Anomaly Diagnosis in Cyber-Physical Systems

Marco Cook, Cory Paterson, Angelos K. Marnerides, and Dimitrios Pezaros
School of Computing Science, University of Glasgow, Glasgow, United Kingdom
m.cook.2@research.gla.ac.uk, 2249177P@student.gla.ac.uk, {angelos.marnerides, dimitrios.pezaros}@glasgow.ac.uk

ICC 2022 - IEEE International Conference on Communications | 978-1-5386-8347-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICC45855.2022.9838968

***Abstract*—Cyber-Physical Systems (CPS) constitute the operational basis for a number of critical national infrastructure (CNI) sectors including but not limited to manufacturing, smart electrical grids and water utilities, where programmable networked systems enable physical processes. Programmable Logic Controllers (PLCs) play a vital role in this by controlling CPS processes and consequently have become a primary target for cyber attacks that aim to disrupt CPS. By contrast with conventional networked setups, the operational and safety-critical importance of PLCs introduce challenges for CNI operators on empirically determining if an incident is a cyber-attack or a system fault as both occurrences can display similar outputs on the physical process. Moreover, existing anomaly detection techniques explicit to PLCs primarily give indication of an incident rather than attempting to categorise what the incident is. In this paper, we introduce a novel PLC anomaly diagnosis framework defined by a two-stage identification and classification approach based on novelty detection. Through the use of PLC run-time and network communication data generated by physical processes on a representational CPS testbed, we achieve an average of 99.35% on anomaly profiling accuracy and highlight the distinctions between system faults and cyber-attacks. In general, we demonstrate a practical approach that can be adopted by next generation CPS cyber defence tools.**

***Index Terms*—Cyber-Physical Systems, Programmable Logic Controllers, Anomaly Detection, Cybersecurity**

I. Introduction

Cyber-Physical Systems (CPS) bridge networked programmable components and physical processes that are executed in a wide range of Critical National Infrastructure (CNI) environments, including but not limited to smart electrical grids, communications, manufacturing, water treatment and healthcare. Industrial Control Systems (ICS) are a core part of CPS enabling remote command and control of the physical processes encountered in these environments. Networked industrial components such as Programmable Logic Controllers (PLCs) increased industrial performance while allowing remote supervision. PLCs are critical to the function of any CPS as they directly interface with and control the underlying physical process by reading sensor (input) values, executing a user-defined program, and updating actuator (output) values. As the end-goal of an ICS attack vector is to disrupt the underlying physical process, PLCs are highly important targets to defend against cyber attacks. Moreover, the lack of vendor-implemented PLC security controls amplifies the cyber threat. Consequently, additional controls, such as anomaly detection must be integrated to mitigate the threat of a cyber-attack.

As a CPS naturally comprises physical processes such as water treatment or manufacturing, system faults and other non-malicious failures often occur and result in similar physical effects as a malicious cyber intrusion. This can make it challenging to distinguish between these types of anomalous events. Therefore, categorising a CPS anomaly can provide major benefits in both live detection and post-incident analysis. Previous literature has explored anomaly detection for PLCs and CPS, however these have primarily focused on solely the identification of anomalous events [1] [2] [3] [4]. In contrast, this paper presents a novel framework for CPS anomaly diagnosis using a synthesised data set that integrates multiple points of PLC run-time data. We explore how PLC run-time data can be used to provide context to identified anomalies, classifying these events as cyber intrusions or system faults. In general, this paper contributions by:

- Providing the first study that verifies the suitability of assessing the frequency of PLC device log generation within anomaly detection in CPS and thus bypassing the need to assess vendor-specific device log content.
- Introducing a novel and vendor-independent anomaly diagnosis framework for PLCs which is the first to distinguish malicious intent from a system fault by effectively integrating multiple PLC run-time data sources.
- Achieving an average F1 score of 99.35% for determining the nature of a given anomaly (i.e., malicious or fault) via emulations conducted within a real CPS test-bed.

The remainder of this paper is structured as follows. Section II briefly reviews existing literature. Section III describes the elements of PLC runtime data that our approach uses. Section IV presents our detection and diagnosis methodology. Section V discusses our results, and Section VI concludes this paper.

