PLCPrint: Fingerprinting Memory Attacks in Programmable Logic Controllers

Marco M. Cook, *Member, IEEE,* Angelos K. Marnerides, *Member, IEEE,* and Dimitrios Pezaros, *Senior Member, IEEE*

***Abstract*—Programmable Logic Controllers (PLCs) constitute the functioning basis of Industrial Control Systems (ICS) and hence are often a focal point for attackers to exploit. Previous attacks have seen PLC memory maliciously altered in order to disrupt the underlying physical process. Different types of memory attack can cause a similar impact on the PLC's operation and result in indistinguishable physical manifestations. Consequently, delays in triaging attacks through digital forensic practices can induce significant financial loss, physical damage to the infrastructure, and degradation of safety. In this work, we propose PLCPrint, a novel vendor-independent fingerprinting approach that utilises PLC memory artefacts to perform detection and classification of memory attacks. PLCPrint uses PLC memory register mapping, a novel method exploiting the relationship between PLC registers and memory artefacts including the PLC application code. Through this, registers are assigned a Mapping Condition (MC) to indicate how they exist within the PLC memory artefacts. We evaluate the performance of PLCPrint over realistic emulations conducted at a real testbed emulating water filtration and distribution. Through PLCPrint we depict how MC deviations are utilised within supervised learning schemes such as to adequately classify PLC memory attacks with high accuracy performance. In general, we demonstrate that PLCPrint fills the gap in the context of attack technique triaging since this has been a missing element within current ICS forensics schemes.**

***Index Terms*—Anomaly Classification, Anomaly Detection, Cyber-Security, Digital Forensics, Industrial Control Systems, Machine Learning, Programmable Logic Controllers**

I. Introduction

CONTROLLING and monitoring automated physical processes through Industrial Control Systems (ICS) is fundamental to the daily operation of many Critical National Infrastructure (CNI) services. These include but are not limited to water treatment, electrical generation, manufacturing facilities, and transportation. Moreover, ICS are becoming increasingly targeted in cyber attack campaigns executed by advanced adversaries, which could have devastating consequences [1]. ICS have become more interconnected with enterprise Information Technology (IT) systems that use Internet-facing gateways and cloud technologies, further augmenting the cyber threat [2]. Previous attacks have resulted in extreme financial repercussions, the loss of critical utilities, and physical damages through degrading safety controls [3].

Specialised industrial computers, known as Programmable Logic Controllers (PLCs), play a critical role in ICS through electrically controlling actuators and sensors that are part of an underlying physical process. PLCs have become increasingly targeted by numerous real-world ICS attacks, primarily due to their importance within an ICS. A recent attack targeting Israeli water treatment facilities in 2020 and sophisticated malware known as Pipedream developed by the Chernovite threat group are indicative of the continuous effort to maliciously target PLCs in ICS attack campaigns [3] [4].

The threat against PLCs is augmented by the lack of security controls inherent to PLCs. Previous studies have proposed fingerprinting approaches for detecting attacks targeting PLC operations to mitigate the threat [5]–[11]. While many of these studies provide promising results, they are limited to the detection of an attack primarily through modelling normal and anomalous network communication data. As stealth is often a key objective of contemporary cyber adversaries, many PLC attack vectors can be missed by strictly focusing on network traffic as a detection feature set [12]. Consequently, attacks would go unnoticed, delaying the triaging and recovery process resulting in further disruption and damage [13].

In this paper, we introduce PLCPrint, a fingerprinting framework for PLCs that enables attack detection and classification using real-time composition of PLC memory fingerprints through correlating the static and dynamic behaviour of PLC registers. PLC registers are individual variables linked to specific register areas, such as inputs or outputs, and often change while the PLC is running. Registers are then mapped against PLC memory artefacts including run-time memory snapshots and the PLC application code, often referred to as Ladder Logic, to determine each register's dynamic and static status, respectively. We refer to this approach as PLC memory register mapping. PLCPrint is successfully evaluated over a representative water treatment testbed using two PLC models from different vendors, Siemens and Allen-Bradley, highlighting the generalisability of the methodology. To the best of our knowledge, the approach discussed is novel and exclusive to the PLCPrint framework.

We summarise the contributions of this paper as follows:

1) **PLC Memory Register Mapping:** We present an entirely novel vendor-independent method that assesses the relationship between different properties of PLC registers, specifically determining how registers are used in PLC memory in relation to PLC memory artefacts.
2) **Synthesised PLC Memory Artefacts:** We propose a synthesised data set comprising different PLC memory artefacts, including the PLC application code, which has

All authors are with the School of Computing Science, University of Glasgow, Glasgow, Scotland, G12 8RZ. Corresponding Author: Marco M. Cook, e-mail: marco.cook@glasgow.ac.uk.

TABLE I: Comparison of PLCPrint and Existing ICS Fingerprinting Approaches for Attack Detection and Classification. **Key:** ●= Coverage, ◗= Limited Coverage, ❍= No Coverage. For PLC Focus this denotes Multiple PLCs, Single PLC, No PLC. For Testbed Evaluation this denotes Representative Physical Setup, Standalone Device, No Testbed Evaluation, respectively.

