data Type (ADT) 2 (see Section 4.3), PLC application code is the main user-defined program that PLCs cyclically execute to control physical processes and perform related functions. It is programmed using a standardised PLC programming language according to the international standard *ISO/IEC61131-3*, such as Ladder Logic, or Structured Text [37]. PLCs interface with sensors and actuators by assigning them to registers though this programmable logic, which is subsequently programmed by combining registers with program objects. The program objects determine how the register is

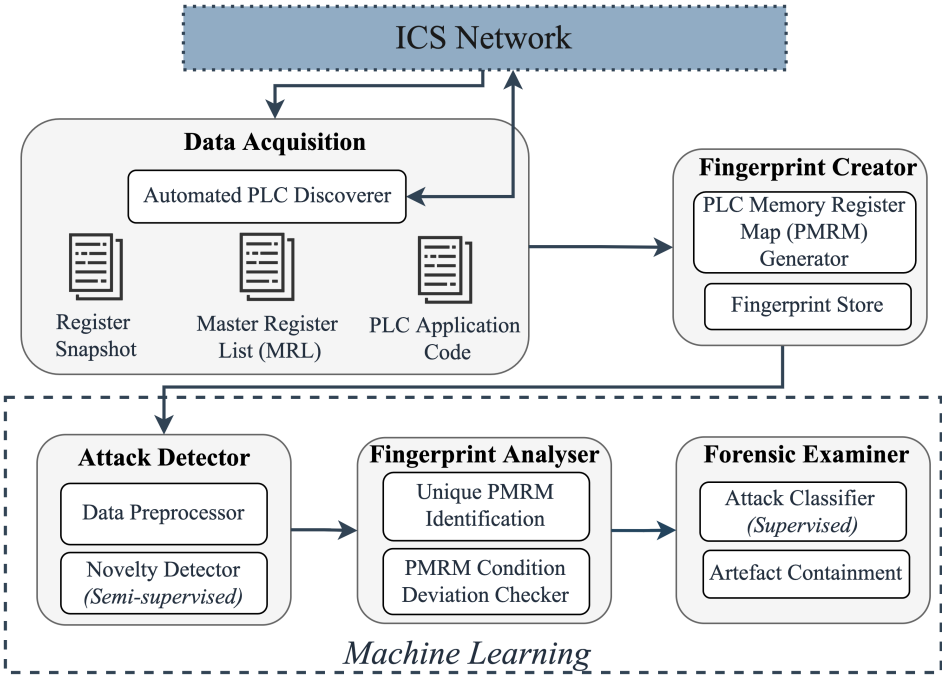


Figure 6.1: PLCPrint high-level architecture

used and ultimately the execution behaviour of the proceeding logic. Hence, we can examine PLC application code at a low-level to identify which registers are used within the code itself. We define an image of PLC application code as B. Each image B_i contains a set of static instances $F = \{f_1, f_2, \ldots, f_m\}$ where $m =$ total number of static instances within the application code. Therefore:

$$\forall r \in R \exists r \in f_m \tag{6.1}$$

6.2.2 PLCPrint Enumeration

The initial stage of PLCPrint involves lightweight automated asset discovery to identify the vendor and models of the PLCs that are accessible over the ICS network. To do this, existing functionality of a Python-based ICS network reconnaissance tool called PLCScan is extended [214]. Originally, PLCScan only reveals PLCs using the Siemens S7comm protocols or Modbus-TCP, however we extend the tool to also identify PLCs using the Common Industrial Protocol (CIP), specifically EtherNet/IP often used by Allen-Bradley PLCs. Specifically, PLCScan performs TCP/UDP port scanning through a pre-defined set of port numbers, such as TCP 102 for Siemens S7Communications and TCP 502 for the Modbus-

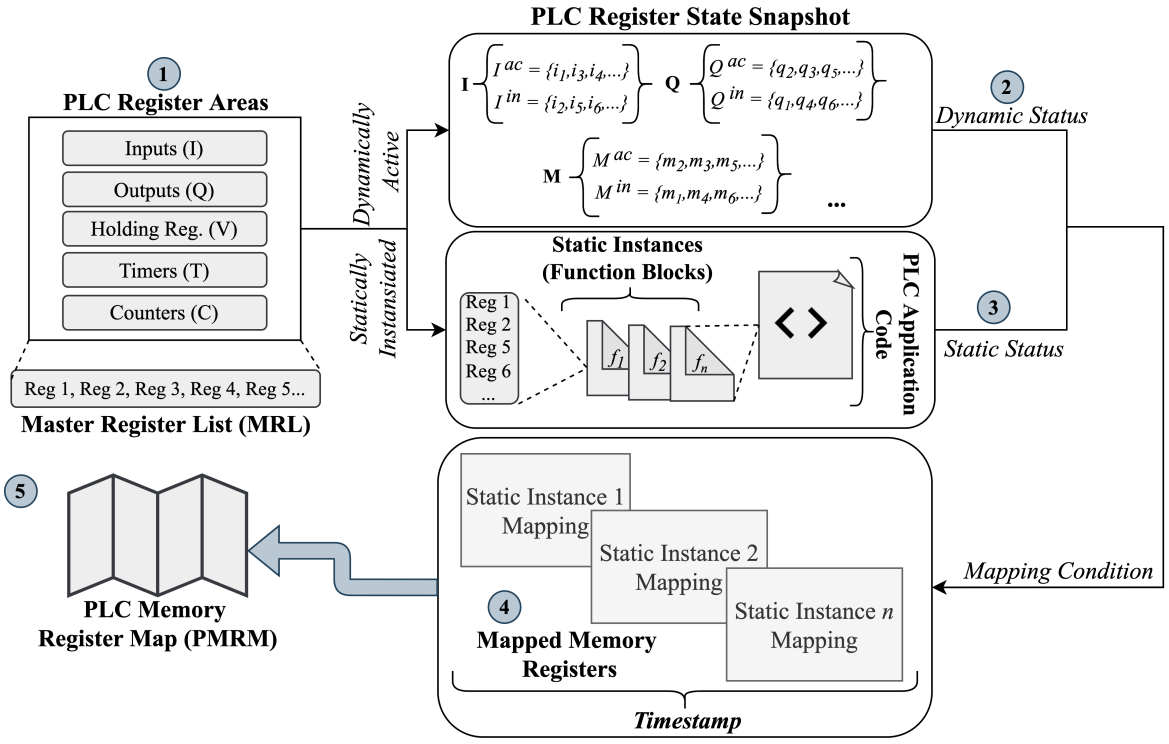


Figure 6.2: PLC Memory Register Map (PMRM) generation illustrating process of defining dynamic and static status for each register in the Master Register List (MRL)

TCP protocol. Therefore, we include TCP port 44818 which is used for messaging and data transfer by Allen-Bradley PLCs. Identifying the PLC vendor and model is an important first step in PLCPrint as although the general methodology is vendor-independent, different software scripts are required to access the memory artefacts from specific PLC models. The automated scan outputs a list of IP addresses and the associated PLC model. Each PLC is also given a unique identifier code so that PLCs of the same model can be distinguished.

PLCPrint defines the accessible registers for a PLC identified through the enumeration process, referred to as the Master Register List (MRL) within five memory register areas, I, Q, H, P, C from the PLC. A hybrid approach for defining the MRL is implemented where PLCPrint can either automatically populate the MRL from a predefined range based on the PLC model identified, or the user can manually enter the number of registers to capture. For our evaluations, we opted for the manual approach as it provided greater control when performing experiments on the ICS testbed.

6.2.3 PLC Data Artefact Acquisition

Figure 6.2 depicts the PLC memory register mapping process which fingerprints the changes in PLC memory artefacts while the PLC is in operation. Subsequent to defining the MRL, a register snapshot is acquired, which defines whether each individual register in the MRL is currently active or inactive at that point in time. Here a similar process to that presented in Section 5.3.1 is used to acquire the register shapshot from the PLC, specifically by issuing data requests through industrial communication protocols. Five binary vectors are returned, where each vector represents a PLC register area. As an example, a binary vector representing eight I registers $[1, 0, 0, 1, 0, 0, 0, 0]$ is read as registers $I0$ and $I3$ being active, and all other I registers inactive during a specific dynamic register snapshot.

An image of the application code is acquired from the PLC by issuing network commands to the PLC in a similar way to acquiring the register snapshot. Specifically, we utilise the Snap7 communications library that enabled data acquisition from Siemens PLCs running the S7comm protocol [144]. For the Allen-Bradley PLCs, we used an approach implemented in Chapter 4 that extended the PyLogix [143] communication library supporting the EtherNet/IP protocol (see Section 4.2.2).

6.2.4 Fingerprint Creator

The fingerprint creator module uses the acquired PLC memory artefacts to determine how the registers in the PLC MRL are related to these artefacts. We define each register as having a dynamic and a static status, which are determined by the registers existence in the register state snapshot and the PLC application code, respectively. The calculation of these status are then combined to determine the *Mapping Condition* (MC) of a register, resulting in a *PLC Memory Register Map* (PMRM).

As already mentioned, the initial stage of PLCPrint produces the MRL containing a list of registers in each of the five register areas. Every register has a dynamic nature allowing it to change during the operation of the PLC and so we refer to this as the register's *dynamic status*. The changing of register dynamic statuses is correlated with the operation of the physical process and execution of the PLC application code. Thus, the dynamic status of a given register is likely to change while the PLC controls kinetic activity of the physical

process, however transitions would be expected to be deterministic based on the execution of the PLC application code. Therefore, the dynamic status of a register may be 0 at time t_k but change to 1 at t_{k+1}.

Each register also has a *static status*, which is dictated by the register's relationship to the PLC's application code. The application code comprises one or more function blocks, herein referred to as static instances, that contain instructions written in one of the standardised PLC programming languages. Each static instance is then analysed to determine whether a PLC register exists within a particular static instance or not, which determines the register's static status. Separating the application code into individual static instances facilitates a greater level of granularity when performing forensic analysis in subsequent attack response stages discussed later.

Subsequent to a register being assigned both a dynamic status and static status, the final Mapping Condition (MC) of the register at time t is calculated. The dynamic and static statuses are cross-referenced resulting in the register being assigned one of four MCs, which are introduced below.

- **MC 1** - Registers that *do not* exist within a specific static instance of the PLC application code and *are not* currently active: $r_n \notin f_m \wedge r_n \notin R^{ac}$
- **MC 2** - Registers that *do* exist within a specific static instance of the PLC application code but *are not* currently activate: $r_n \in f_m \wedge r_n \notin R^{ac}$
- **MC 3** - Registers that *do not* exist within a specific static instance of the PLC application code but *are* currently active: $r_n \notin f_m \wedge r_n \in R^{ac}$
- **MC 4** - Registers that *do* exist within a specific static instance of the PLC application code and *are* currently active: $r_n \in f_m \wedge r_n \in R^{ac}$

6.2.5 Memory Fingerprint Generation

Subsequent to assigning an MC to every memory register in the PLC's MRL, PLCPrint groups all static instances under one timestamp t (i.e., stage 4 in figure 6.2), and outputs this as a single PMRM shown in stage 5. The timestamp is calculated at the start of stage 2 in the

mapping process and is used in subsequent stages of attack fingerprint analysis. Providing timestamps indicating when anomalies were first identified are also particularly beneficial to the incident response process, enabling investigators to collate additional data artefacts that were generated at a similar time. As a PLC typically has more than one possible state, multiple PMRMs are required to build a PLCPrint fingerprint. The process is repeated from stage 2 to 5 for *n* times, where *n* represents the number of PMRMs to be generated. The value of *n* should include the number of PMRMs required to map every PLC state change that is possible in the current system in relation to the physical process, and can either be set as a predefined fixed value or left unassigned. A multi-set of n PMRMs are then grouped into a single dataset to form a PLCPrint memory fingerprint for a particular PLC device. Once the fingerprint has been created, it is stored in the PLCPrint fingerprint store (database). The baseline fingerprint for a given PLC is shown as $finB_p$ where p represents the PLC.

6.2.6 Memory Fingerprint Attack Detection

The attack detection component of PLCPrint builds on the anomaly detection algorithm presented in Chapter 5 by using the PLC state formulation algorithm defined in Section 5.2. However, the detection algorithm presented in this chapter focuses solely on identifying anomalies resulting from attacks, and therefore utilises additional PLC memory artefacts in the composition of a PLC fingerprint $finB$, as discussed. As the overarching anomaly diagnosis framework comprises dependencies between its components, providing an independent attack detection methodology for the attack fingerprinting stage improves the modulairty of the anomaly diagnosis framework.

The memory fingerprint attack detection algorithm in PLCPrint instigates the continuous generation of a test fingerprint referred to as $finT$. Although $finT$ is created using the same process as $finB$, it is important to note that $finT$ is generated as a partial fingerprint, and is not stored in the fingerprint store at this stage. The attack detector comprises a data pre-processing module, which accesses the fingerprint store and retrieves the $finB$ dataset containing all of the PMRMs for a particular PLC. A $finT$ is then sampled at time intervals of t duration in minutes. $finB$ and $finT$ are used by a novelty detector as training and test datasets, respectively. If the detector identifies anomalous behaviour through the contents of the $finT$, it flags the data and passes the $finT$ to the fingerprint analyser module.

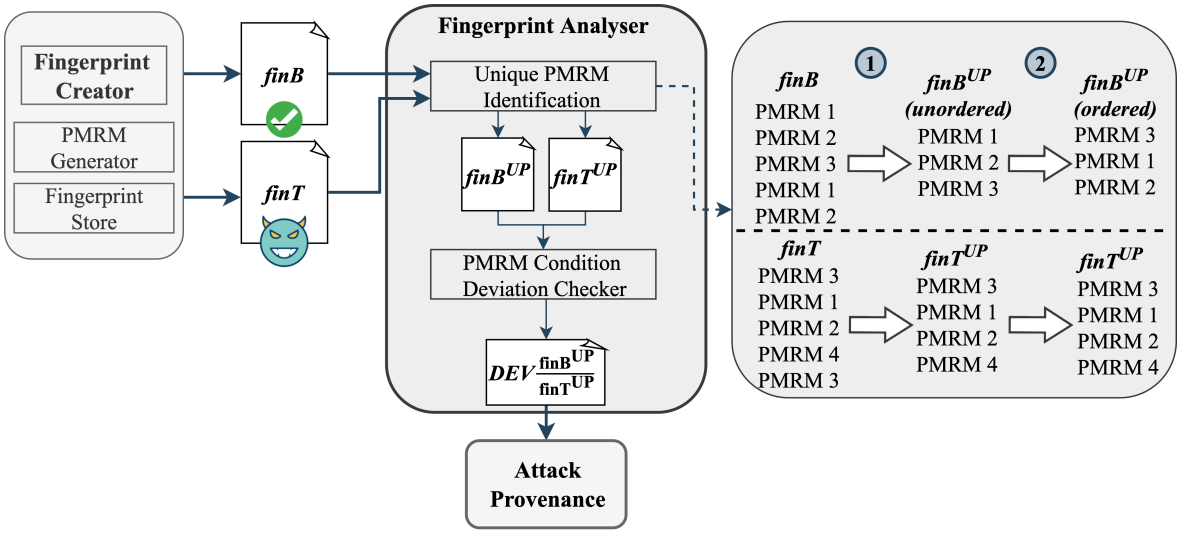


Figure 6.3: PLCPrint PLC Memory Register Mapping (PMRM) Analysis Process. The fingerprint creator generates the $finB$ baseline dataset and the $finT$ fingerprints (test dataset) which are transferred to the fingerprint analyser.

6.2.7 Fingerprint Analyser

The fingerprint analyser module is presented in figure 6.3. The module firstly examines both the baseline $finB$ and test $finT$ to convert their respective PMRM multi-sets into sets so that each element, in this case a unique PMRM, has a multiplicity of 1. Thus, we identify all of the unique PMRMs in $finB$ and $finT$, the quantity of which is dependent on the number of static instances derived from the PLC application code. Generating a set of unique PMRMs removes any repetition within the $finB$ and $finT$ fingerprints, enabling us to define a set of PMRM states that the underlying physical process, controlled by the PLC, can be in. We denote $finB^{UP}$ and $finT^{UP}$ as the identifiers for the unique set of PMRMs in $finB$ and $finT$, respectively. Identifying unique PMRMs in both the baseline and test fingerprints also enables common indexing when comparing the PMRMs in each file.

When $finB$ and $finT$ were generated, the physical process could have been at different stages. Using the water treatment process in GULP as an example, the generation of $finB$ might start during the filtration stage of the of the physical, however the generation of $finT$ might start at during the distribution stage. Consequently, the first index of each datasets would be different and incomparable. Once the sets of unique PMRM states have been created, PLCPrint compares the MC of each register in $finB^{UP}$ PMRMs with the registers at the same index in the $finT^{UP}$ PMRMs. Any deviations between MCs of a register are recorded. If a register's MC has deviated in the test PMRM from the baseline PMRM, we

then examines which MC the register has deviated to.

The comparison of the PMRMs in $finB$ with $finT$ enables PLCPrint to calculate the total average deviation for each of the four MCs that $finT$ has generated from $finB$. An overall rolling average deviation score is also calculated so that we can identify significant deviation changes. The deviation is used instead of a total count of each MC since, if we took only the latter, then it would be possible to miss changes that have occurred to individual registers. For example, if register *I2* had deviated from MC 1 to 3 but register *I3* changed from 3 to 1, then the total count for MCs 1 and 3 in this example would not change. Consequently, a difference in the total number of each MC would not be calculated. The fingerprint analyser outputs deviations found between the PMRMs in the $finB^{UP}$ and $finT^{UP}$, which is referred to as $DEV^{\frac{finB^{UP}}{finT^{up}}}$. This is the cumulative increase of deviations for each of the four MCs, using the timestamps as markers to calculate the seconds that have elapsed between the start and end of $finT$. In addition, a total deviation count and rolling average are also included.

6.2.8 Attack Provenance Analysis

The final but critical stage of PLCPrint is attack provenance process, which we also refer to as attack classification. This is responsible for performing the attack provenance through ML classification using the outputs from the fingerprint analyser module, as illustrated in figure 6.4. For model training and testing, datasets containing $DEV^{\frac{finB^{UP}}{finT^{up}}}$ derived from attack scenario experiments are first labeled with an attack type and then an attack technique, which

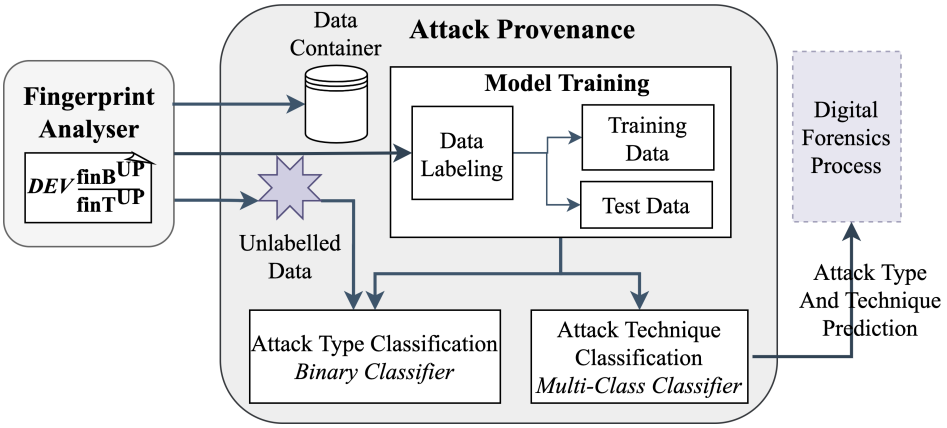


Figure 6.4: PLCPrint attack provenance process utilising supervised algorithms for binary classification and multi-class classification for attack type and attack technique predictions, respectively.

we then curate into sets of training and test data samples. Supervised binary classification algorithms are used to classify the attack type. A second stage of multi-class classification is performed to examine the specific attack techniques that were used to conduct the attack. To improve the forensic utility of PLCPrint, the forensic examiner also obtains a copy of the original $finT$ data sample and contains them as evidence in the fingerprint store along with a SHA-256 hash value to demonstrate data integrity. A copy of the PLC memory artefacts acquired to generate the $finT$ are also contained to maintain the chain of evidence and enable data provenance for subsequent forensic analysis, specifically deviations in PLC state behaviour and the PLC application code in hexadecimal format.

6.3 Evaluation

6.3.1 PLC Attack Scenarios

We revisit the discussion on PLC threat models and attack techniques in Chapter 3 and preset the specific attack vectors used in this chapter to evaluate the functional performance PLCPrint. The PLCPrint methodology is designed to detect and analyse attacks that target PLC memory artefacts which would impact the operational behaviour of the PLC and ultimately the underlying physical process. Therefore, all four attack scenarios defined in Section 3.4.3 are used to provide different cases of how PLC data can be manipulated. Using the comprehensive set of PLC attack techniques identified in Section 3.4.2, a subset of techniques that were identified to target PLC memory artefacts were selected and are presented in table 6.1.

The techniques have been grouped into two attack types depending on whether the technique targets memory artefacts that are not expected to change (*static*) and those that do change (*dynamic*) during normal PLC run-time. Static attacks involve code injection into the PLC either by forcing crafted control logic over the network into the non-volatile memory storage of the PLC. Conversely, dynamic attacks focus on manipulating the real-time values of PLC variables. With reference to the Artefact Data Types (ADT) proposed in Section 4.3, dynamic and static attacks target ADT 1 and ADT 2, respectively. In order to evaluate the high-level attack types and the techniques, the four original overarching PLC attack scenarios are

associated with their corresponding attack type, which is dependent on what areas of PLC memory the scenarios target. Based on the high-level threat model proposed in figure 3.6, we assert that both attack types can be performed by either an outsider or an insider threat actor. The attack scenarios are generalised and implemented against two PLC vendors operating in the GULP testbed.

We evaluate PLCPrint with two PLC models from different vendors to determine whether the PLC memory register mapping process can be generalised across PLC vendors. Specifically, we use Siemens S7-300 and Allen-Bradley ControlLogix 1756-L71 (AB CLX) PLCs. 400 individual attack scenario experiments were conducted over the GULP testbed, 200 scenarios for each PLC to generate datasets for performance evaluations. Each attack type (dynamic and static) has an equal number of attack scenarios executed, resulting in 200 scenarios for each attack type per PLC. Each attack scenario generated a test PLC memory fingerprint $finT$, which was subsequently compared with the original baseline fingerprint $finB$ for a particular PLC.

6.3.2 Evaluation Metric Specifics

We use the F1 Score, the Area Under the Curve (AUC) Receiver Operating Characteristic (ROC) curve, and accuracy to evaluate the memory fingerprinting detection algorithm, and the performance of attack classification, which have been defined earlier in Chapter 2. We identified that AUC-ROC was more informative than other metrics such as Precision-Recall Curve (PRC) due to the balanced class sizes of our data sets. Furthermore, we initially evaluated whether using the weighted *beta* (β) parameter of the F-score would provide more

Table 6.1: MITRE ICS ATT&CK Techniques Targeting PLC Memory Contents

Attack Type	**MITRE Technique**	**Attack Scenarios**
Static	Program Download (T0843)	A_1 and A_4
	Modify Program (T0889) (StaticMP)	
	Modify Controller Tasking (StaticMCT) (T0821)	
Dynamic	I/O Image (DynamicIO) (T0877)	A_2 and A_3
	Brute Force I/O (DynamicBF) (T0806)	

accurate results. A higher beta value can be more informative of classification model performance where low false-negatives quantities are critical. However, changes to the resulting F-score were insignificant when the beta parameter was tested. Hence, we use the default beta value of 1.0. Other evaluation metrics that could have been used include the Matthews Correlation Coefficient (MCC), which is particularly beneficial for highly unbalanced class sizes. However, we ensure that an equal number of both dynamic and static attack scenarios are performed to maintain a balanced dataset, and therefore the F1 and accuracy measures are preferred.