	Objective			**Evaluation**		**Data Artefacts**			
Approach	**Forensic Uses**	**Attack Detection**	**Attack Classification**	**PLC Focus**	**Testbed Evaluation**	**PLC Registers**	**PLC Application Code**	**Network Data**	**Side-channel Data**
Gonzalez & Hinton (2014) [5]	❍	●	❍	◗	❍	❍	❍	❍	●
Peng *et al.* (2015) [6]	❍	●	❍	◗	◗	❍	❍	●	❍
Formby *et al.* (2016) [7]	❍	●	❍	❍	◗	❍	❍	●	●
Yau & Chow (2017) [14]	◗	●	❍	◗	❍	●	❍	❍	❍
Ahmed *et al.* (2018) [15]	❍	●	❍	◗	●	❍	❍	❍	●
Shen *et al.* (2018) [16]	❍	●	❍	❍	❍	❍	❍	●	❍
Chan *et al.* (2019) [9]	❍	●	❍	◗	❍	●	❍	❍	❍
Ghazo & Kumar (2019) [17]	❍	❍	❍	●	●	❍	❍	●	❍
Stockman *et al.* (2019) [10]	❍	●	❍	◗	❍	❍	❍	❍	●
Ahmed *et al.* (2020) [18]	❍	●	❍	◗	●	❍	❍	❍	●
Yang *et al.* (2020) [11]	❍	●	❍	●	◗	●	❍	❍	❍
Yimer *et al.* (2020) [19]	❍	●	❍	❍	❍	❍	❍	●	❍
Ahmed *et al.* (2021) [20]	❍	●	❍	●	●	❍	❍	❍	●
PLCPrint	●	●	●	●	●	●	●	❍	❍

not been explored in previous studies within the context of PLC anomaly detection and classification.

3) **PLC Memory Attack Vector Model:** To evaluate PLCPrint, we propose a threat model targeting different areas of PLC memory. We developed several attack scenarios that use this threat model to target PLCs controlling a representative ICS testbed. We expand on previous work by categorising types of attacks through the MITRE ATT&CK Framework.
4) **Memory Attack Type and Technique Classification:** Through the memory register mapping approach, we demonstrate high performance for PLC attack type and technique classifications by comparing multiple machine learning algorithms to rapidly enable the initial stages of ICS digital forensics.

The remainder of this paper is structured as follows. Section II discusses the literature related to PLC fingerprinting approaches. Section III introduces the PLCPrint fingerprinting model and methodology. Section IV discusses the threat model and attack scenarios that target PLC memory. Our evaluation of PLCPrint is provided in section V. Finally, section VI concludes this paper.

II. Background and Related Work

Fingerprinting is a technique often used in security that uses a set of correlated data points to define the behaviour of an application, device, or system. Table I presents a comparison of the key features from previous ICS fingerprinting studies with the PLCPrint model that our paper presents to emphasise where the key novelty lies. We separate features into three categories; i) approach objective, ii) approach evaluations, and iii) data artefacts used.

A. Objectives of ICS Fingerprinters

The majority of existing fingerprinting approaches for ICS provide functionality for anomaly detection [5]–[7], [9]–[11], [14]–[16], [18]–[20]. There is an insubstantial number of studies that have explored the crossover between anomaly detection and forensic response, with only one study being identified to have limited forensic applications through timestamps associated with state changes [14]. However, this is a passive form of forensic analysis and only provides a benefit to post-incident analysis. To the best of our knowledge, the use of PLC fingerprinting for automated real-time attack detection and classification using PLC memory artefacts has not been explored in the literature at the time of writing. A novel approach proposed in [21] demonstrates how comprehensive images of PLC memory can be acquired remotely and during run-time. Although this approach does not provide utility for anomaly detection or attack classification, it emphasises that the contents of PLC memory provides crucial artefacts for post-incident understanding of attack provenance.

B. Evaluating ICS Fingerprinters

The comparison in table I separates evaluation into two elements. Firstly, *PLC focus* examines the generalisability of the proposed fingerprinting approach and whether it has been tested on one or more different PLC model or vendor. Very few studies perform evaluations on multiple PLC models, with most only evaluating one model, resulting in challenges regarding generalisation of the methodology. Secondly, fingerprinting approaches have been evaluated on representative ICS testbeds [22] or, more commonly, using publicly available existing datasets [15], [17], [18], [20].

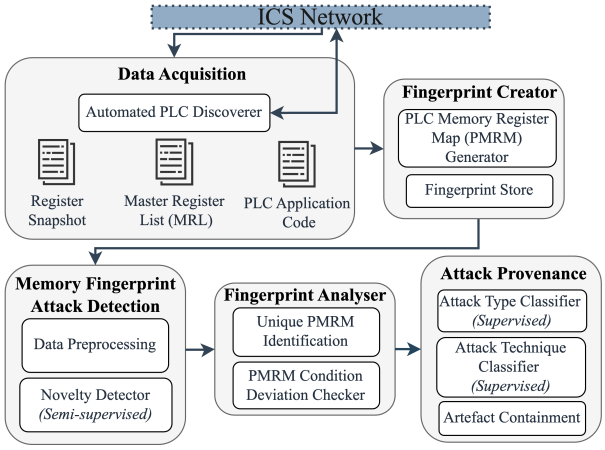


Fig. 1: PLCPrint high-level architecture.

C. Fingerprinting Data Artefacts

A range of different data artefacts have been used as properties of ICS fingerprinting approaches. Several previous studies have used characteristics of ICS network data in fingerprinting methods for anomaly detection [6], [7], [16], [17], [19]. As ICS network traffic is primarily deterministic, models of normal system behaviour are able to be accurately generated in comparison to generic IT systems, where there would typically be a greater amount of noise in the network data [23]. Other studies used the dynamic states of device input/output (I/O) registers to fingerprint ICS processes [9], [11], [14]. None of the studies in Table I have looked to use PLC application code as an artefact for device fingerprinting. In addition, few approaches have used multiple data sources for the fingerprinting approach [7], and no identified studies have examined a hybrid data model using PLC registers and application code in synergy. A selection of works conducted by Ahmed *et al* in 2018 [15], 2020 [18] and 2021 [20] present fingerprinting approaches that use side-channel data sources to identify unique patterns governing the underlying industrial process such as power and electromagnetic waves.

III. Methodology

As presented in section II, there are a number of research limitations with existing ICS fingerprinting approaches, particularly regarding their utility to digital forensics applications. In this section, we introduce PLCPrint, an attack detection and classification framework for PLCs. The high-level architecture of PLCPrint comprising five modules is presented in Fig. 1.

A. PLC Memory Artefacts

PLCs are used in a variety of industrial applications where they are physically connected to different sensors and actuators either directly into the PLC's Input/Output (I/O) modules or through a remote I/O interface. These physical entities are accessed and manipulated through the PLC's programmable logic. PLC memory artefacts have previously been key targets in a variety of real-world and proof-of-concept PLC malware and wider ICS attacks [3], [12].

1) PLC Operation States: A PLC has a finite set of defined states S that it can be in, which are dependent upon how the PLC has been programmed and the physical process it controls. We define a set of states as; $S\{s_1, s_2, \ldots, s_n\}$ where $n \in \mathbb{N}$ and dependent on the physical process. 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:

$$S^{ph}\{\frac{X^{ac}}{X^{in}}\} + \{\frac{Y^{ac}}{Y^{in}}\}. \tag{1}$$

2) PLC Registers: 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:

$$S^{lo}\{r_1, r_2, r_k\} \subset S^{lo}\frac{x^{ac}}{x^{in}} + \frac{y^{ac}}{y^{in}} \tag{2}$$

PLCs use different register areas to generate dynamic variables that can change during the operation of the physical process under control. We defined R as the super-set of registers that a PLC uses from the 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: $I\{i_1, i_2, \ldots, i_v\}$, $Q\{q_1, q_2, \ldots, q_w\}$, $H\{h_1, h_2, \ldots, h_x\}$, $P\{p_1, p_2, \ldots, p_y\}$, $C\{c_1, c_2, \ldots, c_z\}$. All register sets I, Q, H, P, C have a set of active (ac) and inactive (in) registers, for example:

$$I\{I^{ac} \frown I^{in}\}, Q\{Q^{ac} \frown Q^{in}\} \ldots \tag{3}$$

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

$$s_n^{l\ o} = \frac{I \frown Q \frown H \frown P \frown C}{t} \tag{4}$$

3) PLC Application Code: 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 [24]. PLCs interface with sensors and actuators by assigning them to registers though this programmable logic. PLC application code is programmed by combining registers with program objects, which determine how the corresponding register is used and how the proceeding logic should be executed. 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 functions in the application code. Therefore:

$$\forall r \in R\ \exists r \in f_m \tag{5}$$

B. PLCPrint Enumeration

1) Automated PLC Identification: 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, we use and extend the functionality of an existing Python-based network reconnaissance tool called PLCScan[1]. 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. Identifying the PLC vendor and model is an important first step in PLCPrint as although our general methodology is vendor-independent, different software scripts have been developed to access the memory artefacts from specific PLC models.

2) Defining PLC Master Register List (MRL): Once a PLC is identified, PLCPrint defines the accessible registers, referred to as the Master Register List (MRL) within five memory register areas defined in section III-A, I, Q, H, P, C from the PLC, as illustrated at stage 1 in Fig. 2. We have implemented a hybrid approach to this where PLCPrint can either automatically populate the MRL from a predefined range based on the 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.

C. PLC Data Artefact Acquisition

1) Register State Snapshot: Fig 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. Each register is assigned 0 if inactive and 1 if active in PLC memory at the time we acquire the register state snapshot. I, Q and H registers are naively in discrete format, however P and C registers are not as they generate analog data signals such as time elapsed and counter values. Hence, to process a P or C register, the register's current value is compared to its value in the previous dynamic register snapshot. If the value has changed, then the state of the timer or counter object is set to be True for that specific snapshot. 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.

2) PLC Application Code: An image of the application code is acquired from the PLC using over-the-network data acquisition techniques, which have been seen in previous research as a viable method for acquiring data from PLCs [25]. Specifically, we utilise the Snap7 communications library that enabled data acquisition from Siemens PLCs running the S7Comm protocol[2]. For the Allen-Bradley PLCs, we used an approach implemented in our previous research that extended the PyLogix[3] library supporting the EtherNet/IP protocol [26].

[1]https://github.com/meeas/plcscan

[2]http://snap7.sourceforge.net/

[3]https://github.com/dmroeder/pylogix

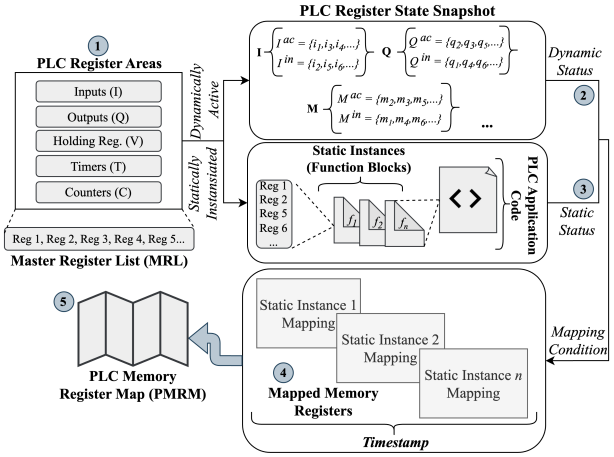


Fig. 2: PLC Memory Register Map (PMRM) generation illustrating process of defining dynamic and static status for each register in the Master Register List (MRL).