6.3.3 Memory Fingerprint Attack Detection Performance

We implement the memory fingerprinting attack detection approach through One Class Support Vector Machine (OCSVM) and κ-Nearest Neighbour (κ-NN) as semi-supervised learning. OCSVM has been used in many previous studies within the literature and so provides a good baseline comparative to related attack detection approaches. As the computation for κ-NN occurs when a classification or prediction is being made rather than at a training stage, the algorithm provides benefits for real-time systems since new training data can be continuously added. A range of test dataset sizes based on the number of generated PMRMs were also evaluated to determine how the sample size impacts the classification performance. Dynamic and static attack scenarios for each PLC were conducted for each test dataset size parameter.

Table 6.2: Memory Fingerprinting Attack Detection Results using OCSVM and κ-NN models - F1 and Accuracy (Acc) scores are out of 1.0, FNR scores are out of 100%

OCSVM						**κ-NN**					
Dynamic			*Static*			*Dynamic*			*Static*		
Acc	F1	FNR	Acc	F1	FNR	Acc	F1	FNR	Acc	F1	FNR
0.95	0.97	0.77%	0.94	0.97	0.74%	0.82	0.91	0.9%	0.94	0.97	0.89%

(a) S7-300 PLC

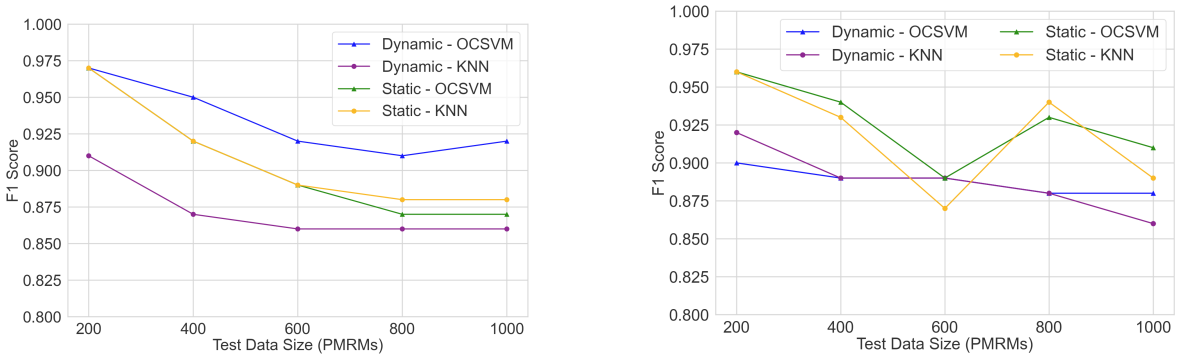
OCSVM						**κ-NN**					
Dynamic			*Static*			*Dynamic*			*Static*		
Acc	F1	FNR	Acc	F1	FNR	Acc	F1	FNR	Acc	F1	FNR
0.86	0.9	0.89%	0.93	0.96	0.76%	0.86	0.92	0.81%	0.93	0.96	0.78%

(b) AB CLX PLC

The accuracy and F1 results for attack detection are shown in table 6.2. Both PLCs show high performance achieving high average F1 scores of 0.96 and 0.94 across both classifiers and attack types for the S7-300 and AB CLX PLCs, respectively, demonstrating the generalisability of PLCPrint at detecting anomalous PLC behaviour. In most cases, PLCPrint attack detection performs best when the PLCs were subject to attack scenarios that utilised static techniques, possibly as such attacks typically comprise lower entropy regarding how PLC registers are manipulated. Dynamic attacks are more likely to include high entropy when targeting PLC registers and so are less deterministic. A full set of performance scores for the attack detection stage of PLCPrint can be found at the end of this thesis in Appendix A.

While both OCSVM and κ-NN models achieved high performance measures, OCSVM generally is more effective, indicated not only by the F1 scores, but also by the very low False Negative Rate (FNR) which is often considered a more important metric than false positives in the context of attack detection in order to mitigate the persistence of ongoing attacks. Overall, OCSVM achieves an average FNR = $< 0.8\%$, with the κ-NN classifier performing slightly worse but not significantly at FNR = $< 0.9\%$. High accuracy measures are also recorded, highlighting that attacks were detected positively in most cases.

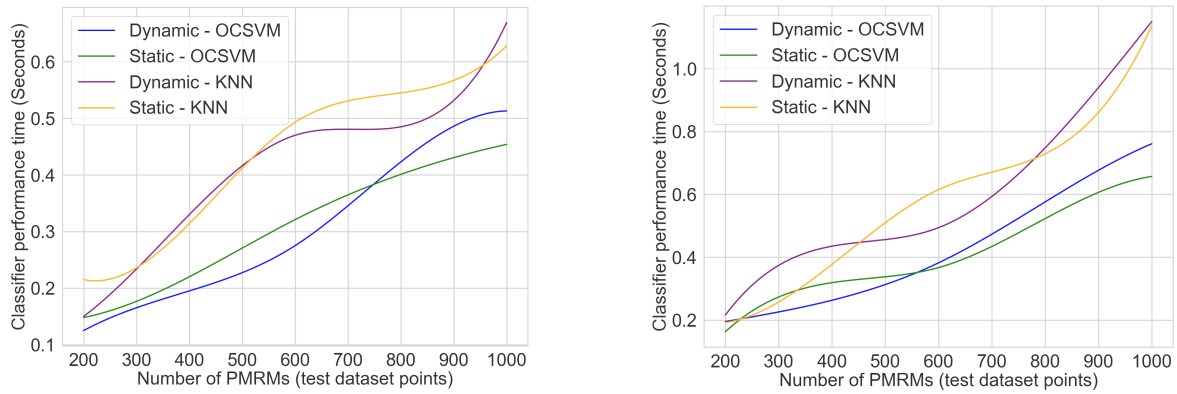
In addition to evaluating two classifiers, we also examine how the size of the test dataset impacts on the performance of detection. Size is represented by the number of PMRMs that a dataset contains. The detection performance of PLCPrint is poorer with larger test datasets containing more PMRMs that were previously unseen in the training dataset, as demonstrated in figure 6.5. However, PLCPrint is able to identify anomalous PLC activity with a very small sample of 200 PMRMs. In context, a baseline fingerprint generated by PLCPrint may contain several thousand PMRMs, depending on the complexity of the process the PLC is controlling. Thus, this highlights the sensitivity of the chosen novelty detection algorithms but emphasises that PLCPrint only required low amounts of anomalous data to detect an attack. Smaller sample sizes would benefit a real deployment of PLCPrint as it reduces the active data acquisition from PLCs that are often already resource constrained. Furthermore, the high detection performance scores achieved with smaller amounts of test PMRMs indicates that attacks are likely to be highly detectable early into their existence. In figure 6.5b specifically, we can identify a sudden dip in F1 performance when detecting static memory attacks for the AB CLX PLC. It is likely that this is caused by an increase in



(a) S7-300 PLC

(b) AB CLX PLC

Figure 6.5: F1 Scores for increasing number of test PMRMs during attack detection



(a) S7-300 PLC

(b) AB CLX PLC

Figure 6.6: Computational performance times in seconds for increasing number of test PMRMs during attack detection

unexpected noise within PMRMs of the test dataset, impacting the overall distribution of the deviations cause by the attack. As we will examine later in the attack classification evaluation (section 6.3.4), static attacks typically generate a wider dispersed quantity of deviations in PLC memory, and therefore are more likely to induce noise that will impact the performance of anomaly detection.

Computational performance of each classifier was evaluated by measuring the change in testing time (in seconds) with respect to the size of the test dataset. The setup specifications we use in these tests include a Windows 10 workstation with an Intel Core i5-5257U CPU (2.70 GHz) and 16 GB DDR3 RAM. For the optimal performing iteration where the number of PMRMs was 200, the average performance times of 0.16 and 0.19 seconds were achieved for the OCSVM and κ-NN classifiers, respectively. As illustrated in figure 6.6, the OCSVM classifier out-performed the κ-NN. Although it is to be expected that using smaller dataset sizes will increase computation performance, the significance here is that the very low times

indicate that attack detection can be achieved in under one second, further improving the suitability of OCSVM for lightweight anomaly detection and rapid incident triaging. As data set size is an important design consideration when deploying classifiers in real environments, we calculate the average time consumption percentage increase for each classifier model per every 100 PMRMs that a dataset size is increased by. We identified that, as expected from the graphs in figure 6.6, OCSVM has a lower average increase of 0.04s (40ms), compared with κ-NN which demonstrates an increase of 0.13s (130ms) per 100 PMRMs, and therefore OCSVM provides a more optimised solution for larger datasets. The provided results are cumulative averages from both PLC models. In general, as computations can be achieved in under 1 second in most cases, which is demonstrated by these results, PLCPrint can be considered to support real-time attack detection. Moreover, this is further emphasised by the high attack detection scores achieved with low quantities of PMRMs.

The previous analysis evaluates how the number of PMRMs within the test datasets impacts the performance of the attack detection methodology. From this, we see high performance particularly in smaller fingerprint sizes where fewer PMRMs are generated from the PLC. However, the attack scenarios in these cases consisted of a high range of attack permutations (A_p) to the PLC, leading to questions regarding how the attack detection performance changes with more subtle and stealthy PLC attacks that comprise smaller numbers of permutations. Subtle attacks are often used by adversaries to maintain a low-profile by inflicting a smaller number of changes into the ICS but still causing significant disruption and damage. Within the context of the GULP testbed, an example of a subtle attack would be forcing the disinfectant control valve open to flood the water supply with chemicals. As a single acutator

Table 6.3: Average accuracy and F1 scores for OCSVM across both PLC models with increases in attack permutations

A_p	2% (A_{p1})	10% (A_{p4})	15% (A_{p6})	25% (A_{p10})	50% (A_{p20})
Accuracy	0.95	0.956	0.955	0.944	0.957
F1	0.974	0.975	0.972	0.972	0.976

(a) Dynamic attack permutations

A_p	2% (A_{p1})	10% (A_{p4})	15% (A_{p6})	25% (A_{p10})	50% (A_{p20})
Accuracy	0.928	0.949	0.95	0.933	0.97
F1	0.956	0.961	0.962	0.961	0.985

(b) Static attack permutations

is being manipulated, the resulting PLC register states within the dataset would contain more subtle changes, in comparison to deactivating off all of the water pumps, for example.

Therefore, we now move to evaluate how the number of permutations in an attack scenario impacts the performance of the attack detection process. In this context, the number of permutations is equal to the percentage PLC variables that are manipulated from the total MRL, and the percentage change made to a static instance (function block) in the application code for dynamic and static attacks, respectively. For example, an attack consisting of a single variable manipulation, such as forcing a water pump off, is considered as a one permutation and therefore $A_p = A_{p1}$. Specifically for static attacks, we consider the percentage of code change from the original PLC application code, rather than numbers of variable permutations. From the evaluations of the attack detection methodology presented previously, we see that a strong case is presented to employing OCSVM as the primary classifier, particularly due to the generalisable performance across different PLC vendors and the stable performance provided with varying amounts of PMRMs within the test datasets for each attack type. Therefore, OCSVM is selected for the following evaluation.

A range of attack permutations are evaluated, up to 50% of either MRL manipulations or static instance changes in the application code. The accuracy and F1 performance scores, presented in table 6.3, represent the average for both PLC model separated on attack type. Both the accuracy and F1 scores are high and consistent for the different levels of permutations caused during the attack scenarios for each attack type, indicating that very subtle attacks consisting of a single permutation (2%) are highly detectable. Dynamic attack scenarios achieved slightly more consistent scores across the different permutation levels compared to static attacks. Furthermore, for static attacks there is an greater increase of 0.07 (7%) from highly subtle attacks represented by A_{p1} to attacks with a considerably higher number of permutations A_{p20}, suggesting that the attack detection methodology is more sensitive to stealthy attacks.

In table 6.4, we demonstrate how the performance of memory fingerprinting attack detection algorithm presented in this chapter compares against the detection performance measurements of several existing ICS anomaly detection studies. It is evident that while the approach is highly competitive with existing algorithms, it does not perform as well for detection as the PLC register state-based algorithm presented in Chapter 5. However, in comparison with

Table 6.4: Comparison between anomaly detection performance, measured with accuracy and F1-score (where available), for algorithms proposed in this thesis and those reported in the existing literature. Where more than one accuracy or F1 score is provided by the study, an average was used.

Approach	**Accuracy**	F1 **Score**
Yau & Chow (2017) [18]	0.95	0.83
Lin *et al* (2018) [66]	-	0.82
Chan *et al* (2019) [67]	-	0.94
Yang *et al* (2020) [103]	0.75	0.97
Ahmed *et al* (2021) [113]	0.96	-
Huang *et al* (2021) [209]	0.89	0.85
PLC Register State-based Anomaly Detection (Chapter 5)	0.98	0.99
Memory Fingerprinting Attack Detection	0.92	0.96

existing work, we demonstrate that the memory fingerprinting approach can be used to detect different types of PLC memory attack. Furthermore, as we will examine in the next section (section 6.3.4), the general methodology of PLCPrint also performs attack classification to improve incident triaging and response, which has not been explored by existing literature.

6.3.4 Attack Type Classification Performance

In this section, we evaluate performance of classifying different attack types set out in Section 6.3.1. The attack type classification is evaluated using the same 400 datasets acquired for the attack detection evaluation. To classify attack types, deviations in the four mapping conditions (MCs) features are used presented, as defined in Section 6.2.4.

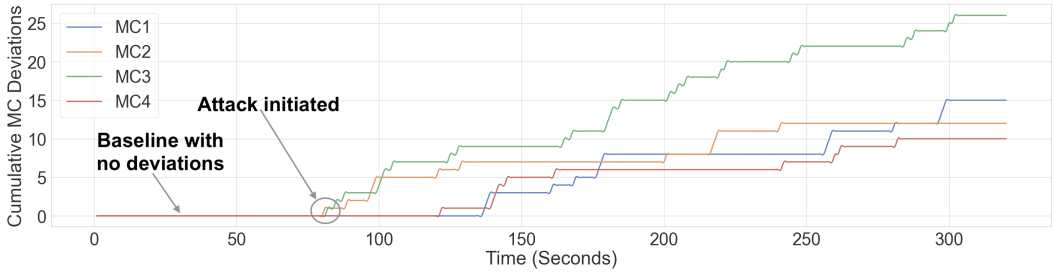
MC Deviation Features

We firstly examine the methodology of generating MC deviations, which was proposed in section 6.2.7 by evaluating how the quantity of deviations is dependent on the attack type being executed, and also how the cumulative total of each MC deviation correlates with time. Figure 6.7 illustrates how the four mapping conditions deviate over time when both the Siemens S7-300 and AB CLX 1756 PLCs are subject to dynamic and static attack types. Each sub figure represents the cumulative change in MC deviations in time series for dif-

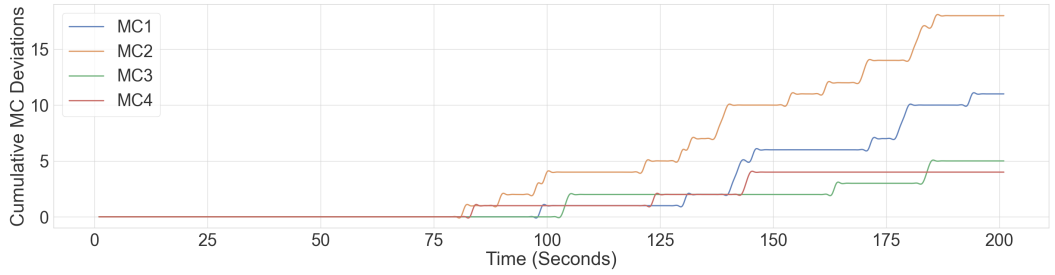
ferent attack types targeting the two PLC models. We see that no deviations are generated when the PLC is operating normally with no anomalous behaviour illustrated at the start of each figure before each attack is initiated and hence when the deviations increase. Through this initial analysis, it was identified that a significant increase in MC3 deviations occurred under dynamic attacks for both PLCs at the point where the attack took place, depicted in figure 6.7a and figure 6.7c. One reason for this is that dynamic attack scenarios do not manipulate the PLC's application code by changing the static declaration of registers, but instead dynamically activate or deactivate PLC registers.

Conversely, figure 6.7b and figure 6.7d present the MC deviations for a static injection attack scenario against the Siemens S7-300 and AB CLX 1756 PLCs, respectively. As static injection attacks do target modification to the PLC's application, unlike dynamic manipulation attacks, we are likely to see a change in which PLC registers are or are not statically instantiated within a specific static instance of the PLC application code. Hence, in comparison to dynamic attack scenarios, we see that static attack scenarios cause a significant increase in the deviations of MC2 and a lower deviation increase in MC3. We also see increases in MC1 for static attacks, which would occur in scenarios where registers are removed from static instances of the application code, resulting in the specific registers no longer being active or present in PLC memory. The significance of the increase in deviations of both MC2 and MC3 for static and dynamic attacks respectively highlights the importance of these deviations as features for attack classification.

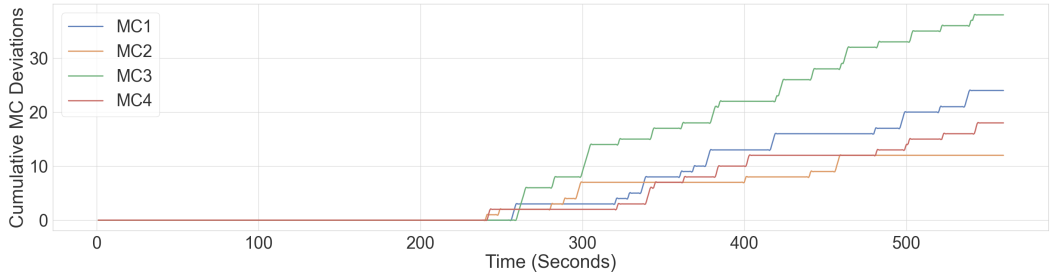
The specific number of deviations for each MC is not particularly important as this number will partially depend on the number of possible perturbations that an adversary can cause. For example, a PLC that has a high number of defined variables and registers in the MRL also has a high amount of potential MC deviations that can be caused during an attack. Furthermore, the real number of perturbations caused by a specific attack instance can also impact the number of perturbations, which is explored later in this section. The relationships between individual MC deviations when the PLCs are subject to dynamic and static attack scenario types are demonstrated in figure 6.8. The deviations have been normalised using a standard scalar from 0.0 to 1.0. Moreover, the diagonal graphs in figure 6.8 illustrate the distribution of MC deviations for static and dynamic PLC attacks. For both PLCs, there is considerable overlap of MC4 deviations compared with the remaining MCs, in particular



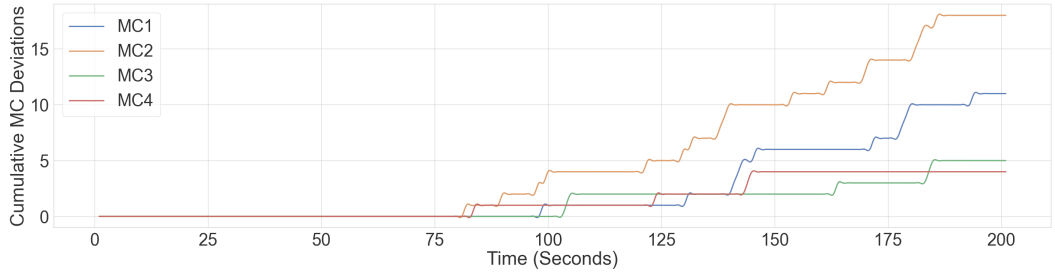
(a) S7-300 Dynamic Attack



(b) S7-300 Static Attack



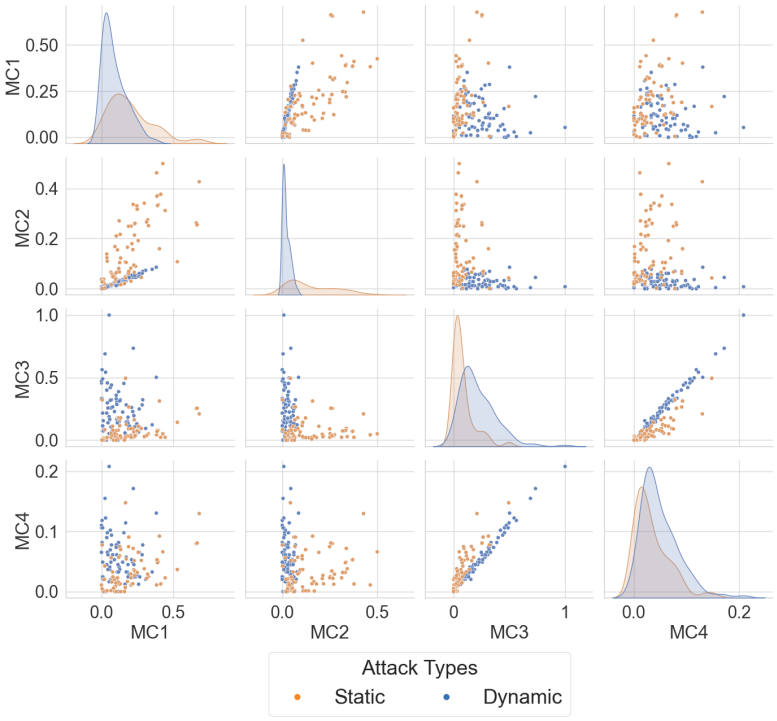
(c) AB CLX Dynamic Attack



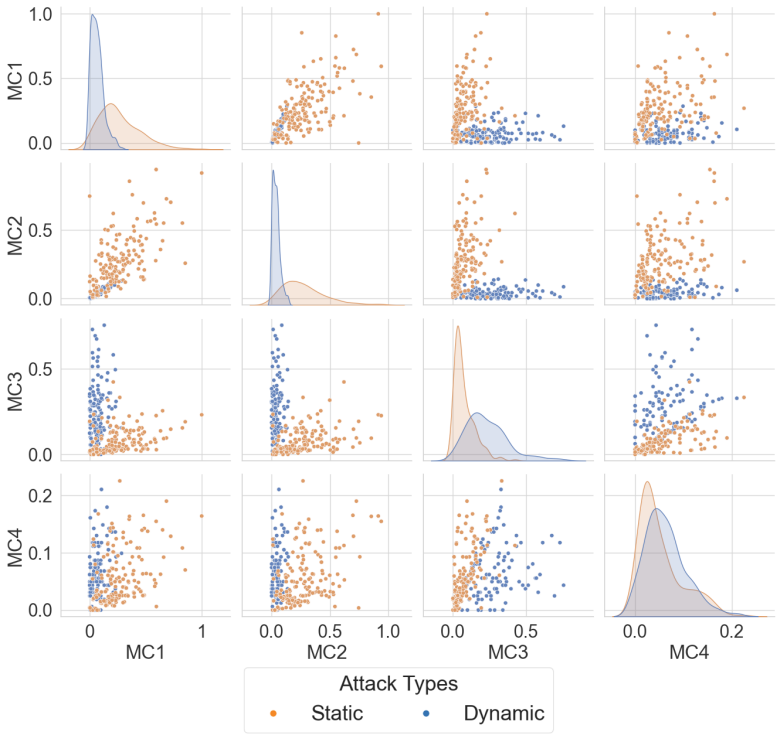
(d) AB CLX Static Attack

Figure 6.7: Cumulative Mapping condition (MC) deviations shown in time series (seconds) when PLCs are subject to dynamic and static memory attack scenarios. MC deviations increase as PLC registers move to different states.