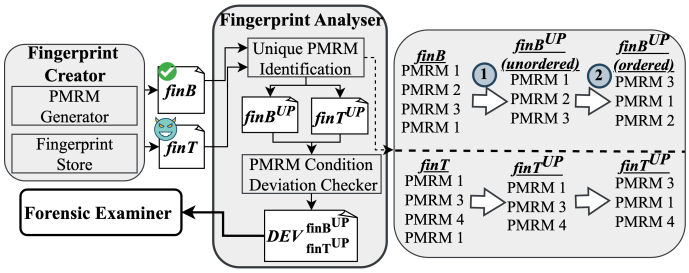


Fig. 3: PLCPrint PLC Memory Register Mapping (PMRM) Analysis Process. The fingerprint creator supplies the $finB$ baseline and the $finT$ test fingerprints into the analyser.

D. 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 MC of a register, resulting in a *PLC Memory Register Map* (PMRM).

1) Register Dynamic Status: 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 often 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 the physical process, however transitions would be expected to be deterministic based on the execution of the PLC application code. For instance, the dynamic status of a register may be 0 at time t_k but change to 1 at t_{k+1}. Hence, PLCPrint accumulates multiple PMRMs over a period of time, which is user defined, by continuously monitoring the value of a register and determine whether it is active or inactive. These values are then collectively compiled into the PMRMs to generate a fingerprint dataset.

2) Register Static Status: 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 the static instance, 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.

3) Register Mapping Conditions (MC): Once a value has been assigned to a given register's dynamic status and static status, we determine the register's final Mapping Condition (MC), which is based on a truth table with four conditions. The dynamic and static statuses are cross-referenced resulting in the register being assigned one of four MC, 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: $r_n \in f_m \wedge r_n \in R^{ac}$

E. 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 (i.e., stage 4 in Fig. 2), and outputs this as a single PMRM shown in stage 5. The size of a PMRM is dependent on the number of static instances derived from the PLC application code, since different static instances are very likely to include contrasting register instantiations within the underlying control logic. The timestamp is calculated at the start of stage 2 in the mapping process and is used in subsequent stages of analysis and digital forensics. 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 multiset 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.

F. Attack Detector

The attack detection component of PLCPrint instigates the continuous generation of test fingerprint samples, referred to as $finT$, which are produced at time intervals of t duration in minutes. The attack detector comprises a data pre-processing module, which accesses the fingerprint store and retrieves the active $finB$ dataset containing all of the PMRMs for a particular PLC. $finB$ 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.

G. Fingerprint Analyser

The fingerprint analyser module is presented in Fig. 3. The module firstly examines both the baseline $finB$ and test $finT$ to convert their respective PMRM multisets 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 $finT$. 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.

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, PLPrint then examines which MC, as presented previously, the register has deviated to.

Upon comparing the PMRMs in $finB$ with $finT$, PLCPrint calculates 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 we found that register *I2* changed 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. Hence, 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 $finT$.

H. Attack Provenance

The final element of PLCPrint is responsible for performing the attack classification using the outputs from the fingerprint analyser module. Multi-class classification algorithms are used with the contents of $DEV^{\frac{finB^{UP}}{finT^{up}}}$ as the data input to classify the attack type. To improve the forensic utility of PLCPrint, a copy of the original $finT$ data sample is also obtains, which is then contained as evidence in the fingerprint store. A copy of the PLC memory artefacts acquired to generate the $finT$ are

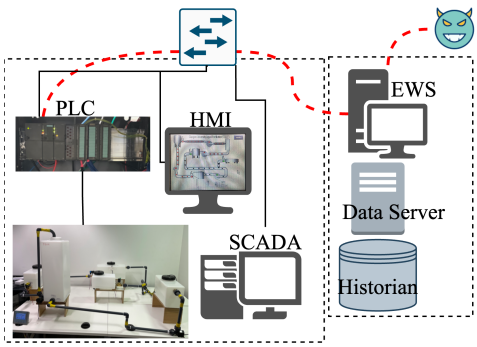


Fig. 4: PLC Memory Threat Model within context of GULP testbed (Red lines indicate attacker path to PLC).

also contained to maintain the chain of evidence and enable data provenance for subsequent forensic analysis.

IV. Experimental Setup and Threat Model

A. Glasgow University Liquid Purification (GULP) Testbed

All of our data sets were generated through the Glasgow University Liquid Purification (GULP) testbed. GULP is a physical ICS testbed that emulates part of the water treatment process, shown in Fig. 4, and features three stages: (i) filtration, (ii) disinfection and (iii) distribution. GULP consists of real physical sensors and actuators such as solenoid values and water pumps to simulate the three water treatment stages.

B. PLC Attack Scenarios

Our threat model depicted in Fig. 4 assumes the position of either an outsider who has gained remote access to the control network through exploited credentials, as seen in previous real-world ICS cyber attacks [3], or an insider who already has physical access to the engineering workstations. The threat model therefore makes the following assumptions:

- The attacker has no prior knowledge of the ICS physical process, however they have performed reconnaissance to target specific PLC models;
- The attacker can access the engineering workstation that has a physical connection to the PLC network;
- The attacker has bypassed PLC password protection controls, if enabled, either through brute-forcing or prior knowledge of hard-coded credentials.

PLCPrint is designed to detect and analyse attacks that target PLC memory artefacts. Therefore, we propose several scenarios that exploit different areas of PLC memory using the MITRE ICS ATT&CK framework as a guide for attack techniques against PLCs [27]. Table II presents the MITRE attack techniques that were identified to target PLC memory artefacts, which we group 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. For instance, a static attack scenario could involve an adversary modifying the logic within the PLC application program by changing a program object that an input or output is assigned to, or altering the use of registers within such logic. Dynamic attack scenarios involve manipulating the run-time data elements of PLC operations, such as brute forcing a sensor to be active. The attack scenarios

TABLE II: MITRE ICS ATT&CK Techniques Targeting PLC Memory Contents.

Attack Type	MITRE Technique	Observed In
Static	Program Download (T0843)	[3], [30]
	Modify Program (T0889) (StaticMP)	[29], [31]
	Modify Controller Tasking (StaticMCT) (T0821)	[3], [12]
Dynamic	I/O Image (DynamicIO) (T0877)	[11], [14]
	Brute Force I/O (DynamicBF) (T0806)	[11]

are generalised and implemented against two PLC vendors. While password protection on PLC application code is now a common security control provided by many ICS vendors [28], we assume that the attacker has bypassed this. Previous studies have demonstrated approaches to circumvent PLC password-protected control logic, for instance by overwriting weak password hashing and or accessing vendor hard-coded credentials that could be exploited [29].

One additional type of insider threat that is not included within the scope of our attack scenarios is the act of physical sabotage. These types of attack can include cutting electrical cables and tampering with power supplies. As 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 is the focus of PLCPrint.

V. Evaluation

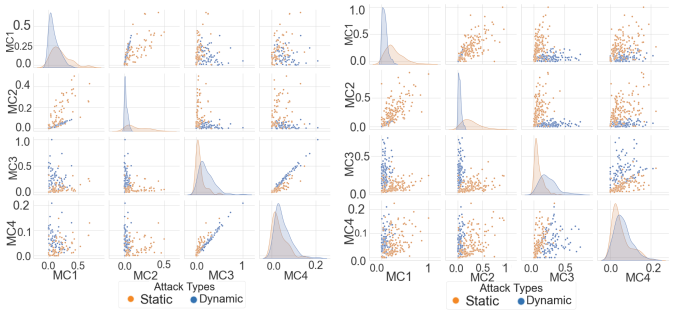
We evaluate PLCPrint with two PLC models from different vendors to determine whether the PLC memory register mapping process can be generalised across PLC vendors. Specifically, we use Siemens S7-300 and Allen-Bradley ControlLogix 1756-L71 (AB CLX) PLCs. Alternative PLC models could be included in future work by extending the functionality of PLCPrint to additional open-source PLC communication libraries implemented for protocols used by alternative leading PLC vendors, including Schneider Electric (Modbus-TCP)[4], General Electric[5], and Mitsubishi (MELSEC)[6]. Such libraries can be used to read PLC registers and acquire various memory artefacts. The existence of these frameworks, and software contributions including decompilers by third-parties, is particularly important due to the lack of tools provided by ICS vendors themselves. However, there are limitations with the ongoing maintenance of such open-source libraries, including providing updates with more recent implementations of vendor protocol stacks, for instance.

We performed 400 individual attack scenario experiments using the GULP testbed, 200 scenarios for each PLC. 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 compared with the original baseline fingerprint $finB$ for a particular PLC.

[4]https://libmodbus.org
[5]https://github.com/mandiant/ics_mem_collect
[6]https://pypi.org/project/pymcprotocol/

	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 III: Novelty Detection Results using OCSVM and K-NN Algorithms.



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

Fig. 5: Distribution of MC features influenced by attack type.

A. Attack Detection Performance

We evaluate the attack detection process through One Class Support Vector Machine (OCSVM) and K-Nearest Neighbour (K-NN) novelty detectors. OCSVM has been used in many previous studies within the literature and so provides a good baseline comparative to related attack detection approaches. K-NN 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.

The accuracy and F1 results for anomaly detection are shown in Table III. Both PLCs show high performance reaching F1 scores of 0.97 and 0.92 for the S7-300 and AB CLX PLCs, respectively, demonstrating the generalisability of PLCPrint at detecting anomalous PLC behaviour. Moreover, the achieved F1 scores are competitive with existing PLC anomaly detection approaches [9], [11], [15]. PLCPrint anomaly detection performs best when the PLCs were subject to static attacks, possibly as static 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.

While both OCSVM and K-NN novelty detectors 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 [32]. Overall, OCSVM achieves an average FNR = $< 0.9\%$, with the K-NN classifier performing slightly worse but not significantly at FNR = $< 1.0\%$. High accuracy measures are also recorded, highlighting that attacks were detected positively in most cases.

Furthermore, 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. 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 requires 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. The steady decrease trend in performance as the quantity of PMRMs increases, also presented in Table III, could be resulting from increased levels of unexpected noise within the test dataset that has impacted the overall distribution of data, particularly when we include larger amounts of attack data points within the datasets.

B. Attack Type Classification Performance

The attack type classification is evaluated using the same 400 datasets acquired for the attack detection evaluation. To classify attack types, we use the deviation in the four MCs presented previously in section III-D as our feature set.

We firstly examine how deviations in the four MCs are dependent on the types of attack. Under dynamic attacks, we identify a significant increase in MC 3 deviations for both PLCs at the time where the attack occurred. 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, as static injection attacks do target modifications to the PLC application code, we see changes in which PLC registers are or are not statically instantiated within a specific static instance of the PLC application code. Hence, we see that static attack scenarios cause a significant increase in the deviations of MC 2 and a lower deviation increase in MC 3. The scale of increase in deviations is not particularly important here as this will depend on external variables such as the number of registers included in the PLC's MRL. However, the significance of the increase in deviations of both MC 2

TABLE IV: Mapping Condition (MC) Feature Set Combinations.