MC2. Furthermore, within the plot graphs, we identify a strong relationship in both figure 6.8a figure 6.8b between MC2 and MC3 for both types of attack, emphasised by clearer separation of clustering. Relationships between deviations in other MC features, including MC 1 and 3, and between MC 3 and 4, also indicated high predictive potential for PLC attack classification. Additionally, we see very similar distributions of MC deviations are caused between the different PLC vendors as well, indicating that the attack type classifica-



(a) AB CLX 1756 PLC



(b) Siemens S7-300 PLC

Figure 6.8: Distribution of MC features influenced by attack type.

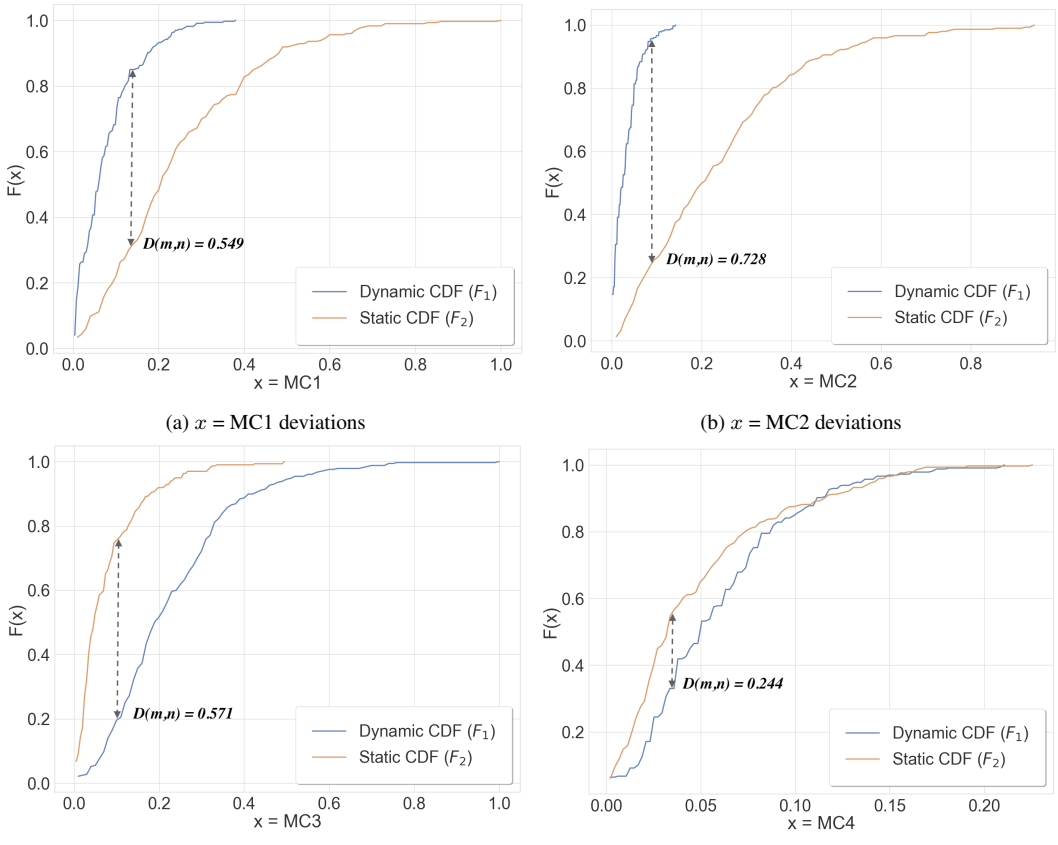
tion approach using MC deviations is highly generalisable, despite technical differences in how PLC models are programmed and configured.

The plots in figure 6.8 demonstrate clearer feature clustering in the MC deviation. To eval-

uate the probability that specific MC deviations will result from dynamic and static attack types, we employ Cumulative Distribution Functions (CDFs) the provide a sum of the probability density functions (PDF) at each data point. The datasets used are aggregated from the attack scenario data points produced by both PLC models in order to examine to what extent the MC deviations are generalisable to different PLC vendors. A two-sample Kolmogorov-Smirnov (K-S) statistical test is used to quantify the distances ($D(m,n)$) between the resulting CDFs for dynamic (F_1) and static (F_2) attack types for each MC feature (x), where m and n are the number of observations in F_1 and F_2, respectively:

$$D_{mn} = max|F_1(x) - F_2(x)| \tag{6.2}$$

From the CDFs illustrated in figure 6.9, we see that MC2 deviations (figure 6.9b produce a K-S value of $D(m,n) = 0.728$, which is higher than any other MC deviation feature suggesting that MC2 deviations are the most important in discriminating between dynamic and static



(a) x = MC1 deviations

(b) x = MC2 deviations

(c) x = MC3 deviations

(d) x = MC4 deviations

Figure 6.9: Gaussian cumulative distribution functions (CDF) of MC deviations for both PLC models under dynamic and static attacks

attack types. Specifically, with regards to MC2 deviations, it is highly probable that dynamic attacks will generate either zero or high quantities of very small deviations to MC2, whereas static attacks will produce high quantities of larger deviation values. Conversely, CDFs for F_1 and F_2 MC4 deviations (figure 6.9d) results in the lowest value of $D(m, n) = 0.244$, indicating that the distributions of MC4 deviations for dynamic and static attacks are more similar, and therefore less distinguishable.

Attack Classifier Selection

For the attack provenance phase of PLCPrint, we implement and evaluate the performance of five supervised ML classifiers, which are used to demonstrate the effectiveness of PMRM generation and MC deviation features for discriminating between dynamic and static attack types. Specifically, Logistic Regression (LR), K-Nearest Neighbour (κ-NN), Gaussian Naive Bayes (GNB), Support Vector Machine (SVM) and Random Forrest (RF) classifiers are chosen due to their general applicability to smaller feature sets and low overhead. As we will see later in Section 6.3.5, we approach the classification and provenance of attack techniques as a multi-class classification problem, and hence all five classifiers were also selected based on their suitability to such tasks [211] [215]. SVM and RF have all been extensively used in previous research and shown to be particularly suitable to the deterministic nature of ICS datasets. Moreover, RF has been demonstrated to be computationally efficient and resistant to large quantities of noise within datasets [215]. Additionally, RF particularly strong at handling larger dataset sizes, and therefore provides a good comparison to algorithms such as κ-NN, regarding both classifier accuracy and computational performance. As we have seen in figure 6.8, the relationship between MC deviations is often non-linear, and therefore

Table 6.5: Selected optimal parameters for classifiers resulting from grid search hyper-parameter tuning

Classifier	Hyper-Parameters
LR	**solver**: Newton-cg
κ-NN	**weights**: distance, **p**: 2
GNB	**var smoothing**: $1e^{-9}$ (default)
SVM	**C**: 0.4, **kernel**: rbf, **gamma**: scale, **tol**: 0.001
RF	**random state**: 0, n **estimators**: 50, **criterion**: entropy

κ-NN is selected due to its suitability for such datasets, in addition to the advantages discussed earlier in Section 6.3.3. However, while we expect the performance of κ-NN to be high for attack classification, the main disadvantage is the computation performance time, particularly with larger datasets as seen in Section 5.4.2 when compared with OCSVM for semi-supervised learning. We performed hyper-parameter tuning on all five classifiers to determine the optimal set of parameters. A grid search was performed instead of a random search due to the low feature set size comprising the four MC deviations, and the resulting chosen hyper-parameters are presented in table 6.5. For LR, the *Newton-cg* solver is selected as we will perform multi-class classification for attack techniques and the datasets have low-dimentionality. Moreover, we use a power parameter (p) of 2 for κ-NN to use Euclidean Distance, which is also preferable in use cases with low-dimentionality [216].

Performance Vs. Dataset Size

Firstly, the accuracy of attack type classification for each classifier is examined, specifically through a k-Fold cross-validation test to establish how different portions of training and test data impact the overall performance. A range of seven k-fold values were selected to provide a range of different dataset sizes. The results illustrated in table 6.6 demonstrate that the accuracy of each chosen classifier function is highly consistent with the range of data divisions, specifically we see very low maximum deltas (δ) of 0.01 (1.0%) and 0.053 (5.3%) for the Siemens and Allen-Bradley PLC models, respectively. Therefore, the performance of classifying different attack types is not dependent upon the size of the dataset itself. Hence, our approach provides benefits for practical applications in the cases when limited data is available for training the classifier, which is a common general challenge within the area of ML for ICS. Regarding classifier performance, the LR and RF classifiers achieved high consistent accuracy for both PLCs, reaching high metrics of 0.969 and 0.965, respectively. The high performance across different vendors of PLC with similar classifier models demonstrates that the ML model selection is also generalisable, in addition to the overarching attack fingerprinting methodology, which is particularly beneficial for real-world deployment.

Table 6.6: k-fold cross validation for attack type classification on MC deviation datasets for both PLC models (**bold** represents highest accuracy score for each PLC)

	Classifier Accuracy				
k-fold	LR	κ-NN	GNB	SVM	RF
2	0.938	0.960	0.905	0.945	0.963
3	0.945	0.963	0.910	0.940	0.960
5	0.940	0.958	0.915	0.940	**0.965**
10	0.940	0.958	0.913	0.940	**0.965**
15	0.938	0.957	0.913	0.943	0.961
20	0.938	0.958	0.913	0.940	0.958
30	0.938	0.957	0.915	0.940	0.960
δ	*0.007*	*0.006*	*0.01*	*0.005*	*0.005*

(a) S7-300 PLC

	Classifier Accuracy				
LR	κ-NN	GNB	SVM	RF
0.964	0.892	0.807	0.839	0.937
0.969	0.915	0.807	0.856	0.937
0.964	0.906	0.812	0.866	0.933
0.967	0.919	0.802	0.892	0.923
0.964	0.928	0.812	0.884	0.937
0.964	0.932	0.808	0.874	0.928
0.963	0.927	0.808	0.883	0.938
0.006	*0.04*	*0.01*	*0.053*	*0.015*

(b) AB-CLX PLC

Performance Vs. Number of Permutations

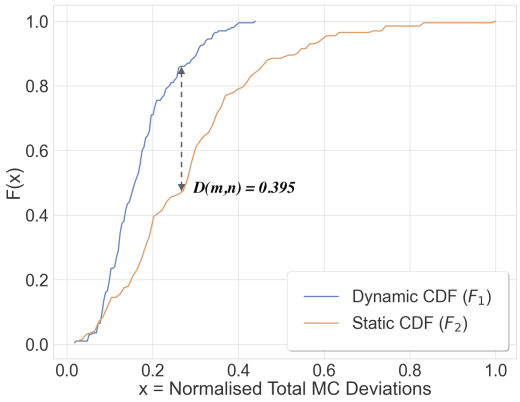
Similarly to the attack detection evaluation, it is important to examine how more subtle attacks inflicting fewer perturbations can impact the performance of attack fingerprinting. We have seen in figure 6.7 that deviations increase as an attack is executed and the PLC changes into different states that are considered abnormal. Furthermore, as the number of MC deviations that are identified in the $finT$ is positively correlated with the number of permutations that are implemented by the attacker, we quantify the subtlety of an attack as the amount of MC deviations within the context of attack type classification. Using the full datasets comprising all 200 attack scenarios for each PLC, the total number of deviations for each individual attack scenario is first calculated. These values are then normalised using a scaler function to generate a range of total deviations between 0.0 and 1.0. We perform normalisation as the specific number of deviations is not particularly important, as this value is dependent on the PLC models being used and the complexity of the underlying physical process. For example, one PLC might use only three registers and therefore the maximum number of possible deviations would be smaller than a PLC that uses thirty registers.

We firstly examine how the performance of attack type classification is impacted when the total deviation values are used as a single feature dataset instead of using the values of the four MCs as individual features. The results illustrated in table 6.7 demonstrate that using the total amount of deviations on it's own for attack detection does not perform as well compared with some of the individual MC features, in particular MC2 and MC3, achieving maximum accuracy scores of 0.78 and 0.75 for the S7-300 and AB CLX PLCs, respectively. One reason

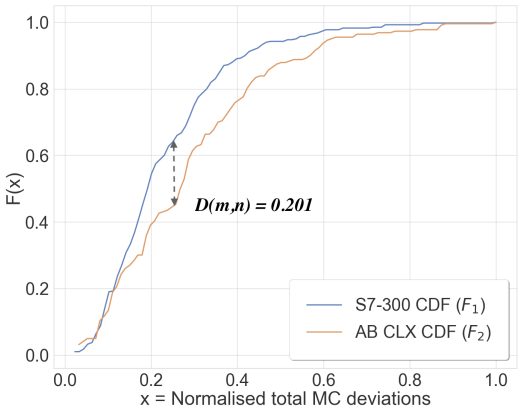
Table 6.7: Accuracy results for attack type classification using total deviations as single feature for attack discrimination

	S7-300			AB CLX		
Classifier	Dyn	Sta	Acc	Dyn	Sta	Acc
LR	0.76	0.71	0.73	0.71	0.69	0.65
κ-NN	**0.81**	**0.74**	**0.78**	0.76	0.71	0.74
GNB	0.78	0.69	0.76	0.69	0.61	0.63
SVM	0.75	0.69	0.71	0.69	0.64	0.66
RF	0.75	0.72	0.72	**0.77**	**0.72**	**0.75**

for this is that the distribution of data points between dynamic and static attacks is much tighter than MC2 and MC3 individually, and therefore the ability to discriminate between different attack types is more challenging, as illustrated by the CDFs plotted in figure 6.10a. The resulting lower K-S value of 0.395 for $D(m,n)$ between the dynamic and static CDFs is an additional indication that using the total count of deviations is not as powerful in attack type classification compared with MC1, MC2 and MC3 features. The low score here implies that the primary strength of the attack type classification algorithm lies within contrasting distributions of the individual MC features, rather than the entire feature set as a whole. Moreover, the CDFs presented in figure 6.10b, indicate that unique fingerprints are able to be generated for individual PLC models by using total MC deviations, providing a K-S value of $D(m,n) = 0.201$ between the CDFs for F_1 and F_2. Therefore, in addition to distinguishing between attack types, we can also use signatures in PLC memory artefacts, which are produced from the nuances resulting from how specific PLCs are programmed



(a) CDFs of dynamic and static attacks for both PLCs with KS statistic of $D(m,n)$



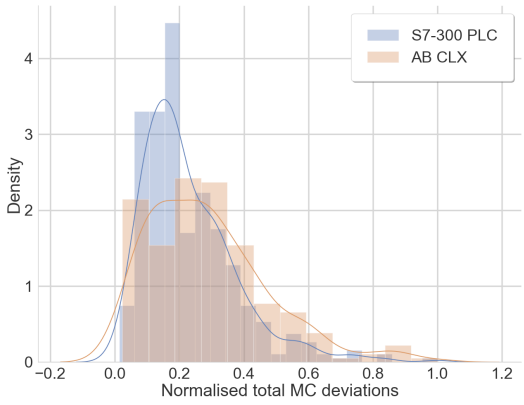
(b) CDFs of different PLC vendors with KS statistic of $D(m,n)$ demonstrating that individual models can generate unique fingerprints

Figure 6.10: CDFs and data distribution for performance of normalised total MC deviation values in attack type classification

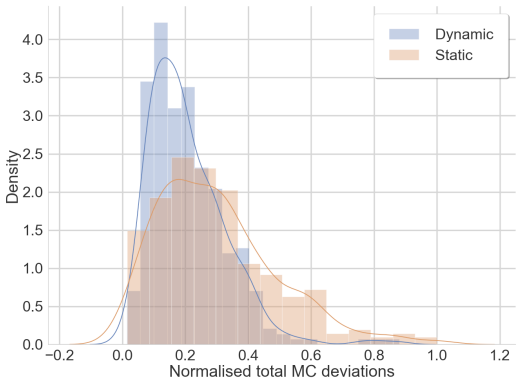
using registers and tags. Hence, these signatures are specific to different vendors and can therefore be used to identify which PLC has been targeted in an attack. This further enables root-cause analysis and incident triaging in for PLC. In general, the specific identification of targeted PLC models supports the scalability of the the overarching anomaly diagnosis framework, where tens to hundreds of PLC could be used within a real ICS environment.

Furthermore, dynamic attacks are more clearly identified by both PLCs reaching 0.81 and 0.77 compared with 0.74 and 0.72. We can see in figure 6.11b that the higher densities of low total deviation values between 0.1 and 0.4 caused by dynamic attacks provides greater characteristics for attack type discrimination for the classification algorithms. In general, dynamic attacks typically produce lower number of deviations represented by the high density points between approximately 0.05 and 0.25 normalised total MC deviations, and comprise a very low density towards the higher end of the scale. Conversely, static attack scenarios provide a greater distribution of total MC deviations and are lower their dynamic counterparts in most cases. However, we see that towards 0.5, static attacks provide much higher quantities of data points that contain high numbers of MC deviations.

Having evaluated the how the total deviations values impact the performance of attack type classification, we now move to evaluate to what extent more subtle attack scenarios can be accurately classified, and what the subsequent impact is on the performance. In other words, is the performance of the classification algorithms at all correlated with the total number of MC deviations in an attack sample. The full dataset comprising both training and test

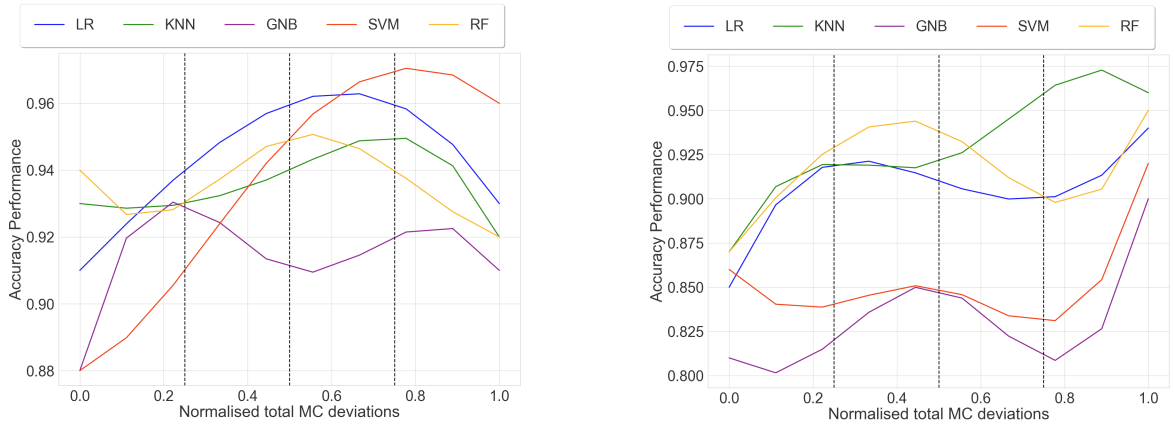


(a) Data distribution of normalised total MC deviation values for different PLC models



(b) Data distribution of normalised total MC deviation values for dynamic and static attacks aggregated for both PLC models

Figure 6.11: Data distribution of normalised total MC deviation values in attack type classification



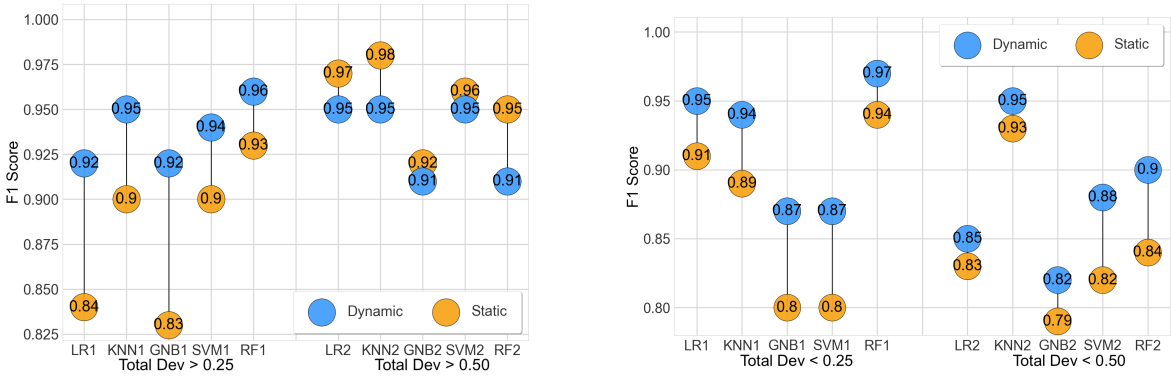
(a) S7-300 PLC

(b) AB CLX 1756 PLC

Figure 6.12: Performance of attack type classification as number of total deviations increases.

data is firstly separated into groups, which are determined by the normalised range of total MC deviations for each individual data point. Four groups were generated with upper set-points of $0.25, 0.5, 0.75$, and 1.0. For the subsequent evaluations, all four MC deviations are used as features when performing attack type classification, and not the total MC deviation values as presented in table 6.7. The individual PLC models are evaluated separately to examine if the classification of subtle attacks is vendor independent, and whether there are any major performance differences between dynamic and static attack types. The distribution of normalised total MC deviations for each PLC model is illustrated in figure 6.11a where we see that the S7-300 PLC generated high quantities of low total deviations compared with the AB CLX PLC, which is likely because of the different ways in which the PLC models are programmed.

Figure 6.12 illustrates how accuracy performance changes for each classifier as the number of total deviations within attack attack points increases, fundamentally demonstrating a high degree of varying performance levels for different classifiers when subject to increasing values of total MC distributions. Generally, performance does increase with the number of deviations that are generated, however it is not a clear linear increase. Specifically, we see that for very subtle attacks that inflict smaller manipulations to PLC memory states and thus result in smaller numbers of deviations where the total is < 0.25, classifier accuracy is higher that 0.88, particularly for the LR, κ-NN and RF classifiers where performance is higher than 0.90 for all four groups of total deviation values with the Siemens S7-300 PLC (figure 6.12a. The accuracy for the AB CLX PLC illustrated in figure 6.12b, does not perform as well



(a) S7-300 PLC

(b) AB CLX 1756 PLC

Figure 6.13: Distribution of MC features influenced by attack type.

initially when the total number of MC deviations is very low, however the LR, κ-NN and RF classifiers rapidly climb as the deviation total reaches 0.2. Therefore, we can have high confidence that very subtle attacks will be correctly identified, and that the classification algorithms are generalisable between PLC vendors. If we take these three classifiers since they are the top performing and more consistent with different quantities of MC deviations, we see that there are varying levels of performance for different amounts of total MC distributions, and therefore there is no clear best quantity for discriminating between dynamic and static attacks. Specifically, we see that attack classification for the Siemens PLC performs particularly well between total MC deviation quantities of 0.50 to 0.75, whereas the classifiers used with the AB CLX PLC data all show a spike in performance towards 0.95 total MC deviations. Moreover, we see that the κ-NN and RF classifiers for the S7-300 PLC perform better when subject to more subtle attacks with fewer MC deviations compared with attacks that will cause a greater number of permutations within PLC memory artefacts, such as registers and the application code.