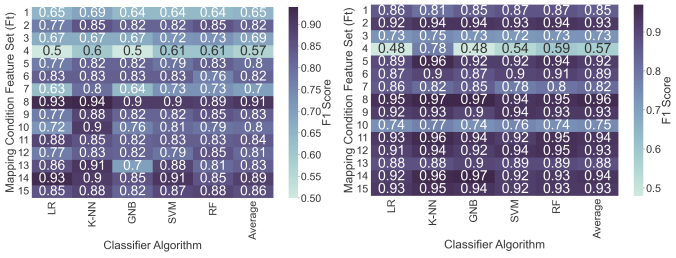
MC Feature	Ft1	Ft2	Ft3	Ft4	Ft5	Ft6	Ft7	Ft8	Ft9	Ft10	Ft11	Ft12	Ft13	Ft14	Ft15
MC1	✓	×	×	×	✓	✓	✓	×	×	×	✓	✓	✓	×	✓
MC2	×	✓	×	×	✓	×	×	✓	✓	×	✓	✓	×	✓	✓
MC3	×	×	✓	×	×	✓	×	✓	×	✓	✓	×	✓	✓	✓
MC4	×	×	×	✓	×	×	✓	×	✓	✓	×	✓	✓	✓	✓

and 3 for static and dynamic attacks respectively highlights the importance of these 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. The relationships between individual MC deviations when the PLCs are subject to dynamic and static attack scenario types are demonstrated in Fig 5. The deviations have been normalised using a standard scalar from 0.0 to 1.0. Moreover, the diagonal graphs in Fig 5 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 MC2. Furthermore, we identify a strong relationship 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 classification approach using MC deviations is highly generalisable, despite technical differences in how PLC models are programmed and configured.

While some of the graphs illustrated in Fig 5 display clearer clustering of features 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 by running the datasets through five supervised machine learning classifiers. Specifically, we used Logistic Regression (LR), K-Nearest Neighbour (K-NN), Gaussian Naive Bayes (GNB), Support Vector Machine (SVM) and Random Forrest (RF) classifiers due to their applicability to smaller feature sets and low overhead. We arranged 15 feature sets based on the different possible MC combinations, which are presented in Table IV.

The subsequent F1 score results are presented as two heat maps illustrated in Figs. 6a and 6b for the AB CLX and Siemens S7-300 PLCs, respectively. Each performance test provides two F1 scores for classifying dynamic and static attack scenarios, respectively. The F1 scores in Fig. 6 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 IV). An 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

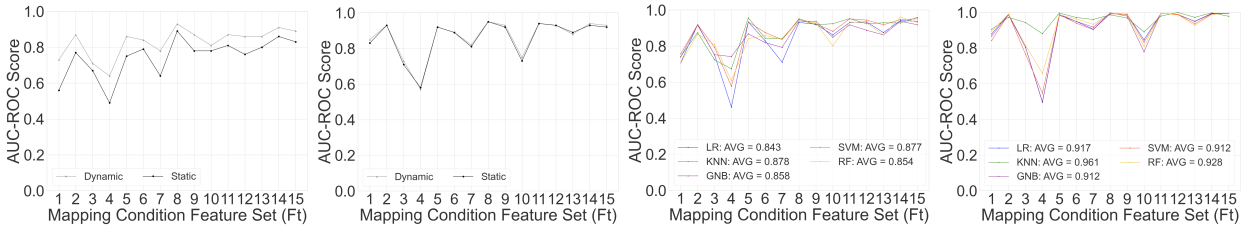


(a) AB CLX PLC (b) S7-300 PLC

Fig. 6: Heat maps showing the average F1 scores for each classifier with each MC feature set as described in table IV. (Average performance of dynamic and static attacks.

both PLCs. Feature set Ft8, which combines MC 2 and 3, produces the highest performance for attack type classification for both PLCs, reaching F1 scores with the K-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 K-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 Fig. 5. 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 classifier performance, with some larger feature sets such as Ft10 and Ft13 for the S7-300 PLC in Fig. 6b performing worse than smaller feature sets. Moreover, feature sets that do not contain MC 2 generally perform worse than those that do contain it, emphasising the importance of static statuses when performing PLC attack classification.

Fig. 7 depicts the average F1 and AUC-ROC scores composed from the results of the aforementioned five classifier algorithms. The graphs in Figs 7a and 7b 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 purtubations being possible on the S7-300 PLC compared with the AB CLX, which is a consequence of how the PLCs were programmed



(a) AB CLX PLC F1 Scores. (b) S7-300 PLC F1 Scores. (c) AB CLX PLC AUC-ROC. (d) S7-300 PLC AUC-ROC.

Fig. 7: 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.

to control the GULP testbed physical process. The graphs in Figs. 7c and 7d illustrate the attack classification AUC-ROC scores for each classifier when different feature sets are used. The K-NN classifier performs best with a slightly higher AUC score for the AB CLX PLC and significantly increased for 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 MC 4, identified to be the weakest feature, and omit the strongest feature, MC 2. From the perspective of deploying PLCPrint in a real-world ICS, the K-NN algorithm is particulary beneficial as it involves instance-based learning and can adapt to new training data as it is collected over long periods of time. Thus, K-NN can respond quickly to input changes during real-time.

C. 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 II, can also be differentiated. We demonstrate this analysis only on the S7-300 PLC since it achieved higher performance scores in the aforementioned attack classification stage compared with the AB CLX.

Using the GULP testbed, we execute an additional 200 attack scenarios using the same process and tools as discussed in section IV and conduct an equal number of scenarios for four of the attack techniques presented in table II, 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 PLCPrint methodology is identical to previous experiments and the *finT* datasets generated from the attack scenarios are analysed through the same approach as in section III. We evaluate all datasets using the five classifier algorithms and the eight best performing feature sets from the attack classification evaluation in section V-B.

The plots in Fig. 8 illustrate the distribution of the four different attack techniques when evaluated with pairs of the MC feature sets. The combination of MC 2 and 3 continues to be the most identifiable feature set at distinguishing between individual attack techniques. However, unlike the graphs in Fig. 5, 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.