Furthermore, figure 6.13 displays the F1 performance for dynamic and static attack type classification for both PLC vendors. When total MC deviations are < 0.25, the classification performance of dynamic attacks is higher for both PLCs and all classifiers, indicating that subtle dynamic attacks produce clearer permutations than static attacks, as supported by the distributions illustrated in figure 6.11b. Conversely, we see that for MC deviations > 0.25 and < 0.50, attack type classification performance is much closer, and higher for static attacks specifically for the S7-300 PLC.

As the total number of deviations varies depending on the number of perturbations inflicted

by an attacker and the specific logical states of a particular PLC, some attack scenarios will exhibit very few deviations. When examining the time series elements, a particular question arises which is, at what stage during an attack do we reach the optimal performance for attack type classification. In other words, how early into the detection of a given attack scenario can the classifier models provide high confidence on the type of attack being conducted. The analysis of total MC deviations that has been performed within this section addresses this question as the very initial stages of an attack will produce a low amount of total deviations starting from zero, since the baseline of normal operating behaviour comprises no MC deviations, and increasing thereafter demonstrated in figure 6.7. The resulting high accuracy for attacks with low quantities of MC deviations infers that early stages of an attack, which ultimately comprise low values for cumulative deviations, will also provide high accuracy for attack type classification. Therefore, PLCPrint and the attack fingerprinting phase of the overarching PLC anomaly diagnosis framework is able to rapidly triage attacks, enabling system recovery and restoration.

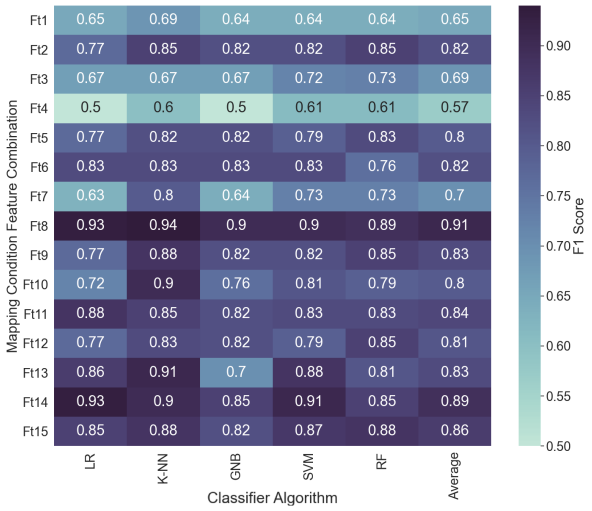
MC Feature Combination Evaluation

While some of the combinations of MC features illustrated in figure 6.8 display clearer clustering for classifying different types of PLC attack, it is not the case for all of the MC pairs shown in the graphs. To better understand the attack classification performance of each of the four MCs, we evaluate each MC individually and as a feature set combination through the five supervised ML classifiers. We arranged 15 feature sets based on the different possible MC combinations, which are presented in table 6.8.

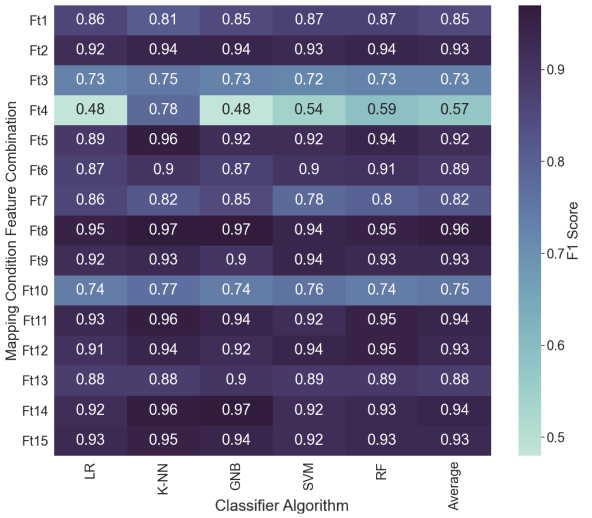
The subsequent F1 score results are presented as two heat-maps illustrated in figure 6.14a and figure 6.14b for the AB CLX and Siemens S7-300 PLCs, respectively. Each performance

Table 6.8: Mapping Condition (MC) Feature Set Combinations.

	Feature Set														
MC Features	Ft1	Ft2	Ft3	Ft4	Ft5	Ft6	Ft7	Ft8	Ft9	Ft10	Ft11	Ft12	Ft13	Ft14	Ft15
MC1	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗	✓
MC2	✗	✓	✗	✗	✓	✗	✗	✓	✓	✗	✓	✓	✗	✓	✓
MC3	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✓	✗	✓	✓	✓
MC4	✗	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗	✓	✓	✓	✓



(a) AB CLX PLC.



(b) S7-300 PLC.

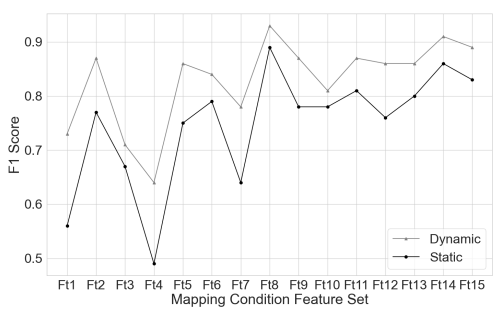
Figure 6.14: Heatmaps showing the average F1 scores for each classifier with each MC feature set as described in table 6.8. F1 scores are an average of the performance at classifying dynamic and static attack scenarios. An overall average for each feature set is also included. Classifier abbreviations are: *LR* = Logistic Regression; *κ-NN* = K-Nearest Neighbour; *GNB* = Gaussian Naive Bayes; *SVM* = Support Vector Machine; *RF* = Random Forrest.

test provides two F1 scores for classifying dynamic and static attack scenarios, respectively. The F1 scores in figure 6.14 represent the harmonic mean composed of the dynamic and static F1 scores for each tested classifier and feature set combination (Ft1 - Ft15 defined in table 6.8). A total average for each feature set is also included, which is composed of the F1 scores from the individual classifiers. Generally, the classifiers perform better for the S7-300 PLC compared to the AB CLX PLC, although similar feature sets provided similar scales of

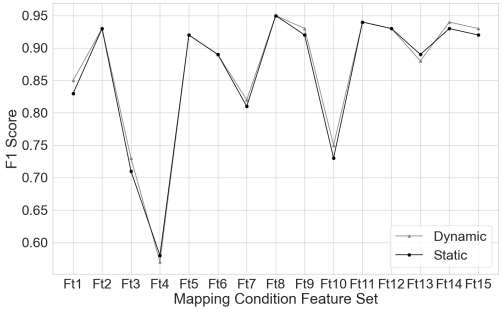
performance for both PLCs. Feature set Ft8, which combines MC2 and MC3, produces the highest performance for attack type classification for both PLCs, reaching F1 scores with the κ-NN classifier of 0.94 for the AB CLX and 0.97 for the S7-300 PLC. The datasets generated by PLCPrint are particularly suited to κ-NN, which results in high accuracy for most of the feature sets for both PLCs. The F1 scores from Ft8 reinforce our initial hypothesis that was identified from analysing the graphs in figure 6.8. Conversely, we can see from the results that Ft4, comprising solely MC 4, was the weakest feature set for classifying different attack types scoring an average F1 score of 0.57 for both PLCs. Interestingly, increasing the feature set size does not strongly correlate with the overall performance of the classifier, with some larger feature sets such as Ft10 and Ft13 for the S7-300 PLC in figure 6.14b performing worse than smaller feature sets. Moreover, feature sets that do not contain MC 2 as a feature generally perform worse than those that do contain it, emphasising the importance of static statuses when performing PLC attack classification.

Figure 6.15 depicts the average F1 and AUC-ROC scores composed from the results of the aforementioned five classifier algorithms. The graphs in figure 6.15a and figure 6.15b illustrate the average F1 scores for dynamic and static attack type classifications, and the performance of each attack type based on the chosen feature set. Generally, dynamic attack scenarios result in higher F1 scores for classification performance compared with static attacks with both PLCs. One justification for the differences in F1 scores is that static attacks produced greater distribution in MC deviations since they can induce changes in both the dynamic and static status of a register. Conversely, dynamic attacks can only influence a register's dynamic status. For the AB CLX PLC, the classification of dynamic attacks always out-performed the classification of static attacks, however this is not the case for the S7-300 PLC, where the F1 scores are often reciprocal. In particular, static attacks are classified more accurately for the S7-300 PLC. It is likely that this is due to a larger number of register perturbations being possible on the S7-300 PLC compared with the AB CLX, which is a consequence of how the PLCs were programmed to control the GULP testbed physical process.

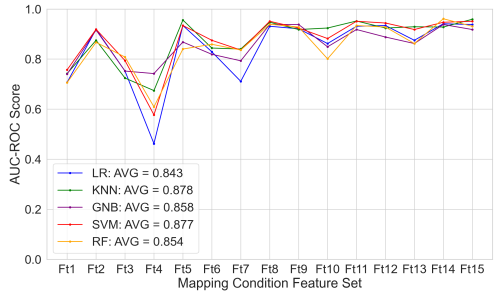
The graphs in figure 6.15c and figure 6.15d illustrate the attack classification AUC-ROC scores for each classifier when different feature sets are used. The κ-NN classifier performs best with a slightly higher AUC score for the AB CLX PLC and significantly increased for



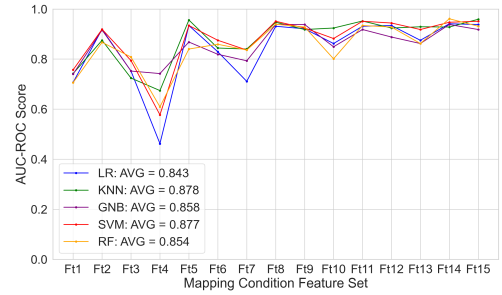
(a) AB CLX PLC F1 Scores.



(b) S7-300 PLC F1 Scores.



(c) AB CLX PLC AUC-ROC Scores.



(d) S7-300 PLC AUC-ROC Scores.

Figure 6.15: Attack classification performance of each feature set for both PLC models: a & b) Average F1 Scores composed from 5 classifier algorithms for dynamic and static attack scenarios; and c & d) AUC-ROC scores for each classifier.

the S7-300 PLC. The performance of all five classifiers generally improves as the feature set sizes increased, with the exceptions of Ft4, Ft7, Ft10 and Ft13, which all contain MC4, identified to be the weakest feature, and omit the strongest feature, MC2. From the perspective of deploying PLCPrint as a component in the wider anomaly diagnosis framework in a real-world ICS, the κ-NN algorithm is particularly beneficial as it involves instance-based learning and can adapt to new training data as it is collected over long periods of time. Thus, κ-NN can respond quickly to input changes during real-time.

Computational Performance

To highlight the rapidity of attack type classification to assist with incident triaging, we evaluate the computational performance of PLCPrint. The consumption time when performing classification testing for the five classifiers is compared for both PLC models, demonstrated in figure 6.16. Here we see that the κ-NN classifier has the lowest test time of approximately 0.3 seconds for both PLCs, which is particularly beneficial as κ-NN also achieved the highest classification AUC-ROC and F1 scores, demonstrated in figure 6.15 and figure 6.14, respec-

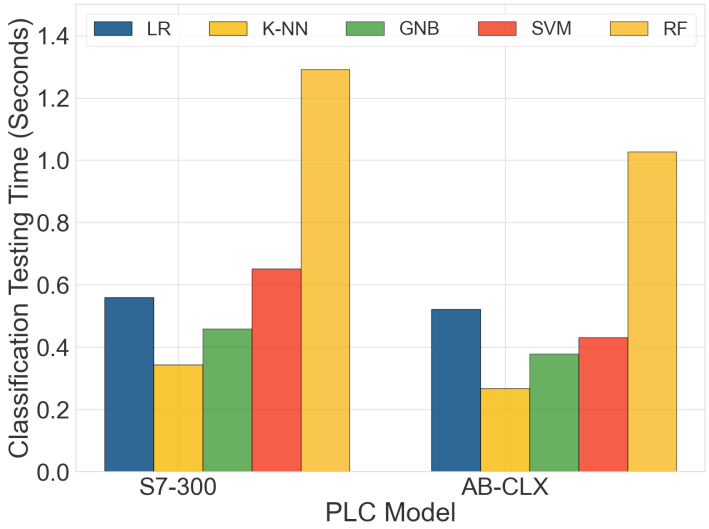


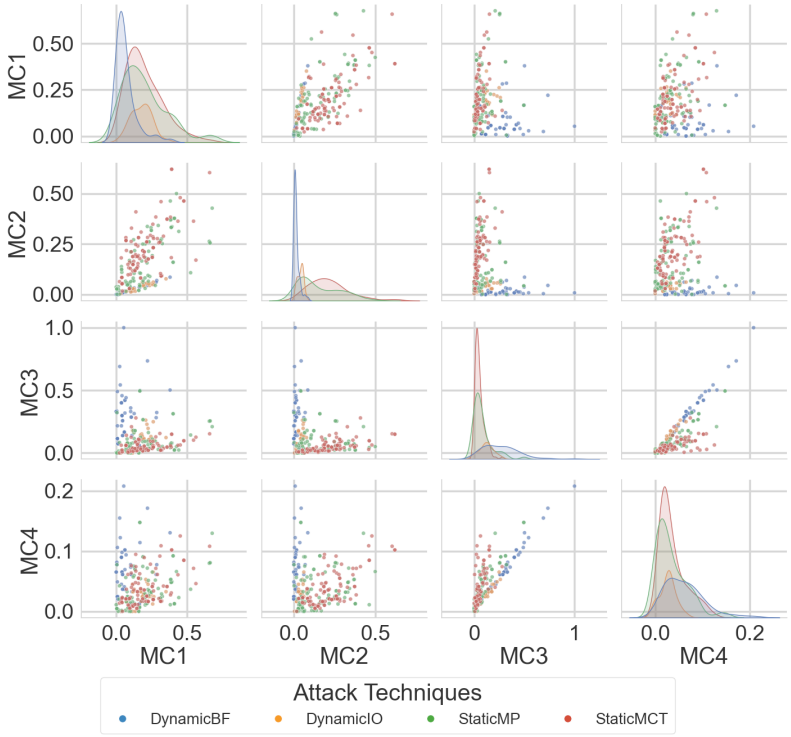
Figure 6.16: Time consumption of attack type classification testing using full feature set for both PLC models

tively. In comparison, the RF classifier took longer to test the same dataset for attack type classification, taking over 1.1 seconds, although this is still results in a very effective classification time. Similarly to time required to perform attack detection, presented earlier in figure 6.6, we also achieve for attack classification within the scope of real-time with most classifiers performing in under 1 second. The time consumption demonstrated here are with the full feature set comprising all four MC variables.

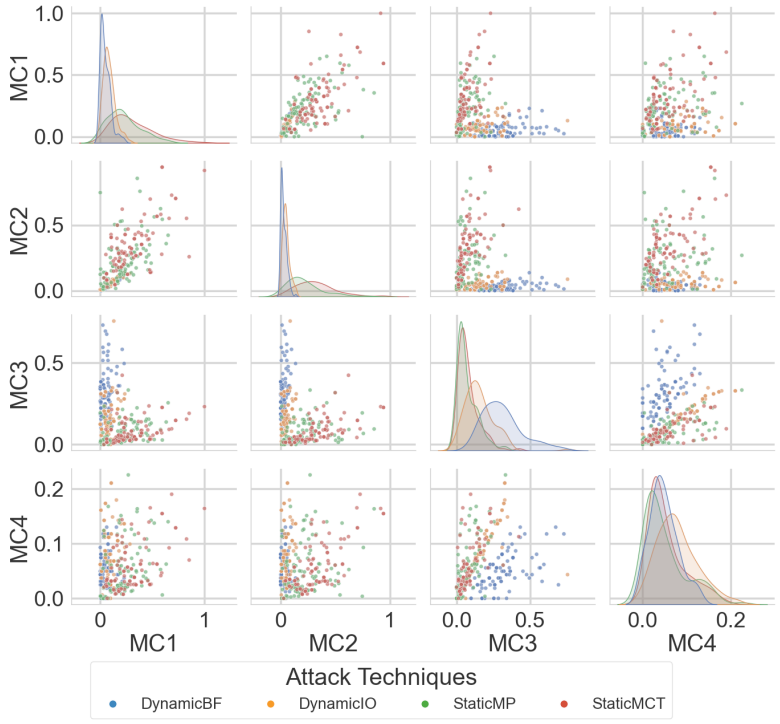
6.3.5 Attack Technique Classification

In addition to evaluating PLCPrint's ability to classify between static and dynamic attack scenarios, we examine whether the individual attack scenario techniques, presented earlier in table 6.1, can also be differentiated. Classifying the specific attack technique that is used to cause data manipulation on the PLC can assist the triaging process further by indicating a root-cause.

Using the GULP testbed, we execute an additional 200 attack scenarios using the same process and tools as discussed in Section 6.3.1 and conduct an equal number of scenarios for four of the attack techniques presented in table 6.1 on each PLC model, resulting in 50 scenarios for each attack technique. The *modify program* and *modify controller tasking* techniques always require the *program download* technique to be used in conjunction and so the program download technique was not evaluated independently. The PLC memory fingerprinting ap-



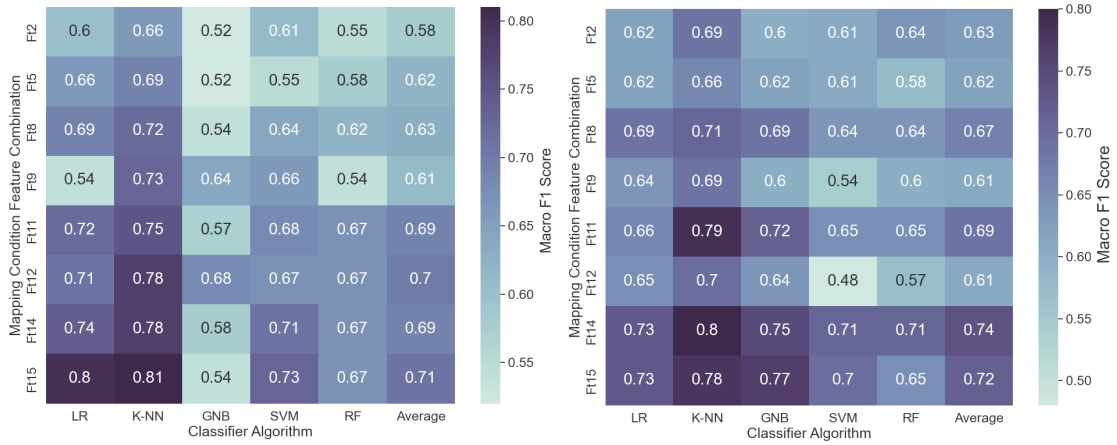
(a) AB CLX 1756 PLC



(b) Siemens S7-300 PLC

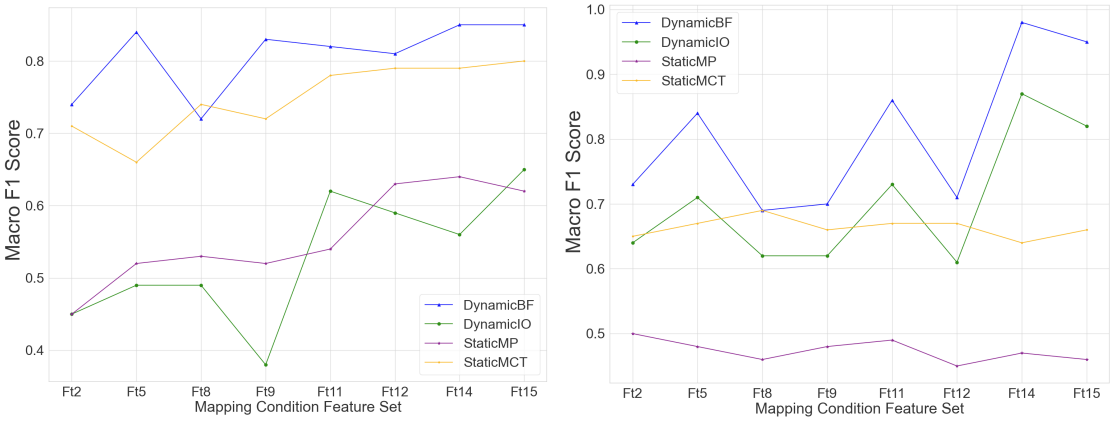
Figure 6.17: Distribution of MC features influenced by attack technique

proach is identical to previous experiments and the *finT* datasets generated from the attack scenarios are evaluated using the five classifier algorithms and the eight best performing MC feature sets from the attack classification evaluation in Section 6.3.4.



(a) AB CLX 1756 PLC (b) Siemens S7-300 PLC

Figure 6.18: F1 scores for classification of individual attack techniques for both PLCs



(a) AB CLX 1756 PLC (b) Siemens S7-300 PLC

Figure 6.19: Average F1 score performance of classifying attack techniques for both PLCs

The plots in figure 6.17 illustrate the distribution of the four different attack techniques when evaluated with pairs of the MC feature sets. The combination of MC2 and MC3 continues to be the most identifiable feature set at distinguishing between individual attack techniques. However, unlike the graphs in figure 6.8, there is additional noise when trying to identify the specific attack techniques since there are more similarities between two techniques from the same attack type than there is between the two attack types (dynamic and static) themselves.

As shown in figure 6.18, the resulting macro F1 scores for attack technique classification are significantly lower than for the attack type classification discussed previously in figure 6.14. Feature sets Ft11 and Ft14 achieve the highest scores of 0.79 and 0.8 for the S7-300 PLC, respectively, with the κ-NN classifier. Similarly, the AB-CLX PLC also achieves a highest macro F1 score with the κ-NN classifier of 0.81, however with the Ft15 MC feature set

combination. When we also consider the performance of κ-NN for attack type classification, illustrated by the ROC-AUC scores in figure 6.15, we can deduce that this particular classifier algorithm is generalisable between different PLC models and also for different stages of the PLC fingerprinting process. Moreover, attack technique classification generally achieves higher performance scores for all classifiers when more MC features are included in the feature set, indicating that the subtleties between MC deviations are more important for discriminating between attack techniques than for attack types. Attack type classification with Ft8 in particular performs very well compared with other feature sets with only two MC features (Ft5 - Ft10), as illustrated in figure 6.14, however this performance is not reflected when we conduct attack technique classification where Ft8 achieves lower macro F1 scores of 0.72 and 0.71 for AB-CLX and S7-300.

In addition, we investigate the classification performance for discriminating between individual attack techniques, as shown in figure 6.19. Generally, we observe that dynamic attack techniques are better classified in most cases for both PLC models. Attacks that used the *DynamicBF* technique perform best, reaching a macro F1 score of 0.98 and 0.85 with MC feature set Ft14 for the S7-300 and AB-CLX PLCs, respectively. The classification of static attack techniques, in particular *StaticMP* for the S7-300 generally yield lower measures. From further analysis of this, it is unclear why the classification of the StaticMP attack technique was much poorer than the other three techniques. One potential reason is that StaticMP has the lowest impact on the PLC register manipulation and focuses more on altering the structure of the PLC application code rather than the logic itself.