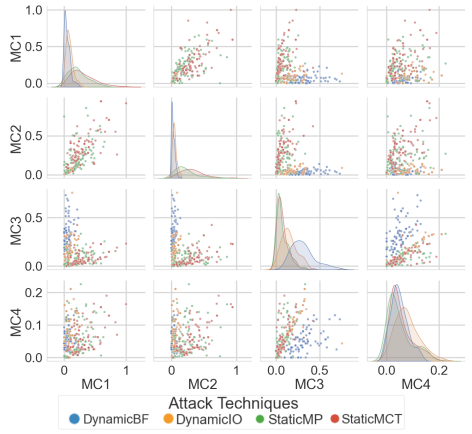
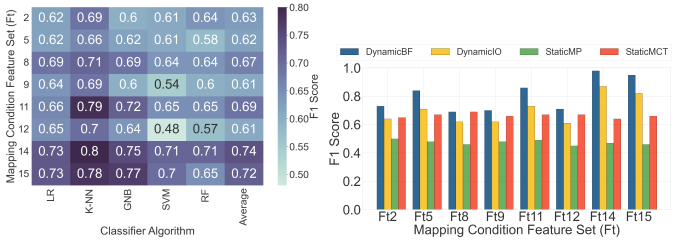


Fig. 8: Attack techniques MC feature set distribution showing greater ambiguity between classification clusters.



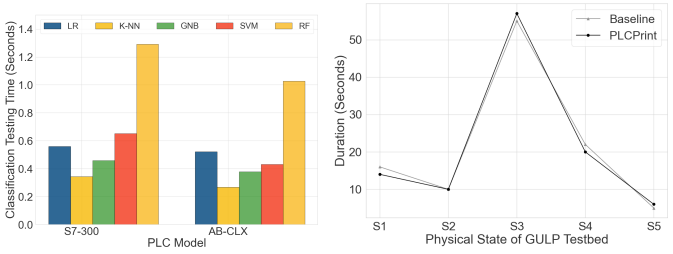
(a) Classifier F1 scores for highest performing MC feature sets. (b) Average F1 scores for classifying attack techniques.

Fig. 9: Attack technique classification performance for S7-300

As shown in Fig. 9a, the resulting F1 scores for attack technique classification are significantly lower than for the attack type classification discussed previously in Fig. 6. Feature sets Ft11 and Ft14 achieve the highest scores of 0.79 and 0.8, respectively, with the K-NN classifier. In addition, we investigate the performance of classifying each individual attack technique, as shown in Fig. 9b. We observe that dynamic attack techniques are better classified in most cases. Attacks that used the *DynamicBF* technique perform best, reaching an F1 score of 0.98 for Ft14. The classification of static attack techniques, in particular *StaticMP* 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.

D. Computational Performance and Overheads

To highlight the rapidity of attack type classification to assist with incident triaging, we evaluate the computational performance of PLCPrint. The data acquisition process of PLCPrint is conducted at a rate of approximately 200 milliseconds, however this rate will fluctuate depending on how many registers are in the MRL and the size of the PLC application code. 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. The consumption time when performing classification testing for the five classifiers is compared for both PLC models, demonstrated in Fig. 10a. Here we see that the K-NN classifier has the lowest test time of approximately 0.3 seconds for both PLCs, which is particularly beneficial as K-NN also achieved the highest classification AUC-ROC and F1 scores, demonstrated in Figs. 6 and 7, respectively. 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. The time consumption demonstrated here are with the full feature set comprising all four MC variables.



(a) Classifier F1 scores for highest performing MC feature sets.

(b) Average F1 scores for classifying attack techniques.

Fig. 10: Attack technique classification for S7-300

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 computational impacts introduced by PLCPrint as particularly valuable. 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.

We firstly evaluate how PLCPrint impacts PLC operations. Fig. 10b illustrates the time consumption for each physical state in the GULP testbed, such as water filtration. The duration for each state while PLCPrint is interfacing with the PLC is always within $\pm$ 2s of the baseline, 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 briefly 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 not responsible for this millisecond increase. The cycle time for the AB-CLX PLC did not change while PLCPrint was performing data acquisition.

E. PLCPrint Limitations

Although PLCPrint is vendor independent, the current implementation of PLCPrint is only evaluated with two vendor-specific industrial protocols. Furthermore, the evaluations attack scenarios that were performed did not include manipulations to live PLC firmware or exploiting vulnerabilities within the firmware code.

Moreover, we note that configurations of certain PLC access controls, such as CPU password protection, could inhibit the acquisition process of PLCPrint, particularly when imaging the PLC application code. However, PLCs are generally likely to enable full or partial access to the registers, such as inputs and outputs, primarily as other devices require access to perform tasks. For instance, human-machine interfaces (HMIs) may require PLC registers in order to display updated sensor values, or where there are logical dependencies between the registers of two PLCs in multi-PLC environments.

VI. Conclusions

In this article, we have presented PLCPrint, a novel fingerprinting approach that enables real-time attack detection and classification for digital forensics. We proposed PLC Memory Register Map (PMRM) generation, a novel method which 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. We evaluated two novelty detection and five supervised classification algorithms for attack detection and attack type classification, respectively. Results from our experimental evaluation, which was conducted over a physical water treatment testbed, reveal high F1 scores for both attack detection and type classification with multiple PLC models, highlighting the vendor-independence of the PMRM generation approach. High accompanying accuracy and AUC-ROC scores demonstrate the generalisation to multiple machine learning models. Generally, PLCPrint assists with the digital forensic process by enabling the chain of evidence and rapid triaging of cyber incidents.

Acknowledgment

This work has been supported in part by EPSRC industrial CASE award grant EP/R511936/1, and by the Defence Science and Technology Laboratory (Dstl) under the Defence and Security Accelerator (DASA) Contract Purchase Agreement (CPA) PA0000000223.

References

[1] 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.

[2] 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

[3] 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.

[4] Dragos, "Pipedream: Chernovite's emerging malware targeting industrial control systems," Dragos, Inc., Tech. Rep., 2022.