6.3.6 Network and Computational Impact

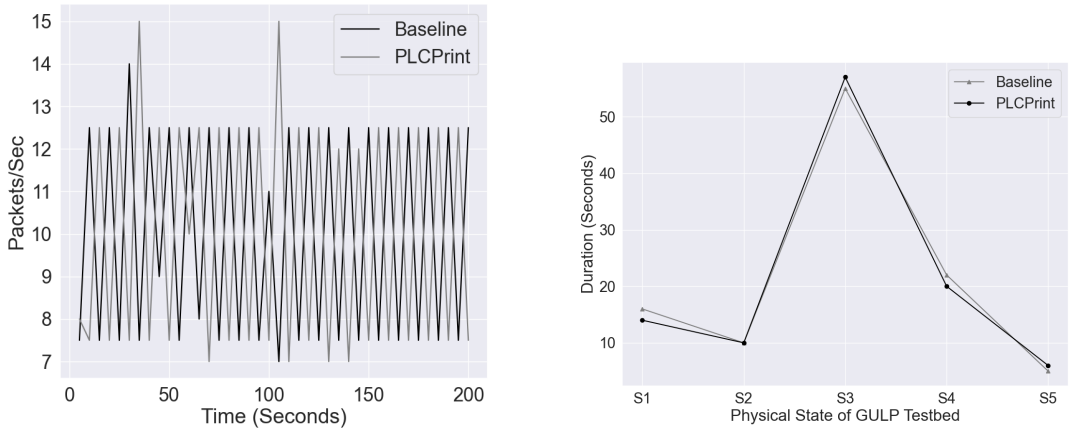
Due to the active approaches taken in acquiring memory artefacts from PLCs and the requirement to continuously generate test fingerprints in order to detect attacks, we regard evaluating the potential network and computational impacts introduced by PLCPrint as particularly valuable. In this section, we evaluate the network and computational impacts of PLCPrint operating in real-time with the GULP testbed. Understanding these impacts is critical to determining whether the introduced affects of the proposed solution do not inhibit the operability of the PLC or wider ICS and physical process. It should be noted that the aim of this evaluation was to examine the impacts caused by PLCPrint and not determine whether

PLCPrint is an efficient tool. Furthermore, while we discuss the importance of these impacts, we do not provide a comparison with existing approaches and therefore a good baseline to aim for. Primarily, this is due to comparable approaches not providing such evaluations.

We separated this evaluation into two areas of analysis; i) Network performance, and ii) PLC computational performance. For each analysis area, two use cases are considered regarding how PLCPrint could be deployed in real-world ICS environments; a) Standalone-PLC use where the PLC being fingerprinted is in a single-PLC network and only connected to an HMI, b) Networked-PLC where the PLC being fingerprinted is connected into a client-server architecture with one additional PLC. The GULP testbed presented in Section 3.3.1 is used for the evaluation of both use cases with the same experimental setup as the previous evaluation sections in this paper. For the network performance analysis, Wireshark was used to capture and analyse traffic between the PLC and HMI, and inter-PLC communications for the second use case. We used PLC performance features provided in the IDE of each PLC model. In our case, this was TIA Portal Version 16 for the S7-300 PLC, and Studio 5000 for the AB CLX PLC.

While PLCPrint introduces a significant amount of traffic into the network, we identified that it does not impact on the throughput of the network and no packet loss between the PLC and other devices on the ICS network was identified. The packet transfer rate per second with PLCPrint running was largely consistent with the baseline rate, shown in figure 6.20a. There was also no identified packet loss when multiple instances of PLCPrint were running at the same time for two different PLCs, although the amount of network traffic increased, as expected.

Moreover, we evaluated if PLCPrint had an impact on the computational performance of the PLC. Figure 6.20b illustrates the time it took for GULP to complete a physical state, such as filtration or disinfection as discussed earlier in Section 3.3.1. The duration for each physical state while PLCPrint is running is always within $\pm$ 2s, indicating that the active data acquisition approach of PLCPrint does not noticeably impact on the PLC's ability to monitor and control the physical process. Additionally, we monitored the PLC scan cycle time to determine if the data acquisition methods of PLCPrint introduced latency into the PLC's main cyclical operation. The baseline scan cycle time for both PLCs evaluated was 1ms ($\pm 500\mu$s). While PLCPrint was running, the S7-300 PLC scan cycle time increased



(a) Network performance impact of PLCPrint

(b) Performance of physical process under control

Figure 6.20: Performance Evaluation

at one instance to 2ms ($\pm 500\mu$s), however it then resumed at 1ms as in the baseline. As the PLC scan cycle time can fluctuate within a small range, it is likely that PLCPrint was responsible for this millisecond increase.

6.4 Summary

In this chapter, we have presented PLCPrint, a novel fingerprinting approach that enables real-time attack detection and classification for the wider anomaly diagnosis framework proposed in Chapter 3. Furthermore, we proposed PLC Memory Register Map (PMRM) generation, a method that examines how PLC registers are used by different memory artefacts. PMRM generation assigns one of four possible Mapping Conditions (MC) to each register used by the PLC, resulting in a PLC memory fingerprint comprising multiple PMRMs. An attack detection and classification algorithm was introduced that uses deviations in PLC register MCs as a feature set. Results from our experimental evaluation reveal high F1 scores for both attack detection and type classification with multiple PLC models, highlighting the vendor-independence of the PMRM generation approach. Furthermore, we have demonstrated that the OCSVM algorithms is particularly suitable for PLC attack detection in this chapter and also for more general anomaly detection in Chapter 5. Specifically, the computational and high detection performance of OCSVM makes it more suitable than other algorithms such as κ-NN, as well as being generalisable to multiple PLC vendors. High ac-

Table 6.9: Comparison of PLCPrint (Chapter 6) and existing fingerprinting approaches for PLC attack detection. **Key:** ●= Coverage, ◗= Limited Coverage, ❍= No Coverage. For Vendor Neutrality this denotes Multiple PLCs, Single PLC. For Testbed Evaluation this denotes Representative Physical Setup, Standalone Device, No Testbed Evaluation, respectively.

	Objective			**Method**		**Evaluation**		**Data Artefacts**			
Approach	**Forensic Applications**	**Attack Detection**	**Attack Provenance**	**Machine Learning**	**Data Synthesis**	**Vendor Neutrality**	**Testbed Evaluation**	**PLC Registers**	**PLC Application Code**	**Network Data**	**Side-channel Data**
Peng *et al* (2015) [115]	❍	●	❍	❍	❍	◗	◗	❍	❍	●	❍
Formby *et al* (2016) [73]	❍	●	❍	●	●	❍	◗	❍	❍	●	●
Yau & Chow (2017) [18]	◗	●	❍	●	❍	◗	❍	●	❍	❍	❍
Ahmed *et al* (2018) [109]	❍	●	❍	●	❍	◗	●	❍	❍	❍	●
Chan *et al* (2019) [67]	❍	●	❍	●	❍	◗	❍	●	❍	❍	❍
Stockman *et al* (2019) [63]	❍	●	❍	●	❍	◗	❍	❍	❍	❍	●
Ahmed *et al* (2020) [111]	❍	●	❍	●	❍	◗	●	❍	❍	❍	●
Yang *et al* (2020) [103]	❍	●	❍	●	❍	●	◗	●	❍	❍	❍
Yimer *et al* (2020) [127]	❍	●	❍	❍	❍	❍	❍	❍	❍	●	❍
Ahmed *et al* (2021) [113]	❍	●	❍	●	●	●	●	❍	❍	❍	●
PLCPrint Attack Fingerprinting	●	●	●	●	●	●	●	●	●	❍	❍

companying accuracy and AUC-ROC scores for the detection and classification algorithms were achieved, demonstrating the generalisation to multiple ML models.

The key novelty of PLCPrint is emphasized by the comparison in table 6.9 where we examine the key features employed by the existing literature. We separate features into four categories; i) Objective of study, ii) Method design, iii) Evaluation consideration and iv) Data artefacts used. Few approaches have integrated multiple data sources within their proposed fingerprinting approach [73], but most importantly no identified studies utilise a synthesised data model that synergises artefact dependencies. Moreover, PLCPrint is the first approach to combine both attack detection with classification, using ML, in order to assist with the provenance of cyber incidents. Regarding evaluation, PLCPrint is one of a few existing approaches that has been evaluated with real PLCs over a physical testbed emulation, increasing the validity of the results and demonstrating that PLCPrint can be generalised to multiple PLC vendors.

Work presented in [18] applies detection timestamps to a forensic use case, however this does not address challenges regarding attack provenance. Generally, PLCPrint assists with the digital forensic process by fingerprinting PLC memory artefacts and performing proactive artefact containment before an attack is identified, thus enabling the chain of evidence and rapid triaging of cyber incidents. PLCPrint can be particularly useful for operators in mission-critical environments where high availability is fundamental to the continuing operation of particular organisation and to the wider CNI. Due to the type of threat actors targeting ICS and the emphasis placed on restoring the system availability, it is likely that digital forensics will play multiple roles in ICS. While the need for conventional digital forensics that enable legal proceedings will still exist, we argue that a key objective of ICS operators will be on rapid triaging response, an area that PLCPrint can provide benefits for. By determining the type of attack in addition to containing potential forensic evidence, PLCPrint enables both a proactive and reactive forensic response.

Chapter 7

Conclusions and Future Work

7.1 Overview

The following Chapter provides the concluding points of this thesis. A summary of the contributions is first provided, referring back to the original research questions set out in Chapter 1. Following on from this, a number of future research directions are presented, primarily derived from the limitations of the research presented in the thesis and how the work could be extended further. Finally, the concluding remarks are provided.

7.2 Summary of Contributions

The overarching aim of the work presented in this thesis focused on the development and evaluation of a framework for anomaly diagnosis in the context of ICS, with a particular emphasis on PLCs. Drawing on the review of current literature provided in Chapter 2, the research proposes that anomaly diagnosis is a key interim stage, sitting critically between the detection of anomalies and the subsequent digital forensic analysis, which aims to classify anomalies and place them into context, based on their type, ultimately providing greater clarity for incident response.

The contributions of this thesis are summarised as follows:

- **PLC anomaly diagnosis framework**

  The work conducted in this thesis has contributed to the development of a central

framework for anomaly diagnosis in ICS, introduced in Section 3.2. Anomaly diagnosis is proposed as an intermediary stage between anomaly detection and digital forensics, that uses machine learning (ML) models combined with signatures in PLC data artefacts to provide a greater understanding of the context and provenance of identified anomalies. We propose three core stages to the anomaly diagnosis framework, which are developed and evaluated in throughout this thesis.

- **Taxonomy of PLC data artefacts**

  Focusing on the identification of feature sets to be used at each stage of the PLC anomaly diagnosis framework, a taxonomy of generalisable PLC data artefacts is proposed in Chapter 4. The taxonomy focuses on device data, which we define as data generated by the PLC itself and stored within its memory. Firstly, a set of four Artefact Data Types (ADTs) is identified that form high-level categories into which individual acquirable artefacts will fit. For each ADT, there is a set of underlying conditions that determines the categorisation of artefacts, including the dynamic or static behaviour of an artefact, and the functionality that it provides. Through evaluations of three different PLC models, we demonstrate that while the existence of the artefacts can be generalised across different vendors of PLC, the ability to acquire them is not universal. However, we demonstrated that the use of network commands integrated into the functionality of industrial protocols to perform data acquisition on PLCs, offers a highly promising approach. For example, an existing framework for Allen-Bradley PLC acquisition using the EtherNet/IP protocol was extended so that the application

Table 7.1: Usage of ADTs throughout PLC Anomaly Diagnosis Framework

	Data Artefact Type (ADT)			
Anomaly Diagnosis Framework Component	ADT 1	ADT 2	ADT 4	Network Data
Register State-Based Anomaly Detection (Chapter 5)	✓	✗	✗	✗
Anomaly Contextualisation (Chapter 5)	✗	✗	✓	✓
Memory Fingerprinting Attack Detection (Chapter 6)	✓	✓	✗	✗
Attack Provenance Classification (Chapter 6)	✓	✓	✗	✗

code from the PLC could be obtained. The proposed artefact taxonomy provides a framework for both ICS digital forensic investigations where PLCs are often a focal point, and for the development of future feature sets for ICS/PLC anomaly analysis, which is central to the subsequent stages of the PLC anomaly diagnosis framework, as detailed in table 7.1.

- **Real-time anomaly detection algorithms**

  The first stage in the anomaly diagnosis framework is performing anomaly detection based on changes in PLC operation behaviour. Two real-time anomaly detection algorithms are proposed in Chapter 5 and Chapter 6. The PLC register state-based approach uses signatures in PLC registers to generate states through subsets of active and inactive registers and model the operational behaviour of the PLC. A data pre-processing technique is proposed for feature set dimentionality reduction while maintaining the underlying feature information provided. As the number of registers that are used by a particular PLC positively correlates with the number of actuators and sensors within an ICS setup, we note that this technique would be particularly useful for real ICS environments where high numbers of industrial apparatus are often deployed to perform the physical process. The memory fingerprinting attack detection approach proposed in Chapter 6 uses a combination of memory artefacts in addition to the PLC register stage generation process used in the first, as defined in Section 5.2. Specifically, a composition of both static and dynamic data is used to formulate memory snapshots of the PLC. Both algorithms are evaluated using semi-supervised machine learning techniques, and the resulting high F1 and accuracy scores demonstrate the effectiveness of the proposed approaches compared with existing models described in previous literature. Moreover, vendor independence is also emphasised through the evaluation of two PLC vendors using the Glasgow University Liquid Purification (GULP) testbed.

- **Anomaly contextualisation model**

  Regarding the second stage of the anomaly diagnosis framework, this thesis has proposed a novel methodology for PLC anomaly contextualisation. A synthesised dataset is generated comprising dynamic PLC run-time data artefacts, specifically features pertaining to PLC registers, outbound network packets, and PLC device logs. A weighted deviation in network and device log metrics is calculated for a given anomalous data

point to infer a contextualisation of cyber-attacks or system fault. Through the use of supervised learning, a multi-class classification algorithm is implemented to discriminate between attacks and fault, where high F1 scores are produced. Furthermore, evaluation through Permutation and SHapley Additive exPlanations (SHAP), feature importance analysis reveals that network metrics, specifically PLC write acknowledgement packets, demonstrate likely cyber-attacks, whereas an increase in PLC device logs suggests a likely system fault. Moreover, the use of small dataset sizes further supports the strength of using PLC-specific artefacts, in particular ADT 1 and ADT 4, to infer anomaly contextualisation despite subtle deviations in feature metrics. Results from classifier performance evaluations using the ROC-AUC measure, indicate that the Random Forest (RF) classifier is particularly suitable, reaching average scores of 0.96. In fact, all three classifiers perform excellently, thus supporting the idea that the anomaly contextualisation approach can be generalised to different ML models. However, when the evaluation also considers computational performance, then Logistic Regression (LR) provides a more optimised approach, achieving similar ROC-AUC scores but demonstrating decreases of 46% and 68% in time consumption compared to RF, for training and testing activities respectively. In general, the presented contextualisation model establishes strong foundations for the development of vendor-independent intrusion detection components for PLCs and wider ICS.

- **Attack fingerprinting methodology (PLCPrint)**

  The final stage of the anomaly diagnosis framework, whose objective is to inform on attack provenance, introduces a fingerprinting approach to enable real-time attack classification for rapid triaging. We have proposed the PLC Memory Register Map (PMRM) generation, a novel method that examines how PLC registers are used by different memory artefacts. PMRM generation assigns one of four possible Mapping Conditions (MC) to each register used by the PLC, resulting in a PLC memory fingerprint comprising multiple PMRMs.

  An attack detection and classification algorithm was introduced that uses deviations in PLC register MCs as a feature set. As with the other contributions of this thesis, the PLCPrint methodology was evaluated on multiple PLC vendors deployed over the physical GULP testbed containing real industrial equipment. In comparison with

much of the literature that instead, evaluates a single PLC vendor or model, we demonstrate experimental evaluations that reveal high F1, accuracy and FNR scores for attack type and technique classification with multiple PLC vendors, highlighting the vendor-independence of the PMRM generation approach. Several supervised classification, and multi-class classification, algorithms have been evaluated, resulting in high accuracy and AUC-ROC scores and demonstrating the generalisation to multiple machine learning models. Generally, PLCPrint assists with the digital forensic process by performing proactive artefact containment before an attack is identified, and thus secures the chain of evidence. The efficient time consumption for both detection and attack provenance with PLCPrint further reinforces the suitability for rapidly triaiging cyber-attacks. Moreover, PLCPrint could be particularly useful for operators in mission-critical environments where high levels of availability are fundamental to the continuing operation of an organisation and to the wider CNI.

7.3 Example Use Cases

This section describes example use cases that demonstrate the high-level applications of the PLC anomaly diagnosis framework and its components, as have been presented in this thesis. Two proposed use cases will be explored: 1) the integration of anomaly diagnosis in advanced industrial network environments, and 2) Benefits for critical system domains.

7.3.1 Advanced Industrial Network Environments

One group of advanced network technologies that has become increasingly popular within current research is software-based network solutions such as Software-Defined Networks (SDNs) and Network Function Virtualization (NFV). Such technologies can improve the management of network communication flows, network visibility, and the deployment and control of network functions, using software instead of hardware-specific middleboxes [217]. In addition to their use in IT and enterprise data centers, SDNs and NFV also have applications in industrial environments, particularly with the rise in the Industrial Internet of Things, where more sophisticated distributed network architectures are employed, for example in renewable power generation and smart factory domains [218].

Within the context of this thesis, SDNs and NFV could enable the real-time response and execution of defensive techniques as a result of the output from the PLC anomaly diagnosis framework when an unknown anomaly is detected. If it is established that the scenario constitutes an attack, for example where PLC registers are being randomly manipulated through automated malware, functionality provided by an SDN controller can be used to migrate an adversary's control away from the real PLCs, and underlying physical process, to a simulated environment such as a honeypot environment [2]. Furthermore, if the PLC anomaly contextualisation algorithm, proposed in Chapter 5, determines that a system fault is the probable cause of anomalous behaviour, an SDN switch could transfer control from the affected equipment to redundancy devices through virtualisation, thus assisting with fault tolerance [2].

In figure 7.1, we propose an example architecture of how the PLC anomaly diagnosis framework could be utilised within an advanced industrial network environment facilitated by an SDN controller. At stage **A**, the anomaly diagnosis component is monitoring the PLC in real-time and identifies adverse behaviour through the anomaly contextualisation algorithm, which is subsequently determined to be a cyber attack. At **B1**, the SDN controller is informed of the attack and uses the output from PLCPrint, which performs attack fingerprinting to determine the attack type and technique, to restrict the traffic or modify packet flows to the PLC through the SDN switch at stage **C**. Anomalous process data from the PLC is logged in the historian at **B2**.

7.3.2 Critical System Domains

Many different sectors that use ICS technologies operate mission- and safety-critical systems. The optimised incident detection and analysis performance time demonstrated in the evaluations of Chapter 5 and Chapter 6 emphasise the suitability of the PLC anomaly diagnosis framework for critical system domains, where rapid triaging is important to resume safe and essential operations. Extended downtime can result in a loss of system availability through fail-safe protocols that prevent compromised components from performing their tasks. One type of fault tolerance control that is often found in critical environments, such as safety-instrumented systems, (SIS) is redundancy, where an alternative system or controller is implemented as a backup to be used in the event of an incident [3]. The aim is to provide additional reliability to the process control and reduce the downtime of the system. Fig-

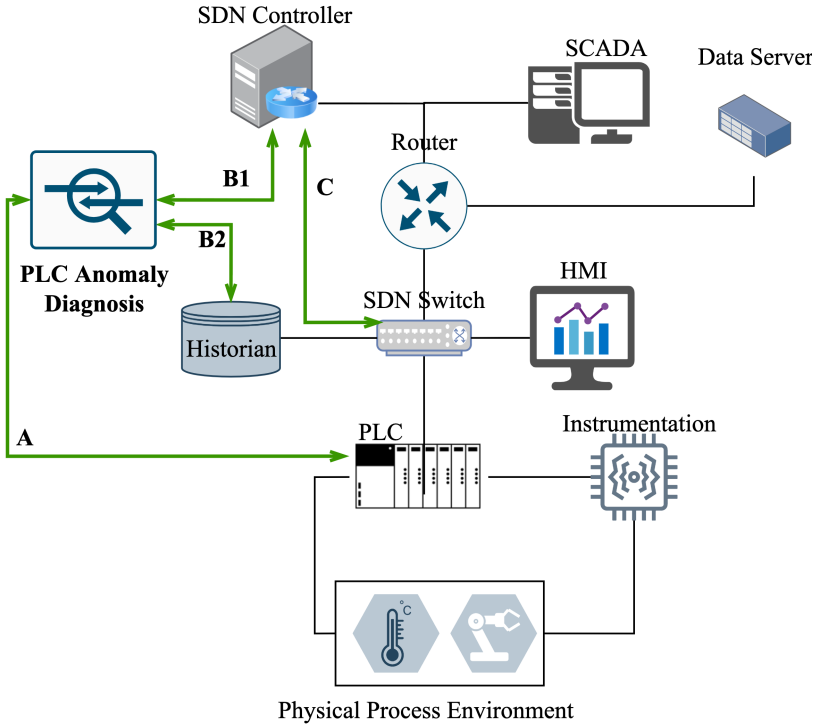


Figure 7.1: Enhanced ICS network architecture demonstrating use case of PLC anomaly diagnosis framework with SDN controller based on architecture proposed in [2]. (**A**, **B1**, **B2**, and **C** demonstrate the work flow for this use case and are discussed within the main text)

ure 7.2 demonstrates an example of controller redundancy comprising two PLCs, a primary (A) and secondary (B). When normal operations are observed, only one PLC will be actively responsible for controlling the sensors and actuators connected to the remote I/O units, while the other PLC serves as a standby controller.

An important use case for the PLC anomaly diagnosis framework that was presented earlier in the thesis, is assisting with the transfer of redundancy control during an incident, as depicted in figure 7.2. Additional incident report logs can be provided to the historian based on how the process data has been manipulated in the event of an attack, and whether the attack techniques used could effect both the primary and secondary controllers.

7.4 Limitations

Section 3.4.4, the research reported in the thesis proposes a set of attack scenarios that underpin the evaluations of the anomaly diagnosis framework, particularly the detection, contextualisation and attack fingerprinting components. However, the techniques involved have

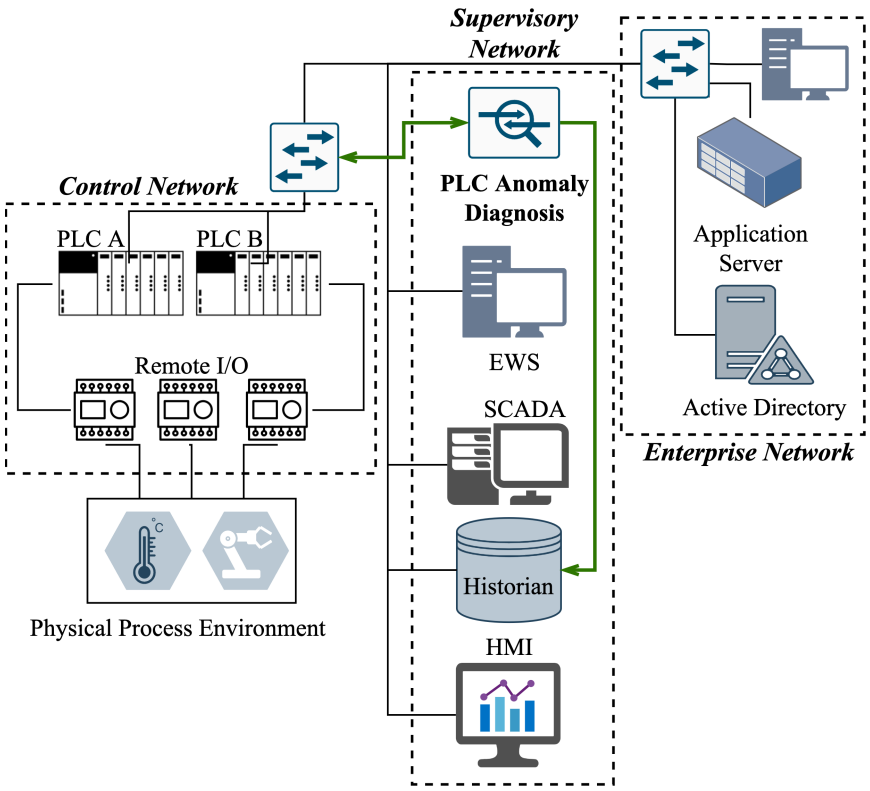


Figure 7.2: PLC redundancy architecture based on [3] demonstrating use case of PLC anomaly diagnosis framework for fault tolerant networks where *PLC A* is the primary controller and *PLC B* is the redundant secondary controller

been limited to those that manipulate PLC memory artefacts, primarily as these attacks have direct consequences on the underlying physical process. Therefore, it is unclear whether alternative attack techniques that perhaps manipulate different PLC data artefacts, for example network communications, would be identified and correctly classified by the anomaly diagnosis algorithms introduced in this thesis. Furthermore, a similar limitation exists with the selection of PLC fault scenarios that were used to evaluate the anomaly contextualisation model in Chapter 5. While these are generalised scenarios and extracted from commonly reported PLC fault cases, it is very likely that there are other conditions that have not been considered in this context.

As the research performs all evaluations with more than one PLC vendor, we can examine whether the anomaly diagnosis framework and its component methodologies are agnostic in deployment. Although vendor independence is demonstrated through real Siemens and Rockwell Allen-Bradley (AB) PLCs, the evaluations are limited to two PLC models. To increase confidence in our argument of vendor neutrality, additional PLC models and also

vendors would need to be explored. A limitation that is linked to this involves a particular engineering challenge that was encountered while conducting the work presented in this thesis. Specifically, it was particularly challenging to implement the methods for acquiring the application code from the PLCs. While the high-level syntax and acquisition method of the application code can be generalised, the acquisition software scripting required is PLC model-specific. Consequently, there were significant overheads in developing functions for acquiring application code, particularly from the AB ControlLogix PLC.

Furthermore, there are limitations regarding the application of the anomaly diagnosis framework to real-world ICS environments. The algorithms and models evaluated throughout this thesis are at a low technology readiness level (TRL) and have been demonstrated through proof-of-concept. Although we perform evaluations using industrial components in a representative ICS testbed presented in Section 3.3.1, the extent to which we can examine real system impacts is somewhat limited and requires more extensive work in the future. However, the safety, logistic and financial challenges facing the application of anomaly diagnosis in a real-life ICS setup confirm the critical role of testbed evaluations.

7.5 Future Research

There are several important directions, driven by the outcomes of the PLC anomaly diagnosis framework, that the next steps in future research could take. Some of these directions are based on the limitations of the work conducted in this thesis, as discussed in Section 7.4. The following sections outline some valuable areas of exploration that could be undertaken.

7.5.1 Multi-PLC Environments

The majority of evaluations conducted in this thesis have focused on single PLC environments, where one PLC has controlled the physical process within the GULP testbed. One challenge that could be addressed in future work is exploring how the PLC anomaly diagnosis framework could be scaled to operate in environments where multiple PLCs are used to control a single physical process. As the anomaly contextualisation and PLCPrint fingerprinting approaches both generate baselines for single target PLCs, environments where

multiple PLCs are used will require a ground-truth for every PLC. One possible solution that could assist with this is the use of containerisation to enable the rapid-deployment of new PLCs. Hence, this would allow the detection, contextualisation, and classification of anomalous behaviour to be controlled by the distributed PLC containers, rather than having a single centralised unit which would likely result in performance and network bottleneck challenges. As PLCs are typically resource-constraint devices, the containers would need to be housed on a separate component, such as a Raspberry Pi or lightweight Arduino controller.

7.5.2 PLC Data Artefact Provenance

Future research exploring how the anomaly diagnosis framework could be extended to increase its utility for digital forensic investigations would be very valuable. The work here could be to explore how the provenance of PLC data artefacts, such as those examined throughout this thesis, can be modelled to generate fingerprints of individual attack techniques in real-time. The use of graph theory is one potential method that could be used to demonstrate data provenance, where vectors would represent artefacts, and dependencies between these artefacts would be shown as edges. Contemporary ML techniques, such as Graph Neural Networks (GNNs) could support vector classification to enable root-cause analysis of attacks targeting the ICS. Moreover, this would provide greater digital forensic benefits as we would be able to establish a chain of evidence for specific devices, such as PLCs. Although ICS are often fairly deterministic, they are highly dynamic, producing large quantities of data which is transferred from field instrumentation (sensors and actuators) to the PLCs that control them. Therefore, using Dynamic graphs where vertices and edges can evolve over time, could be a particularly effective method in the context of ICS. Indeed, the underlying concepts proposed in this section of future work form the basis of a follow-on project currently being undertaken by myself through a Research Associate position at the University of Glasgow, recently funded by the U.K. Ministry of Defence (MoD).

7.5.3 Additional Anomaly Scenarios

A number of anomaly scenarios were developed in this thesis to evaluate the anomaly diagnosis framework, in particular cyber-attacks targeting PLCs. However, the coverage of

potential attack vectors was certainly not complete, and therefore other scenarios should be developed and evaluated, especially network-based attacks such as denial of service and man in the middle threats. Real ICS malware case studies are often modular, where multiple different attack techniques could take place in one scenario, for example injecting code into PLC memory and then blocking the communications link back to the engineering workstation to prevent the user from restoring a baseline configuration. Such formulations have recently been identified in the latest PLC malware, such as the Pipedream framework that was demonstrated to provide data manipulation, reconnaissance, and denial of control attack vectors [219]. Generating datasets that comprise multiple attack types and techniques could be addressed as a multi-label classification problem instead of multi-class. Future work could explore how data signatures can be extracted from combinations of different attack techniques, to then be used in future detection capabilities.

7.5.4 Regression for Anomaly Diagnosis

While the research reported here has made heavy use of ML classification models to perform anomaly contextualisation and attack fingerprinting, one area that could be explored in future work is the use of regression methods to enable real-time attack prediction. Regression models have primarily been used in forecasting applications, such as performing financial predictions, however their ability to explore subtle relationships between variables, for example network communication metrics, could be exploited for predictive detection. In particular, the temporal and physical principles of control systems could be used to continuously model normal operating behaviour using measurable quantities from PLC data.

7.5.5 Adapting for Online Learning

In Chapter 5, we briefly discussed how online machine learning could be used to improve the accuracy of evolving ICS environments by continuously streaming new data to improve the baseline model. Although the processes in ICS are generally much more deterministic than their IT counterparts, such as corporate networks, ICS are still susceptible to changes through configuration and program updates of components, such as PLCs, to improve efficiency and productivity. To the best of our knowledge, the examination of online machine learning in

the context of ICS is currently very limited, quite possibly due to online techniques requiring data streaming, a particular challenge in ICS that are typically resource-constrained environments. Future anomaly detection research could examine how the computational performance of ICS components is impacted by offline and online learning, and more importantly, what the potential trade-offs are regarding detection accuracy and speed. In general, next-generational technologies including the Industrial Internet of Things (IIoT) and Industry 4.0 that utilise advanced network technologies could address some of these concerns.

7.5.6 Adversarial Machine Learning Protection

While ML can bring many benefits to ICS cyber-security and digital forensics, particularly around anomaly detection and attack classification, ML also introduces additional attack vectors that can be exploited by attackers [220]. Adversarial ML introduces a novel form of denial of service by manifesting malicious perturbations referred to as adversarial examples. These are designed to degrade the detection performance of a particular ML system and enable an attacker to circumvent security controls by suppressing early warning alarms [221]. As ML is heavily used within the components of the anomaly diagnosis framework, future research must explore how attacks can manipulate PLC memory baselines to reduce the effectiveness of attack detection and classification, and how to best to mitigate this emerging threat. For instance, the novelty detection algorithms presented in Chapter 5 and Chapter 6 rely heavily on adequate baseline training datasets generated from the normal and highly deterministic operating behaviour of the ICS. An attacker could implement a poising attack to manipulate the training datasets by introducing intentional biases, such as incorrect time durations for particular stages within the water treatment process.

7.6 Final Remarks

The increase in inter-connectivity across ICS networks and components, and particularly that to Internet-facing networks, has brought new security challenges to environments that were traditionally physically segregated. Consequently, there has been a significant growth in number of cyber attacks, and indeed, of the complexity of the malware targeting ICS, particularly PLCs, due to their central role in controlling and monitoring kinetic processes

within the physical system. A significant amount of the existing literature has explored how technologies and approaches, such as ML, can be used to detect adverse behaviour, potentially caused by a cyber attack. However, this body of work is limited to the identification of anomalies, and does not address attack differentiation, contexts and causes. The area of anomaly diagnosis looks to address this problem.

To this end, the research reported in this thesis has proposed, implemented, and extensively evaluated an anomaly diagnosis framework for use with PLCs functioning within the context of ICS. We have argued that anomaly diagnosis plays a critical role between the detection of anomalous behaviour and the subsequent digital forensic investigation that ensues if a real-time, automated analysis of the causes of an anomaly can be provided. Furthermore, this research has demonstrated that contextualising PLC behaviour can be achieved through synthesised datasets of both device data and network communications. Furthermore, compositions of PLC memory artefacts have been successfully used in an attack fingerprinting methodology that identifies how PLCs have been compromised, and what specific attack techniques were employed to in that compromise. Finally, the anomaly diagnosis framework is the first of its kind, offering a structured approach to the analysis of ICS cyber attacks. However, while the framework shows much promise, future research will surely expand, strengthen and optimise its contribution to this very important, and ever-evolving, field.

Bibliography

[1] P. A. Jaskowiak, I. G. Costa, and R. J. Campello, "The area under the roc curve as a measure of clustering quality," *Data Mining and Knowledge Discovery*, vol. 36, no. 3, pp. 1219–1245, 2022.

[2] A. F. Murillo Piedrahita, V. Gaur, J. Giraldo, A. Cárdenas, and S. J. Rueda, "Leveraging software-defined networking for incident response in industrial control systems," *IEEE Software*, vol. 35, no. 1, pp. 44–50, 2018.

[3] K. Liu, J.-Y. Wang, Q. Wei, Z.-Y. Zhang, J. Sun, R.-K. Ma, and R.-L. Deng, "Hrpdf: A software-based heterogeneous redundant proactive defense framework for programmable logic controller," *Journal of Computer Science and Technology*, vol. 36, no. 6, pp. 1307–1324, 2021.

[4] M. Cook, I. Stavrou, S. Dimmock, and C. Johnson, "Introducing a forensics data type taxonomy of acquirable artefacts from programmable logic controllers," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2020, pp. 1–8.

[5] M. Cook, C. Paterson, A. K. Marnerides, and D. Pezaros, "Anomaly diagnosis in cyber-physical systems," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 5445–5450.

[6] S. Kondo, H. Sakashita, S. Sato, T. Hamaguchi, and Y. Hashimoto, "An application of stamp to safety and cyber security for ics," in *13th International Symposium on Process Systems Engineering (PSE 2018)*, ser. Computer Aided Chemical Engineering, M. R. Eden, M. G. Ierapetritou, and G. P. Towler,

Eds. Elsevier, 2018, vol. 44, pp. 2335–2340. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444642417503840

[7] P. Arora, B. Kaur, and M. A. Teixeira, "Security in industrial control systems using machine learning algorithms: An overview," in *ICT Analysis and Applications*, S. Fong, N. Dey, and A. Joshi, Eds. Singapore: Springer Nature Singapore, 2022, pp. 359–368.

[8] Y. Mekdad, G. Bernieri, M. Conti, and A. El Fergougui, "The rise of ics malware: A comparative analysis," in *Computer Security. ESORICS 2021 International Workshops*, S. Katsikas, C. Lambrinoudakis, N. Cuppens, J. Mylopoulos, C. Kalloniatis, W. Meng, S. Furnell, F. Pallas, J. Pohle, M. A. Sasse, H. Abie, S. Ranise, L. Verderame, E. Cambiaso, J. Maestre Vidal, and M. A. Sotelo Monge, Eds. Cham: Springer International Publishing, 2022, pp. 496–511.

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

[10] J. E. Sullivan and D. Kamensky, "How cyber-attacks in ukraine show the vulnerability of the u.s. power grid," *The Electricity Journal*, vol. 30, no. 3, pp. 30–35, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1040619017300507

[11] A. Hassanzadeh, A. Rasekh, S. Galelli, M. Aghashahi, R. Taormina, A. Ostfeld, and M. K. Banks, "A review of cybersecurity incidents in the water sector," *Journal of Environmental Engineering*, vol. 146, no. 5, p. 03120003, 2020. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29EE.1943-7870.0001686

[12] J.-w. Myung and S. Hong, "Ics malware triton attack and countermeasures." *INTERNATIONAL JOURNAL OF EMERGING MULTIDISCIPLINARY RESEARCH (IJEMR)*, vol. 3, no. 2, pp. 13–17, 2019.

[13] T. Miller, A. Staves, S. Maesschalck, M. Sturdee, and B. Green, "Looking back to look forward: Lessons learnt from cyber-attacks on industrial control systems," *International Journal of Critical Infrastructure Protection*, vol. 35, p.

100464, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548221000524

[14] A. Adamov, A. Carlsson, and T. Surmacz, "An analysis of lockergoga ransomware," in *2019 IEEE East-West Design Test Symposium (EWDTS)*, 2019, pp. 1–5.

[15] M. Sparkes, "How do we solve the problem of ransomware?" *New Scientist*, vol. 250, no. 3336, p. 13, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S026240792100899X

[16] A. K. Marnerides, V. Giotsas, and T. Mursch, "Identifying infected energy systems in the wild," in *Proceedings of the Tenth ACM International Conference on Future Energy Systems*, ser. e-Energy '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 263–267. [Online]. Available: https://doi.org/10.1145/3307772.3328305

[17] D. Formby, M. Rad, and R. Beyah, "Lowering the barriers to industrial control system security with GRFICS," in *2018 USENIX Workshop on Advances in Security Education (ASE 18)*. Baltimore, MD: USENIX Association, Aug. 2018. [Online]. Available: https://www.usenix.org/conference/ase18/presentation/formby

[18] K. Yau, K. P. Chow, S. M. Yiu, and C. F. Chan, "Detecting anomalous behavior of plc using semi-supervised machine learning," in *2017 IEEE Conference on Communications and Network Security (CNS)*, 2017.

[19] R. A. Awad, S. Beztchi, J. M. Smith, B. Lyles, and S. Prowell, "Tools, techniques, and methodologies: A survey of digital forensics for scada systems," ser. Industrial Control System Security Workshop (ICSS) 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–8. [Online]. Available: https://doi.org/10.1145/3295453.3295454

[20] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[21] I. Friedberg, X. Hong, K. McLaughlin, P. Smith, and P. C. Miller, "Evidential network modeling for cyber-physical system state inference," *IEEE Access*, vol. 5, pp. 17 149–17 164, 2017.

[22] C. Hwang and T. Lee, "E-sfd: Explainable sensor fault detection in the ics anomaly detection system," *IEEE Access*, vol. 9, pp. 140 470–140 486, 2021.

[23] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 817–831. [Online]. Available: https://doi.org/10.1145/3243734.3243781

[24] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *ICT Systems Security and Privacy Protection*, G. Dhillon, F. Karlsson, K. Hedström, and A. Zúquete, Eds. Cham: Springer International Publishing, 2019, pp. 33–48.

[25] "A road map for digital forensic research: Report from the first digital forensic research workshop (DFRWS)," Digital Forensic Research Workshop (DFRWS), New York, USA, Technical Report, 2001.

[26] C.-F. Chan, K.-P. Chow, S.-M. Yiu, and K. Yau, "Enhancing the security and forensic capabilities of programmable logic controllers," in *Advances in Digital Forensics XIV*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2018, pp. 351–367.

[27] I. Ahmed, S. Obermeier, M. Naedele, and G. G. Richard III, "Scada systems: Challenges for forensic investigators," *Computer*, vol. 45, no. 12, pp. 44–51, 2012.

[28] European Commission, "Shaping europe's digital future - european commission," 2020. [Online]. Available: https://ec.europa.eu/digital-single-market/en/network-and-information-security-nis-directive

[29] C. Chance, "Cyber-security: What regulators are saying around the world," Tech. Rep., 2020. [Online]. Avail-

able: https://www.cliffordchance.com/content/dam/cliffordchance/briefings/2018/06/cyber-security-what-regulators-are-saying-around-the-world.pdf

[30] T. Pavleska, H. Aranha, M. Masi, and G. P. Sellitto, "Drafting a cybersecurity framework profile for smart grids in eu: a goal-based methodology," in *European Dependable Computing Conference*. Springer, 2020, pp. 143–155.

[31] C. M. Ahmed and A. P. Mathur, "Hardware identification via sensor fingerprinting in a cyber physical system," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2017, pp. 517–524.

[32] I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev, "Programmable logic controller forensics," *IEEE Security Privacy*, vol. 15, no. 06, pp. 18–24, 2017.

[33] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, my malware knows physics! attacking plcs with physical model aware rootkit." in *Network and Distributed System Security Symposium NDSS*, 2017.

[34] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Cyber–physical systems forensics: Today and tomorrow," *Journal of Sensor and Actuator Networks*, vol. 9, no. 3, 2020. [Online]. Available: https://www.mdpi.com/2224-2708/9/3/37

[35] T. J. Williams, "The purdue enterprise reference architecture," *Computers in industry*, vol. 24, no. 2-3, pp. 141–158, 1994.

[36] A. Zanutto, B. O. Shreeve, K. Follis, J. S. Busby, and A. Rashid, "The shadow warriors: In the no man's land between industrial control systems and enterprise it systems," 2017.

[37] "Programmable Controllers," Standard ISO/IEC 61131, 2013.

[38] "Programmable Controllers Part 3: Programming Languages," Standard ISO/IEC 61131-3:2013, 2013.

[39] "Programmable Controllers Part 6: Functional Safety," Standard IEC 61131-6:2012, 2012.

[40] D. Sullivan, E. Luiijf, and E. J. M. Colbert, *Components of Industrial Control Systems*. Cham: Springer International Publishing, 2016, pp. 15–28. [Online]. Available: https://doi.org/10.1007/978-3-319-32125-7_2

[41] D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A survey on scada systems: Secure protocols, incidents, threats and tactics," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1942–1976, 2020.

[42] N. I. for Standards and Technology, "Framework for improving critical infrastructure cybersecurity version 1.1," 2018-04-16 2018.

[43] L. A. Gordon, M. P. Loeb, and L. Zhou, "Integrating cost–benefit analysis into the nist cybersecurity framework via the gordon–loeb model," *Journal of Cybersecurity*, vol. 6, no. 1, p. tyaa005, 2020.

[44] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, jul 2009. [Online]. Available: https://doi.org/10.1145/1541880.1541882

[45] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[46] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[47] "State of endpoint security risk," Ponemon Institute, Tech. Rep., January 2020.

[48] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao, "Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[49] R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

[50] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[51] C. M. Ahmed, G. R. MR, and A. P. Mathur, "Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems," in *Proceedings of the 6th ACM on cyber-physical system security workshop*, 2020, pp. 23–29.

[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[53] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, 2021.

[54] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[55] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[56] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: a big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[57] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[58] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," vol. 12, 01 1999.

[59] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, p. 273–297, Sep 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[60] M. Elnour, N. Meskin, K. Khan, and R. Jain, "A dual-isolation-forests-based attack detection framework for industrial control systems," *IEEE Access*, vol. 8, pp. 36 639–36 651, 2020.

[61] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *2008 8th IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[62] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, 2001.

[63] M. Stockman, D. Dwivedi, R. Gentz, and S. Peisert, "Detecting control system misbehavior by fingerprinting programmable logic controller functionality," *International Journal of Critical Infrastructure Protection*, vol. 26, p. 100306, 2019.

[64] Y.-Y. Song and L. Ying, "Decision tree methods: applications for classification and prediction," *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.

[65] K. Yau and K.-P. Chow, "Detecting anomalous programmable logic controller events using machine learning," in *IFIP International Conference on Digital Forensics*. Springer, 2017, pp. 81–94.

[66] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, "Tabor: A graphical model-based approach for anomaly detection in industrial control systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 525–536. [Online]. Available: https://doi.org/10.1145/3196494.3196546

[67] C.-F. Chan, K.-P. Chow, C. Mak, and R. Chan, "Detecting anomalies in programmable logic controllers using unsupervised machine learning," in *Advances in Digital Forensics XV*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2019, pp. 119–130.

[68] Y.-j. Xiao, W.-y. Xu, Z.-h. Jia, Z.-r. Ma, and D.-l. Qi, "Nipad: a non-invasive power-based anomaly detection scheme for programmable logic controllers," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 4, pp. 519–534, 2017.

[69] J. Inoue, Y. Yamagata, Y. Chen, C. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *IEEE ICDMW 2017*, 2017.

[70] M. Abdelaty, R. Doriguzzi-Corin, and D. Siracusa, "Daics: A deep learning solution for anomaly detection in industrial control systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 10, no. 2, pp. 1117–1129, 2021.

[71] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[72] C. Liu, S. Ghosal, Z. Jiang, and S. Sarkar, "An unsupervised spatiotemporal graphical modeling approach to anomaly detection in distributed cps," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–10.

[73] D. Formby, P. Srinivasan, A. M. Leonard, J. D. Rogers, and R. A. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems." in *NDSS*, 2016.

[74] C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and lstm networks," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017, pp. 261–272.

[75] D. Formby and R. Beyah, "Temporal execution behavior for host anomaly detection in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1455–1469, 2020.

[76] A. Ghosh, S. Qin, J. Lee, and G.-N. Wang, "Plat: an automated fault and behavioural anomaly detection tool for plc controlled manufacturing systems," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[77] A. Ghosh, S. Qin, J. Lee, and G. Wang, "Fbmtp: An automated fault and behavioral anomaly detection and isolation tool for plc-controlled manufacturing systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 12, pp. 3397–3417, 2017.

[78] S. Krishnamurthy, S. Sarkar, and A. Tewari, "Scalable anomaly detection and isolation in cyber-physical systems using bayesian networks," in *Dynamic Systems and Control Conference*, vol. 46193. American Society of Mechanical Engineers, 2014, p. V002T26A006.

[79] M. Conti, D. Donadel, and F. Turrin, "A survey on industrial control system testbeds and datasets for security research," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2248–2294, 2021.

[80] Y. Geng, Y. Wang, W. Liu, Q. Wei, K. Liu, and H. Wu, "A survey of industrial control system testbeds," *IOP Conference Series: Materials Science and Engineering*, vol. 569, 2019.