[5] C. Aguayo Gonzalez and A. Hinton, "Detecting malicious software execution in programmable logic controllers using power fingerprinting," in *Critical Infrastructure Protection VIII*, J. Butts and S. Shenoi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 15–27.

[6] Y. Peng, C. Xiang, H. Gao, D. Chen, and W. Ren, "Industrial control system fingerprinting and anomaly detection," in *Critical Infrastructure Protection IX*, M. Rice and S. Shenoi, Eds. Cham: Springer International Publishing, 2015, pp. 73–85.

[7] 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*. The Internet Society, 2016.

[8] K. Yau and K.-P. Chow, "Detecting anomalous programmable logic controller events using machine learning," in *Advances in Digital Forensics XIII*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2017, pp. 81–94.

[9] 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.

[10] 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.

[11] 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, pp. 955–967, 2020.

[12] 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.

[13] M. Cook, C. Patterson, A. K. Marnerides, and D. Pezaros, "Anomaly diagnosis in cyber-physical systems," in *IEEE ICC*. Seoul, South Korea: IEEE, 2022.

[14] K. Yau and K.-P. Chow, "Detecting anomalous programmable logic controller events using machine learning," in *Advances in Digital Forensics XIII*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2017, pp. 81–94.

[15] 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*, ser. ACSAC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 566–581. [Online]. Available: https://doi.org/10.1145/3274694.3274748

[16] 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.

[17] 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)*, 2019, pp. 19–24.

[18] 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*, ser. WiSec '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 219–230. [Online]. Available: https://doi.org/10.1145/3395351.3399364

[19] T. Yimer, M. T. Arafin, and K. Kornegay, "Securing industrial control systems using physical device fingerprinting," in *IOTSMS*, 2020, pp. 1–6.

[20] C. M. Ahmed, M. Ochoa, J. Zhou, and A. Mathur, *Scanning the Cycle: Timing-Based Authentication on PLCs*. New York, NY, USA: Association for Computing Machinery, 2021, p. 886–900. [Online]. Available: https://doi.org/10.1145/3433210.3453102

[21] N. Zubair, A. Ayub, H. Yoo, and I. Ahmed, "PEM: Remote forensic acquisition of PLC memory in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 40, pp. 1–10, 2022.

[22] 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.

[23] G. Walkup, S. Etigowni, D. Xu, V. Urias, and H. W. Lin, "Forensic investigation of industrial control systems using deterministic replay," in *2020 IEEE CNS*, 2020, pp. 1–9.

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

[25] 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.

[26] 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.

[27] O. Alexander, M. Belisle, and J. Steele, "Mitre att&ck® for industrial control systems: Design and philosophy," *The MITRE Corporation: Bedford, MA, USA*, 2020.

[28] A. Ayub, H. Yoo, and I. Ahmed, "Empirical study of plc authentication protocols in industrial control systems," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 383–397.

[29] W. Alsabbagh and P. Langendörfer, "A stealth program injection attack against s7-300 plcs," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, 2021, pp. 986–993.

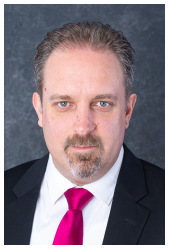
[30] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security*, S. K. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J. Mylopoulos, A. Antón, and S. Gritzalis, Eds. Cham: Springer International Publishing, 2018, pp. 110–126.

[31] J. Lee, H. Choi, J. Shin, and J. T. Seo, "Detection and analysis technique for manipulation attacks on plc control logic," in *Proceedings of the 2020 ACM International Conference on Intelligent Computing and its Emerging Applications*, 2020, pp. 1–6.

[32] M. Almseidin, M. Alzubi, S. Kovacs, and M. Alkasassbeh, "Evaluation of machine learning algorithms for intrusion detection system," in *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, 2017, pp. 000 277–000 282.

Marco M. Cook received his MSc (2015) degree in Software Development (Computer Science) from the University of Glasgow. He has just completed his Ph.D in Computing Science from the same University, funded through an EPSRC iCASE award in partnership with the UK Defence Science and Technology Laboratory (Dstl). Currently, he is a Research Associate in the School of Computing Science at the University of Glasgow. His research focuses on cyber-security for industrial control systems (ICS) and cyber-physical systems more generally, specifically the fields of anomaly detection and digital forensics. He is a member of the IEEE and the ACM.

Angelos K. Marnerides is Senior Lecturer (eq. Associate Professor) in the School of Computing Science (SoCS) at the University of Glasgow (UofG) leading the Glasgow Cyber Defence Group and coordinating all cybersecurity research activities across all the research sections at UofG SoCS. Angelos' research revolves around the formalisation, design and implementation of measurement-based security and resilience mechanisms addressing challenges resulted by the intersection of the IoT with cyber-physical systems that were traditionally isolated (e.g., Critical National Infrastructures). His research has received significant funding from the industry (e.g., Fujitsu, BAE, Raytheon), governmental bodies (e.g., EU, IUK, EPSRC, MoD Dstl, GCHQ NCSC). He has been a member of the IEEE and the ACM since 2007.



Dimitrios Pezaros (M'04-SM'14) received the B.Sc. (2000) and Ph.D. (2005) degrees in Computer Science from Lancaster University. He is (full) Professor of Computer Networks in the School of Computing Science at the University of Glasgow, where he currently holds the Dstl & CINIF / RAEng Research Chair in Digital Resilience for Critical National Infrastructure. He has received significant funding for his research from various funding agencies and industry, and has published widely in the areas of computer communications, network and service management, and network resilience. Prof Pezaros is a chartered engineer, a fellow of the BCS and the IET, and a senior member of the IEEE and the ACM.