[81] H. Holm, M. Karresand, A. Vidström, and E. Westring, "A survey of industrial control system testbeds," in *Secure IT Systems*, S. Buchegger and M. Dam, Eds. Springer International Publishing, 2015, pp. 11–26.

[82] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[83] V. S. Harichandran, D. Walnycky, I. Baggili, and F. Breitinger, "Cufa: A more formal definition for digital forensic artifacts," *Digital Investigation*, vol. 18, pp. S125 – S137, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287616300366

[84] D. Hanes, G. Salgueiro, P. Grossetete, R. Barton, and J. Henry, *IoT fundamentals: Networking technologies, protocols, and use cases for the Internet of Things*. Cisco Press, 2017.

[85] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 76 – 84, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1874548213000231

[86] G. Denton, F. Karpisek, F. Breitinger, and I. Baggili, "Leveraging the SRTP protocol for over-the-network memory acquisition of a GE Fanuc series 90-30," *Digital Investigation*, vol. 22, pp. 26 – 38, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1742287617301925

[87] R. M. van der Knijff, "Control systems/scada forensics, what's the difference?" *Digital Investigation*, vol. 11, no. 3, pp. 160–174, 2014.

[88] P. Eden, A. Blyth, P. Burnap, Y. Cherdantseva, K. Jones, H. Soulsby, and K. Stoddart, "Forensic readiness for SCADA/ICS incident response," in *Proceedings of the 4th International Symposium for ICS and SCADA Cyber Security Research*, ser. ICS-CSR '16, 2016, pp. 142–150. [Online]. Available: https://doi.org/10.14236/ewic/ICS2016.16

[89] R. Altschaffel, M. Hildebrandt, S. Kiltz, and J. Dittmann, "Digital forensics in industrial control systems," in *Computer Safety, Reliability, and Security*, A. Romanovsky, E. Troubitsyna, and F. Bitsch, Eds. Cham: Springer International Publishing, 2019, pp. 128–136.

[90] C. M. Rondeau, M. A. Temple, and J. Lopez, "Industrial IoT cross-layer forensic investigation," *Wiley Interdisciplinary Reviews: Forensic Science*, vol. 1, no. 1, p. e1322, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/wfs2.1322

[91] T. Wu, J. F. P. Disso, K. Jones, and A. Campos, "Towards a SCADA forensics architecture," in *Proceedings of the 1st International Symposium on ICS amp; SCADA Cyber Security Research 2013*, ser. ICS-CSR 2013. Swindon, GBR: BCS, 2013, p. 12–21.

[92] T. Spyridopoulos, T. Tryfonas, and J. May, "Incident analysis digital forensics in scada and industrial control systems," in *8th IET International System Safety Conference incorporating the Cyber Security Conference 2013*, 2013, pp. 1–6.

[93] J. Stirland, K. Jones, H. Janicke, T. Wu *et al.*, "Developing cyber forensics for scada industrial control systems," in *The International Conference on Information Security and Cyber Forensics (InfoSec2014). The Society of Digital Information and Wireless Communication*, 2014, pp. 98–111.

[94] P. Van Vliet, M.-T. Kechadi, and N.-A. Le-Khac, "Forensics in industrial control system: A case study," in *Security of Industrial Control Systems and Cyber Physical Systems*, A. Bécue, N. Cuppens-Boulahia, F. Cuppens, S. Katsikas, and C. Lambrinoudakis, Eds. Cham: Springer International Publishing, 2016, pp. 147–156.

[95] E. Peter, B. Andrew, P. Burnap, Y. Cherdantseva, K. Jones, and H. Soulsby, "A forensic taxonomy of scada systems and approach to incident response," in *Proceedings of 3rd International Symposium for ICS SCADA Cyber Security Research 2015 (ICS-CSR 2015) (ICS-CSR)*, 2015. [Online]. Available: http://dx.doi.org/10.14236/ewic/ICS2015.5

[96] E. Sohl, C. Fielding, T. Hanlon, J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell, "A field study of digital forensics of intrusions in the electrical power grid," in *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*, ser. CPS-SPC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 113–122. [Online]. Available: https://doi.org/10.1145/2808705.2808716

[97] E. M. Iqbal A, Mahmood F, "Digital forensic analysis of industrial control systems using sandboxing: A case of WAMPAC applications in the power systems," *Energies*, vol. 12, no. 13, 2019.

[98] R. Chan and K.-P. Chow, "Forensic analysis of a siemens programmable logic controller," in *Critical Infrastructure Protection X*, M. Rice and S. Shenoi, Eds. Cham: Springer International Publishing, 2016, pp. 117–130.

[99] T. Wu and J. R. C. Nurse, "Exploring the use of PLC debugging tools for digital forensic investigations on SCADA systems," *Journal of Digital Forensics, Security and Law*, vol. 10, no. 4, pp. 79–96, 2015. [Online]. Available: https://kar.kent.ac.uk/67499/

[100] K. Yau and K.-P. Chow, "PLC forensics based on control program logic change detection," in *Journal of Digital Forensics, Security and Law*, vol. 10, no. 4, 2015.

[101] S. Senthivel, I. Ahmed, and V. Roussev, "SCADA network forensics of the PCCC protocol," *Digital Investigation*, vol. 22, no. S, p. S57–S65, 2017. [Online]. Available: https://doi.org/10.1016/j.diin.2017.06.012

[102] K. Yau, K.-P. Chow, and S.-M. Yiu, "A forensic logging system for siemens programmable logic controllers," in *Advances in Digital Forensics XIV*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2018, pp. 331–349.

[103] K. Yang, Q. Li, X. Lin, X. Chen, and L. Sun, "ifinger: Intrusion detection in industrial control systems via register-based fingerprinting," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, 2020.

[104] A. W. Amit Kleinmann, "Accurate modelling of the Siemens S7 SCADA protocol for intrusion detection and digital forensics," *The Journal of Digital Forensics, Security and Law (JDFSL)*, vol. 9, no. 4, 2014. [Online]. Available: https://doi.org/10.15394/jdfsl.2014.1169

[105] P. N. Taveras, "Scada live forensics: Real time acquisition process to detect, prevent or evaluate critical situations," *European Scientific Journal, ESJ*, vol. 9, no. 21, 2013. [Online]. Available: http://eujournal.org/index.php/esj/article/view/1457

[106] J. Parry, D. Hunter, K. Radke, and C. Fidge, "A network forensics tool for precise data packet capture and replay in cyber-physical systems," in *Proceedings of the Australasian Computer Science Week Multiconference*, ser. ACSW '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2843043.2843047

[107] K. Yau, K.-P. Chow, and S.-M. Yiu, "Detecting anomalous programmable logic controller events using process mining," in *International Conference on Critical Infrastructure Protection*. Springer, 2021, pp. 119–133.

[108] S. Velliangiri, S. Alagumuthukrishnan *et al.*, "A review of dimensionality reduction techniques for efficient computation," *Procedia Computer Science*, vol. 165, pp. 104–111, 2019.

[109] C. M. Ahmed, J. Zhou, and A. P. Mathur, "Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in cps," in *Proceedings of the 34th Annual Computer Security Applications Conference*, 2018, pp. 566–581.

[110] C. M. Ahmed, M. Ochoa, J. Zhou, A. P. Mathur, R. Qadeer, C. Murguia, and J. Ruths, "Noiseprint: Attack detection using sensor and process noise fingerprint in cyber physical systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 483–497.

[111] C. M. Ahmed, J. Prakash, R. Qadeer, A. Agrawal, and J. Zhou, "Process skew: fingerprinting the process for anomaly detection in industrial control systems," in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 219–230.

[112] G. F. Welch, "Kalman filter," *Computer Vision: A Reference Guide*, pp. 1–3, 2020.

[113] C. M. Ahmed, M. Ochoa, J. Zhou, and A. Mathur, "Scanning the cycle: timing-based authentication on plcs," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 886–900.

[114] M. Caselli, D. Hadžiosmanović, E. Zambon, and F. Kargl, "On the feasibility of device fingerprinting in industrial control systems," in *Critical Information Infrastructures Security*, E. Luiijf and P. Hartel, Eds. Cham: Springer International Publishing, 2013, pp. 155–166.

[115] Y. Peng, C. Xiang, H. Gao, D. Chen, and W. Ren, "Industrial control system fingerprinting and anomaly detection," in *International Conference on Critical Infrastructure Protection*. Springer, 2015, pp. 73–85.

[116] C. Shen, C. Liu, H. Tan, Z. Wang, D. Xu, and X. Su, "Hybrid-augmented device fingerprinting for intrusion detection in industrial control system networks," *IEEE Wireless Communications*, vol. 25, no. 6, pp. 26–31, 2018.

[117] A. T. Al Ghazo and R. Kumar, "Ics/scada device recognition: A hybrid communication-patterns and passive-fingerprinting approach," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2019, pp. 19–24.

[118] G. Walkup, S. Etigowni, D. Xu, V. Urias, and H. W. Lin, "Forensic investigation of industrial control systems using deterministic replay," in *2020 IEEE Conference on Communications and Network Security (CNS)*, 2020, pp. 1–9.

[119] H. Dong and D. Peng, "Research on abnormal detection of modbustcp/ip protocol based on one-class svm," in *2018 33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2018, pp. 398–403.

[120] A. Iqbal, M. Ekstedt, and H. Alobaidli, "Digital forensic readiness in critical infrastructures: A case of substation automation in the power sector," in *Digital Forensics and Cyber Crime*, P. Matoušek and M. Schmiedecker, Eds. Cham: Springer International Publishing, 2018, pp. 117–129.

[121] H. R. Ghaeini and N. O. 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*, 2016, pp. 103–111.

[122] D. Myers, S. Suriadi, K. Radke, and E. Foo, "Anomaly detection for industrial control systems using process mining," *Computers & Security*, vol. 78, pp. 103–125, 2018.

[123] H. R. Ghaeini, D. Antonioli, F. Brasser, A.-R. Sadeghi, and N. O. Tippenhauer, "State-aware anomaly detection for industrial control systems," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1620–1628.

[124] G. Settanni, F. Skopik, A. Karaj, M. Wurzenberger, and R. Fiedler, "Protecting cyber physical production systems using anomaly detection to enable self-adaptation," in *2018 IEEE ICPS*, 2018.

[125] M. Hussain, E. Foo, and S. Suriadi, "An improved industrial control system device logs processing method for process-based anomaly detection," in *2019 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2019, pp. 150–1505.

[126] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[127] T. Yimer, M. T. Arafin, and K. Kornegay, "Securing industrial control systems using physical device fingerprinting," in *IOTSMS*, 2020, pp. 1–6.

[128] (2022) Forescout. [Online]. Available: https://www.forescout.com

[129] (2022) Nozomi networks. [Online]. Available: https://www.nozominetworks.com

[130] (2022) Microsoft defender for iot. [Online]. Available: https://azure.microsoft.com/en-us/services/iot-defender/#overview

[131] P. Eden, A. Blyth, P. Burnap, Y. Cherdantseva, K. Jones, H. Soulsby, and K. Stoddart, "A cyber forensic taxonomy for scada systems in critical infrastructure," in *International Conference on Critical Information Infrastructures Security*. Springer, 2015, pp. 27–39.

[132] Wireshark. Wireshark. [Online]. Available: https://www.wireshark.org

[133] R. A. Awad, J. Lopez, and M. Rogers, "Volatile memory extraction- based approach for level 0-1 cps forensics," in *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2019, pp. 1–6.

[134] P. Eden, A. Blyth, K. Jones, H. Soulsby, P. Burnap, Y. Cherdantseva, and K. Stoddart, "Scada system forensic analysis within IIoT," in *Cybersecurity for Industry 4.0*. Springer International Publishing, 2017, pp. 73–101. [Online]. Available: https://doi.org/10.1007/978-3-319-50660-9_4

[135] M. H. Rais, R. A. Awad, J. Lopez Jr, and I. Ahmed, "Jtag-based plc memory acquisition framework for industrial control systems," *Forensic Science International: Digital Investigation*, vol. 37, p. 301196, 2021.

[136] A. Abbasi, T. Scharnowski, and T. Holz. Doors of durin: The veiled gate to siemens S7 silicon. [Online]. Available: https://i.blackhat.com/eu-19/Wednesday/eu-19-Abbasi-Doors-Of-Durin-The-Veiled-Gate-To-Siemens-S7-Silicon.pdf

[137] Libmodbus. [Online]. Available: https://libmodbus.org

[138] Libplctag. [Online]. Available: https://github.com/libplctag/libplctag

[139] Modbus-tcp client. [Online]. Available: https://github.com/goddland16/Modbus-TCP

[140] Nodes7. [Online]. Available: https://www.npmjs.com/package/nodes7

[141] J. Searle. Plcscan. [Online]. Available: https://github.com/meeas/plcscan

[142] Pycomm. [Online]. Available: https://github.com/ruscito/pycomm

[143] B. Peterson. Pylogix. [Online]. Available: https://github.com/dmroeder/pylogix

[144] D. Nardella. Snap7. [Online]. Available: http://snap7.sourceforge.net

[145] J. Triplett. Ics mem collect. [Online]. Available: https://github.com/mandiant/ics_mem_collect

[146] M. Pollitt, "A history of digital forensics," in *Advances in Digital Forensics VI*, K.-P. Chow and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 3–15.

[147] "Information technology — Security techniques — Guidelines for identification, collection, acquisition and preservation of digital evidence," Standard ISO 27037:2012, 2012.

[148] "Information technology — Security techniques — Guidelines for the analysis and interpretation of digital evidence ," Standard ISO 27042:2015, 2015.

[149] "Information technology — Security techniques — Incident investigation principles and processes," Standard ISO 27043:2015, 2015.

[150] R. Agarwal and S. Kothari, "Review of digital forensic investigation frameworks," in *Information Science and Applications*, K. J. Kim, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 561–571.

[151] R. Rowlingson *et al.*, "A ten step process for forensic readiness," *International Journal of Digital Evidence*, vol. 2, no. 3, pp. 1–28, 2004.

[152] C. P. Grobler and C. P. Louwrens, "Digital forensic readiness as a component of information security best practice," in *New Approaches for Security, Privacy and Trust in Complex Environments*, H. Venter, M. Eloff, L. Labuschagne, J. Eloff, and R. von Solms, Eds. Boston, MA: Springer US, 2007, pp. 13–24.

[153] T. Kilpatrick, J. Gonzalez, R. Chandia, M. Papa, and S. Shenoi, "Forensic analysis of scada systems and networks," *International Journal of Security and Networks*, vol. 3, no. 2, pp. 95–102, 2008.

[154] R. F. Cassidy, A. Chavez, J. Trent, and J. Urrea, "Remote forensic analysis of process control systems," in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi, Eds. Boston, MA: Springer US, 2008, pp. 223–235.

[155] K. Nance, B. Hay, and M. Bishop, "Digital forensics: Defining a research agenda," in *2009 42nd Hawaii International Conference on System Sciences*, 2009, pp. 1–6.

[156] J. Slay and E. Sitnikova, "The development of a generic framework for the forensic analysis of scada and process control systems," in *Forensics in Telecommunications, Information and Multimedia*, M. Sorell, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 77–82.

[157] Opentext Security. Encase forensic. [Online]. Available: https://www.guidancesoftware.com/encase-forensic

[158] M. Betts, J. Stirland, F. Olajide, K. Jones, and H. Janicke, "Developing a state of the art methodology toolkit for ICS/SCADA forensics," *International Journal of Industrial Control Systems Security (IJICSS)*, vol. 1, 2016.

[159] 504ensicsLabs. Lime: Linux memory extractor. [Online]. Available: https://github.com/504ensicsLabs/LiME

[160] AccessData. Ftk imager - evidence acquisition tool. [Online]. Available: https://accessdata.com/products-services/forensic-toolkit-ftk/ftkimager

[161] E. Chan, S. Venkataraman, F. David, A. Chaugule, and R. Campbell, "Forenscope: A framework for live forensics," in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 307–316. [Online]. Available: https://doi.org/10.1145/1920261.1920307

[162] I. Sutherland, J. Evans, T. Tryfonas, and A. Blyth, "Acquiring volatile operating system data tools and techniques," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 3, pp. 65–73, 2008.

[163] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, and V. Roussev, "Denial of engineering operations attacks in industrial control systems," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 319–329. [Online]. Available: https://doi.org/10.1145/3176258.3176319

[164] S. A. Qasim, J. Lopez, and I. Ahmed, "Automated reconstruction of control logic for programmable logic controller forensics," in *International Conference on Information Security*. Springer, 2019, pp. 402–422.

[165] AccessData. Forensic toolkit (ftk) digital investigations. [Online]. Available: https://accessdata.com/products-services/forensic-toolkit-ftk

[166] Belkasoft. Belkasoft Evidence Center. [Online]. Available: https://belkasoft.com

[167] B. Carrier. The Sleuth Kit (TSK). [Online]. Available: https://www.sleuthkit.org

[168] S. Software. 101 Editor. [Online]. Available: https://www.sweetscape.com/010editor/

[169] S. Mansfield-Devine, "Fighting forensics," *Computer Fraud Security*, vol. 2010, no. 1, pp. 17–20, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361372310701123

[170] Plc-logger. [Online]. Available: https://sourceforge.net/projects/plclogger/

[171] Y. Liu, T. Dillon, W. Yu, W. Rahayu, and F. Mostafa, "Noise removal in the presence of significant anomalies for industrial iot sensor data in manufacturing," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7084–7096, 2020.

[172] A. P. Mathur and N. O. 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.

[173] J. Goh, S. Adepu, K. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," 10 2016.

[174] G. Bernieri, F. Del Moro, L. Faramondi, and F. Pascucci, "A testbed for integrated fault diagnosis and cyber security investigation," in *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, 2016, pp. 454–459.

[175] [Online]. Available: https://www.cdc.gov/

[176] [Online]. Available: https://www.ofwat.gov.uk

[177] Siemens. Delivery release simatic step 7 professional / basic v15.1. [Online]. Available: https://support.industry.siemens.com/cs/document/109758795/delivery-release-simatic-step-7-professional-basic-v15-1?dti=0&lc=en-AE

[178] Rockwell Automation. (9324-rldx) studio 5000 programming software. release notes. [Online]. Available: https://compatibility.rockwellautomation.com/GeneratedReleaseNote.aspx?v1=55163

[179] I. Automation. (2018) Market share of different plcs. [Online]. Available: https://ipcsautomation.com/blog-post/market-share-of-different-plcs/

[180] R. Lopez Perez, F. Adamsky, R. Soua, and T. Engel, "Forget the myth of the air gap: Machine learning for reliable intrusion detection in scada systems," *EAI Endorsed Transactions on Security and Safety*, vol. 6, no. 19, p. e3, 2019.

[181] Y. Wu, J. Zhuge, T. Yin, T. Li, J. Zhu, G. Guo, Y. Liu, and J. Hu, "From exposed to exploited: Drawing the picture of industrial control systems security status in the internet age." in *7th International Conference on Information Systems Security and Privacy (ICISSP)*, 2021, pp. 237–248.

[182] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the shodan search engine to identify internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548214000213

[183] Z. Drias, A. Serhrouchni, and O. Vogel, "Taxonomy of attacks on industrial control protocols," in *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, 2015, pp. 1–6.

[184] C.-C. Andrei, G. Tudor, M. Arhip-Călin, G. Fierăscu, and C. Urcan, "Raspberry pi, an alternative low-cost plc," in *2020 International Symposium on Fundamentals of Electrical Engineering (ISFEE)*. IEEE, 2020, pp. 1–6.

[185] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and it: A survey of insider threat taxonomies, analysis, modeling, and

countermeasures," *ACM Comput. Surv.*, vol. 52, no. 2, apr 2019. [Online]. Available: https://doi.org/10.1145/3303771

[186] J. Slay and M. Miller, "Lessons learned from the maroochy water breach," in *Critical Infrastructure Protection*, E. Goetz and S. Shenoi, Eds. Boston, MA: Springer US, 2008, pp. 73–82.

[187] M. Geiger, J. Bauer, M. Masuch, and J. Franke, "An analysis of black energy 3, crashoverride, and trisis, three malware approaches targeting operational technology systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1537–1543.

[188] 2021. [Online]. Available: https://www.industrialdefender.com/florida-water-treatment-plant-cyber-attack/

[189] Microsoft, "CVE-2021-34527," 2021. [Online]. Available: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-34527

[190] MITRE. (2022) Mitre att&ck matrix for ics. [Online]. Available: https://attack.mitre.org/matrices/ics/

[191] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *1st International Symposium for ICS & SCADA Cyber Security Research 2013 (ICS-CSR 2013) 1*, 2013, pp. 22–29.

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

[193] D. Formby, S. Durbha, and R. Beyah, "Out of control: Ransomware for industrial control systems," in *RSA conference*, vol. 4, 2017.

[194] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, no. 3, pp. 447–489, 2019.

[195] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," *Machine Learning*, vol. 101, no. 1, pp. 59–84, 2015.

[196] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, 2017, pp. 1285–1298.

[197] S. Jose, D. Malathi, B. Reddy, and D. Jayaseeli, "A survey on anomaly based host intrusion detection system," in *Journal of Physics: Conference Series*, vol. 1000, no. 1. IOP Publishing, 2018, p. 012049.

[198] (2017) Global plc market share as of 2017, by manufacturer. [Online]. Available: https://www.statista.com/statistics/897201/global-plc-market-share-by-manufacturer/

[199] S. A. Qasim, J. M. Smith, and I. Ahmed, "Control logic forensics framework using built-in decompiler of engineering software in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 33, p. 301013, 2020.

[200] A. Roderick. Troubleshooting a programmable logic controller. [Online]. Available: https://eepower.com/technical-articles/troubleshooting-a-programmable-logic-controller/

[201] PLC Technician. Five common issues with plcs how to solve them. [Online]. Available: https://www.plctechnician.com/news-blog/five-common-issues-plcs-how-solve-them

[202] Global Electronic Services Repair. Common causes of programmable logic controller failure. [Online]. Available: https://gesrepair.com/programmable-logic-controller-failure/

[203] E. Byres and J. Lowe, "The myths and facts behind cyber security risks for industrial control systems," in *Proceedings of the VDE Kongress*, vol. 116. Citeseer, 2004, pp. 213–218.

[204] F. Schuster, F. M. Kopp, A. Paul, and H. König, "Attack and fault detection in process control communication using unsupervised machine learning," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*. IEEE, 2018, pp. 433–438.

[205] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A comparative study on online machine learning techniques for network traffic streams analysis," *Computer Networks*, vol. 207, p. 108836, 2022.

[206] J. Choi, H. Kim, S. Choi, J.-H. Yun, B.-G. Min, and H. Kim, "Vendor-independent monitoring on programmable logic controller status for ics security log management," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, p. 682–684.

[207] E. H. Budiarto, A. Erna Permanasari, and S. Fauziati, "Unsupervised anomaly detection using k-means, local outlier factor and one class svm," in *2019 5th International Conference on Science and Technology (ICST)*, vol. 1, 2019, pp. 1–5.

[208] G. Berrada and J. Cheney, "Aggregating unsupervised provenance anomaly detectors," in *11th International Workshop on Theory and Practice of Provenance (TaPP 2019)*, 2019.

[209] D. Huang, X. Shi, and W.-A. Zhang, "False data injection attack detection for industrial control systems based on both time-and frequency-domain analysis of sensor data," *IEEE Internet of Things Journal*, vol. 8, no. 1, pp. 585–595, 2020.

[210] N. Anwar, G. Jones, and S. Ganesh, "Measurement of data complexity for classification problems with unbalanced data," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 7, no. 3, pp. 194–211, 2014. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11228

[211] M. Aly, "Survey on multiclass classification methods," *Neural Netw*, vol. 19, no. 1, p. 9, 2005.

[212] F. Osisanwo, J. Akinsola, O. Awodele, J. Hinmikaiye, O. Olakanmi, and J. Akinjobi, "Supervised machine learning algorithms: classification and comparison," *International Journal of Computer Trends and Technology (IJCTT)*, vol. 48, no. 3, pp. 128–138, 2017.

[213] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, "An up-to-date comparison of state-of-the-art classification algorithms," *Expert Systems with Applications*, vol. 82,

pp. 128–150, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417302397

[214] J. Searle. (2015) Plcscan. [Online]. Available: https://github.com/meeas/plcscan

[215] A. Chaudhary, S. Kolhe, and R. Kamal, "An improved random forest classifier for multi-class classification," *Information Processing in Agriculture*, vol. 3, no. 4, pp. 215–222, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214317316300099

[216] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*. Springer, 2001, pp. 420–434.

[217] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[218] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, and J. J. P. C. Rodrigues, "Sdn-enabled multi-attribute-based secure communication for smart grid in iiot environment," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 6, pp. 2629–2640, 2018.

[219] Dragos. (2022) Chernovite's pipedream malware targeting industrial control systems (ics). [Online]. Available: https://www.dragos.com/blog/industry-news/chernovite-pipedream-malware-targeting-industrial-control-systems/

[220] E. Anthi, L. Williams, M. Rhode, P. Burnap, and A. Wedgbury, "Adversarial attacks on machine learning cybersecurity defences in industrial control systems," *Journal of Information Security and Applications*, vol. 58, p. 102717, 2021.

[221] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320318302565

Appendix A

PLC Data Acquisition

This appendix presents a detailed technical description of the acquisition process for Siemens and Allen-Bradley PLC models. The methods presented are specifically related to the content of Chapter 4, but are also used throughout the thesis, especially in Chapter 5 and Chapter 6.

A.1 Siemens PLCs

Siemens PLCs use the proprietary S7 Communications (S7Comm) protocol as an application and session layer protocol, typically placed over ISO TCP (RFC1006) and IP protocols using Ethernet as the physical layer. The S7Comm Protocol uses request-response functionally to issue commands to, and received information back from, the Siemens S7-300 PLC, with each request containing a request function code contained in the parameters of the S7Comm Protocol Data Unit (PDU), illustrated in figure A.1 which demonstrates the encapsulation of an S7Comm PDU within a TCP telegram. The S7 protocol TCP/IP implementation relies on the block-oriented ISO Transport Service on top of the TCP (TPKT), and is wrapped in the TPKT and Connection Oriented Transport Protocol (COTP) protocols, enabling the

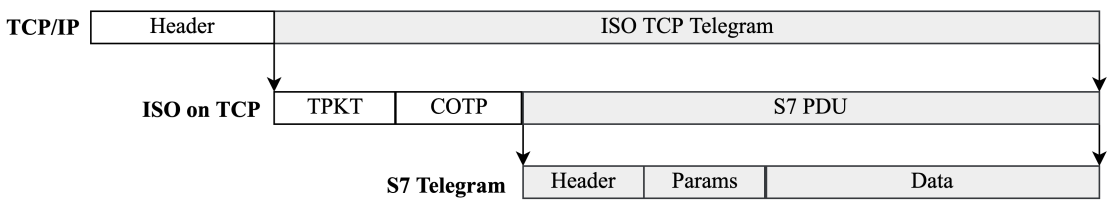


Figure A.1: Siemens S7 Communication Protocol Data Unit (PDU) encapsulation

S7Comm PDU to be carried over TCP.

To acquire device data from the Siemens PLCs, S7Comm packets containing commands can be issued to the target PLC in order to access the contents of the PLC memory, which are crafted using open-source communication libraries. Specifically, for the S7-300 PLC we use Snap7 [144], which consists of an Ethernet communication that leverages the Siemens S7 Communications (S7Comm) Protocol to interface with the Siemens PLCs. The function code $0x04$ is used to issue a read request command for a single or multiple PLC variable(s). The parameter segment of the PDU also contains an access code that references the area of memory to be acquired from the PLC such as an input value using memory parameter $0x81$. Table A.1 provides a list of the variable memory areas that can be accessed on the S7-300 PLC through S7Comm protocol commands crafted using the Snap7 library and corresponding request function code. Furthermore, instances of the PLC application code can be acquired in individual blocks using function codes $0x1d$, $0x1e$ and $0x1f$ to initiate, execute and complete the code acquisition process, respectively. The communication flow for this process and related function codes is illustrated in figure A.2, where $0x1e$ is used to acquire blocks of the PLC application code, such as function blocks or data blocks. If the block is too large to be transferred in a single PDU then the $[0x01]$ flag is appended to indicate that there are additional packets and the remainder of the block will follow.

The structure and syntax of PLC application code is typically different between vendors, with each using their own custom semantics and formats. The application code for Siemens PLCs

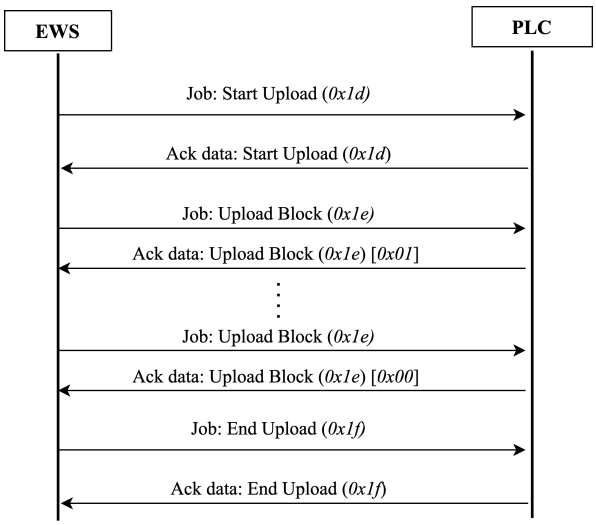


Figure A.2: Communication flow for block (application code) upload from Siemens PLC to EWS including associated function codes

Table A.1: S7Comm Protocol function codes for accessing data stored in PLC memory

Memory Area	Request Function Code
Counters (C)	$0x1c$
Timers (T)	$0x1d$
Inputs (I Memory)	$0x81$
Outputs (Q Memory)	$0x82$
Flags (Markers)	$0x83$
Data Blocks (BD)	$0x84$
Instance Data Blocks (DI)	$0x85$
Local Data (L)	$0x86$

assumes a block-based approach, where there are multiple block types for different functions. Functions (FC) and Function Blocks (FB) are similar to subroutines and typically contain the control logic algorithms related to components in the underlying physical process, including sensors and actuators. They can be called at different points within the application code without the need to repeat the logic. The primary difference between FC and FB is how they store variable data, where the former requires a separate Data Block (DB) to ensure that values are retentive. However, both FBs and FCs are often connected to multiple DBs for data storage, depending on the application context. Organisation Blocks (OB) are the interface between the application code, which is defined by the end-user, and the OS/firmware of the PLC. There are many types of OB that can be used for different purposes, such as time of day interrupts or hardware interrupts. The most important OB is *OB1*, which is the default block for cyclic execution of the PLC application code, as illustrated in figure A.3.

The S7-1500 PLC uses an extended version of the S7Comm Protocol called the Siemens

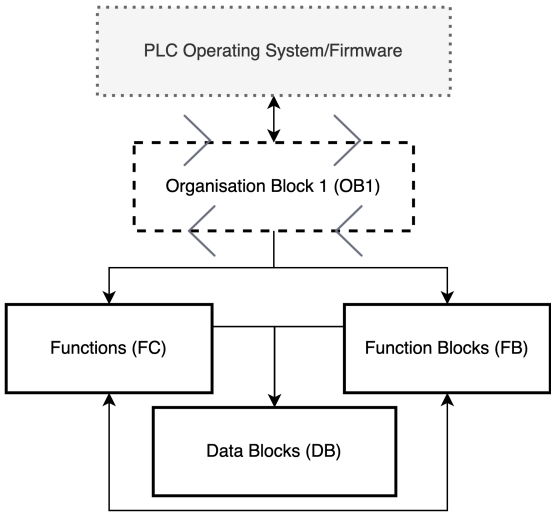


Figure A.3: Simplified Siemens PLC application program block structure

S7 Communications Plus (S7Comm-Plus) protocol. The S7Comm-Plus Protocol is used by contemporary Siemens PLCs, including the S7-1500 and S7-1200 variants, and introduces security controls into the PLC's design, including integrity checking and communication session encryption. It was known prior to conducting the experiments that the Snap7 library has limited support for these models of Siemens PLC and so a second communications library was used for the Siemens S7-1500 PLC called NodeS7 [140]. The NodeS7 library, which is a Node.js module, operates in a similar way to Snap7 by using a network request-response architecture for data acquisition. It is important to note that available documentation outlining the specifications and structure of the S7Comm-Plus Protocol is very limited, and there is no publicly available documentation provided by Siemens. Existing knowledge of the protocol structure is based on previous research and technical work.

A.2 Allen-Bradley PLCs

Allen-Bradley PLCs use a variant of the Common Industrial Protocol (CIP) called EtherNet/IP for application, presentation and session layer functions. Within a CIP packet, Classes, Instances and Services are the main elements that invoke a function. Since Classes represent an abstract element of the PLC, they are of particular interest. The Service element describes the specific function to be invoked from a particular Class, such as reading or writing data. An Instance describes a specific Class object; for example, two different PLC tags might be described by the same Class but are represented as two separate Instances, similar to object-orientated programming languages. Figure A.4 illustrates the packet frame for one CIP packet over EtherNet/IP and the CIP frame structure. Two open-source communication

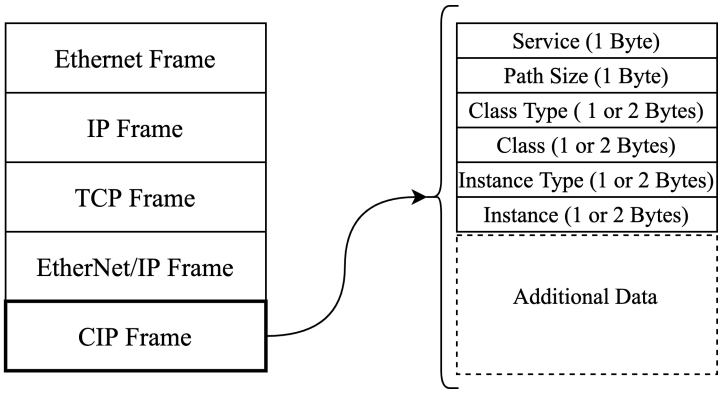


Figure A.4: Structure of the CIP Protocol over EtherNet/IP

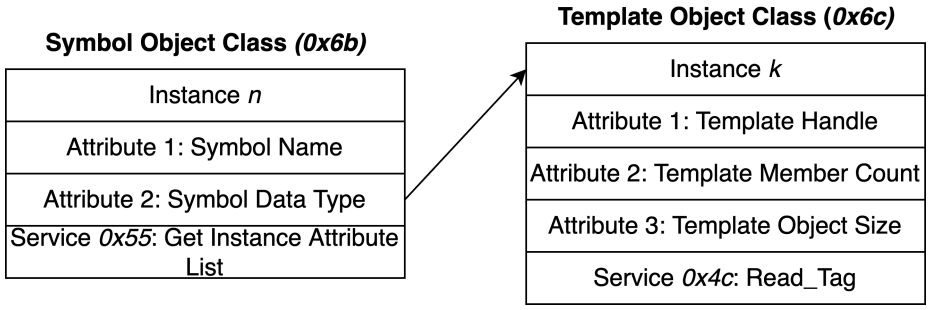


Figure A.5: CIP Service and Class structure to read tag data from PLC memory

libraries were used to acquire data from this PLC, PyLogix [143] and PyComm [142], both of which perform data requests using CIP and EtherNet/IP.

Using the functionality provided by the PyLogix and PyComm frameworks, the values of PLC variables, which are often referred to as tags within the context of Allen-Bradley PLCs, can be read from the controller's memory. There are two types of PLC tag used by Allen-Bradley PLCs. Program tags can be considered as local variables where they are only accessible by the program they are created within, and therefore do not share namespace with other tags that have the same name identifier. Conversely, controller tags are global variables and any program within the controller project can access them. Class $0x6c$ describes various data types that can be accessed, including controller and program tags, and is first invoked through Service $0x55$. In order to specifically access controller tags, we invoke Class $0x6b$, which is referred to as the "Object Class". The relationship between Class $0x6b$ and Class $0x6c$ is presented in figure A.5. It can be seen that the second attribute parameter of the Object Class defines the data type of the target tag, and that the Template Class can be used to to get more information about the data type of the controller tag, for example the size of the tag. Service $0x4c$ is then used in the CIP response to return the data for the requested tags.

The Snap7 library provides pre-defined functions that enable acquisition of the application code from the Siemens PLC. However, this feature does not exist for the AB-CLX PLC in either of the PyLogix or PyComm libraries. Hence, this limitation is addressed by dissecting and reverse engineering the CIP and EtherNet/IP protocols to better understand the communication process of acquiring application code from the AB-CLX PLC. Reverse engineering was used primarily because CIP is an object-orientated protocol where each packet contains well-specified attributes.

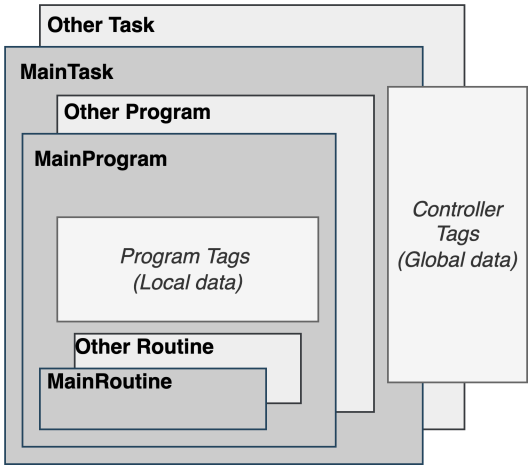


Figure A.6: Simplified Allen-Bradley PLC application program structure with tasks, programs and routines

Allen-Bradley PLC application code is structured differently to Siemens PLCs, and uses *Tasks*, *Programs*, and *Routines* instead of blocks. Tasks are the highest level object in the application code of Allen-Bradley PLCs. They hold information required to schedule the execution of programs within a task, and also set the priority of execution for one or more programs. The continuous task, which is referred to as the MainTask, is executed cyclically. There is only one MainTask task, however Periodic tasks can be used to perform interrupts on the continuous task, and execute for a specific duration and at defined intervals. A task is composed of least one program, cannot be executed concurrently and so only one may be used at a time. The MainProgram is the default program. Routines contain the control logic expressions, and are similar to FCs and FBs used by Siemens PLCs. The MainRoutine is always executed first and is used to call up other routines, referred to as subroutines, within the same program (MainProgram). The structure of Allen-Bradley PLC application code is illustrated in figure A.6.

A connection between the AB-CLX PLC and Rockwell Studio 5000 programming software was established and passively observed through the packet capturing tool Wireshark [132]. By observing the download of a large project, containing application code written in ladder logic, to the PLC, we identified which CIP and EtherNet/IP packets are called multiple times to transmit the code. The programming structure for Allen-Bradley PLCs consist of tasks, programs and routines to manage and execute code. Tasks are at the highest level of the process and control how and when code gets executed, and are then organized into separate programs. Routines are similar to methods in conventional programming languages, and

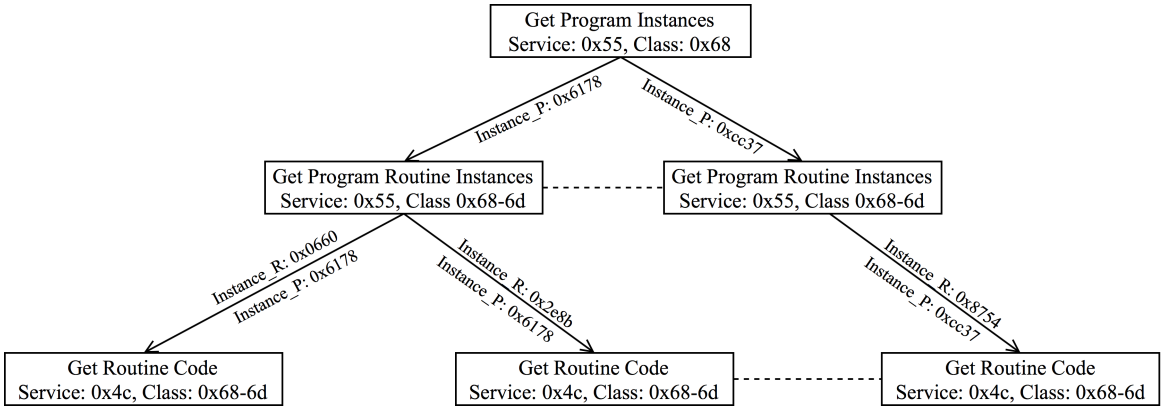


Figure A.7: Acquisition of Allen-Bradley PLC MainRoutine from the application code

contain the control logic code and other data such as variable declarations, with the default routine being the MainRoutine (inside the default MainProgram).

By observing the interactions between the PLC and Studio 5000 programming environment on the workstation while the project was being downloaded, three classes were identified that were of interest; Class $0x68$; Class $0x6d$; and Class $0x6b$. The combination of Classes $0x68$ and $0x6d$ ($0x68 - 0x6d$) is used to transmit the raw code to the PLC from the EWS and represents the application code routines, while the combination of Classes $0x68$ and $0x6b$ ($0x68 - 0x6b$) is used to access the meta-data of a particular routine. The Pylogix library provides open functionality to craft and send CIP packets, enabling the manual generation of these CIP requests to acquire the application code from the AB-CLX PLC. By using Service $0x55$ with Class $0x68$, we are able to retrieve the instances of the programs available on the PLC, in this case the default MainProgram. Furthermore, through the same Service ($0x55$), but with Class $0x68 - 0x6d$, we can acquire instances of all the routines for each program. As the application code is programmed in ladder logic, the retrieved data was separated into multiple sections that reflect the rung-structure of ladder logic. Each rung contains a specific piece of code, describing the relationship between an input and output, which can be accessed by using Service $0x4c$ with the appropriate Instances of Class combination $0x68 - 0x6d$. figure A.7 illustrates this process.

Appendix B

Full PLC Artefact Data Type (ADT) Table

This appendix presents the full PLC artefact data type (ADT) table of results presented in Chapter 4.

Table B.1: PLC Data Types and Artefacts

Data Type and Artefacts	Description	S7-300	S7-1500	AB CLX
ADT 1: Variable Data Content	Dynamic data that is updated through the PLC's operation cycle			
Digital input values	Logic values captured from current states of inputs from the PLC's input I/O module(s)	✓	✓	✓
Digital output values	Logic values captured from the current states of outputs from the PLC's output I/O module(s)	✓	✓	✓
Bit memory values	Values of user-defined areas of memory that can be retentive or non-retentive, depending on how they are configured	✓	✓	✓
Program timer values	Variable data about timers that includes values of current elapsed time and total time	✓	✓	✓
Program counter values	Variable data on counters that includes values of current count	✓	✓	✓
Analog Input Data	Variable data related to readings from analog components that typically measure physical quantities such as temperature or pressure	✓	✗	✓

Continued on next page

Table B.1 – continued from previous page

Data Type and Artefacts	Description	S7-300	S7-1500	AB CLX
Analog Output Data	Variable data issued from the PLC to control analog components, for example a speed frequency to a variable-speed drive (VSD)	✓	✗	✓
ADT 2: PLC Application Program	The static PLC application, typically written by users/operators in one of the standardised PLC programming languages according to ISO/IEC 61131-3, such as ladder logic			
Tag data	Data regarding user-defined tags, which are names given to specific addresses in PLC memory. Tag names and the assigned memory addresses were able to be captured.	✗	✗	✓
Application program logic code	This is the raw application code stored on the PLC that contains logic and constructs used by the PLC to complete a process/task	✓	✗	✓
Application program meta data	This includes data about the PLC's application program such as memory size, vendor-defined checksums, program name(s) and timestamps.	✓	✗	✓

Continued on next page

Table B.1 – continued from previous page

Data Type and Artefacts	Description	S7-300	S7-1500	AB CLX
ADT 3: PLC Meta Data	Data on the PLC hardware and technical specifications.			
PLC system time and date	The PLC's CPU time and date, often referred to as the 'time of day' (TOD)	✓	✓	✓
CPU module information	Hardware information on the PLC's CPU module including the model, version and serial numbers.	✓	✓	✓
Communication processor(s) information	Information regarding any additional communication processor modules attached to the PLC's backplane that extend the PLC's network interfaces (e.g. Modbus/TCP or Profibus.	✓	✗	✓
PLC security configurations	Information regarding currently set inherent security controls and configurations, such as password protected CPUs.	✓	✗	✓
PLC firmware version	The version of the current on-board firmware of the PLC's CPU module.	✓	✗	✓

Continued on next page

Table B.1 – continued from previous page

Data Type and Artefacts	Description	S7-300	S7-1500	AB CLX
ADT 4: Device Diagnostics and Logs	Data and information reported and recorded through inbuilt PLC functionality or through the PLC vendor programming software environment.			
Operation mode status	Indication of the current PLC operation mode, normally either STOP, RUN or PROGRAM.	✓	✓	✓
Firmware updates	Recorded logs indicating current or previous updates to PLC module firmware.	✓	✓	✓
Application program downloads	Recorded logs indicating a program download to the PLC from the vendor programming software such as Siemens TIA Portal.	✓	✓	✓
Hardware/network faults	Any diagnostics or log relating to PLC system faults occurring from a network or hardware configuration error.	✓	✓	✓

Appendix C

Full PLCPrint Detection Results

This appendix provides the full table of results for the PLC Memory Fingerprinting Attack Detection evaluation that is part of the larger PLCPrint model presented in Chapter 6.

	One-Class Support Vector Machine (OCSVM)								K-Nearest Neighbour (K-NN)							
	Dynamic				Static				Dynamic				Static			
	S7-300		AB-CLX		S7-300		AB-CLX		S7-300		AB-CLX		S7-300		AB-CLX	
Test Data Size (PMRMs)	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
200	0.95	0.97	0.86	0.9	0.94	0.97	0.93	0.96	0.82	0.91	0.86	0.92	0.94	0.97	0.93	0.96
400	0.94	0.95	0.86	0.89	0.9	0.92	0.89	0.94	0.79	0.87	0.82	0.89	0.88	0.92	0.86	0.93
600	0.88	0.92	0.8	0.89	0.85	0.89	0.84	0.89	0.77	0.86	0.82	0.89	0.8	0.89	0.85	0.87
800	0.86	0.91	0.82	0.88	0.84	0.87	0.89	0.93	0.76	0.86	0.81	0.88	0.78	0.88	0.88	0.94
1000	0.86	0.92	0.78	0.88	0.88	0.87	0.84	0.91	0.76	0.86	0.8	0.86	0.78	0.88	0.83	0.89

Table C.1: Full Attack Detection Results using OCSVM and K-NN Algorithms