II. Related Work

Behaviour-based anomaly detection methods examine trends in normal system activity to identify deviations from normal behaviour. Therefore, intrusion detection systems (IDS) can dynamically adapt to unseen attack vectors, which is important in CPS security as the cyber-threat landscape is ever-changing and dominated by zero-day exploits [5]. Behaviour-based anomaly detection approaches can be categorized as *supervised* or *unsupervised*. Supervised techniques involve training a classifier on labelled samples, learning the trend between data and label, and subsequently attempting to classify unlabelled data accordingly. Labelling CPS anomalous data is very challenging due to the difficulty in acquiring realistic CPS attack data, particularly from live safety-critical CPS. Conversely, unsupervised learning approaches do not require data labelling and have successfully been applied to PLC anomaly detection [1] [6] [2]. One type of anomaly detection that can be

implemented in an unsupervised manner is novelty detection [7]. Novelty detection learns behavior from a baseline data set and isolates new samples that deviate from the baseline. Novelty detection approaches are advantageous for CPS as they do not require extensive or anomalous data labelling.

PLC data artefacts have been used for ICS anomaly detection in previous research, however primarily in a non-synthesised manner. Inoue *et al*, use sensor and actuator values from devices in the Secure Water Treatment (SWAT) data set [2]. The authors found that when analysing patterns in high-dimensional data (51 actuator and sensor values), a clustering approach would not be suitable as large feature sets impact the clustering distance measure. To counter this, we use a simple aggregation technique to reduce the dimentionality of our data sets. Yau *et al*, also use dynamic memory values to inform their anomaly detection method, additionally introducing PLC time values into their data set [1]. Our approach builds on this further by also including PLC counter values, which are often a core part of PLC programming. A recent study by Yang *et al* in 2020 used PLC register states containing dynamic input/output (I/O) values captured passively over the network [4]. While the authors present a promising framework for PLC anomaly detection using device fingerprinting, in isolation, it does not allow for the categorisation of anomalous events.

Settanni *et al* propose an anomaly detection approach that uses PLC log messages and diagnostic buffer data [6]. While the approach is novel, the PLC logs are defined as messages that report values equivalent to analysing the dynamic I/O variables of the PLC. In addition to using PLC logs, the authors create events from the PLC diagnostic buffer. There are limitations with generalising this data between PLC vendors and therefore the approach may not be scalable. Our approach does not rely on the vendor-specifics of PLC logs as we examine the number of logs generated rather than their contents.

III. PLC Runtime Data

PLCs produce both static and dynamic data in order to perform their operational tasks [8]. In this paper, we define as static the data which remains unchanged during the run-time execution of the physical process, such as the PLC application program, often referred to as the ladder logic. In contrast, we define dynamic data as PLC data that is created and changes during the run-time of a PLC controlling the physical process.

When detecting attacks that subtly impact the underlying physical process, it follows that data which correlates to the physical process should be utilised for proactive detection. We can determine if a run-time data source correlates to the physical process if: (i) it's 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.

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 single instance of dynamic I/O values (D), network communications traffic properties (C) and device log data (L) can be defined collectively as a single state (S), linked through a common timestamp (T). As illustrated in Fig. 1, a normally operating PLC can be said to have n number of states S where S_n is defined as:

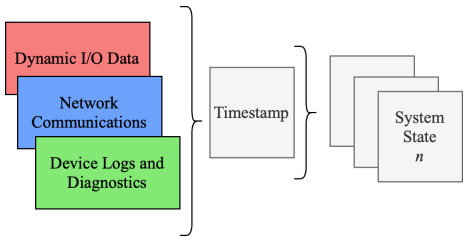


Fig. 1: PLC Run-Time Data Mapped onto PLC System State

$$S_n = T_i(D_n,\ C_n,\ L_n)$$

A. PLC Dynamic I/O Data

We define a PLC's dynamic memory as registers that are changed during the PLC execution cycle by sensors, actuators or the PLC application program. Thus, we compose the following general memory constructs for PLC dynamic memory:

- **Input** (I) and **Output** (Q) Registers: The current value of sensors and actuators respectively.
- **Holding** (M) Registers: Temporary variables for use in ladder logic.
- **Timers** (T): A structure used to denote time duration.
- **Counters** (C): A structure used for enumerating.

Motivated by the notion that dynamic I/O data is used to influence the physical process and the physical process can influence dynamic I/O, a snapshot of this process at a given time can be constructed. This data source represents changes in the physical process, such as a sensor being activated. An instance of dynamic values (D) at time (T), is defined as:

$$D_n = T_i(I,\ Q,\ M,\ T,\ C)$$

We use dynamic I/O data to inform a machine learning classifier in the initial stage of detection.

B. Unidirectional Network Communications

We define a PLC's run-time network communications as *responses* to other devices on the network during the PLC execution cycle. In contrast to other studies [9] [3], this results in a one-way packet flow which is advantageous since traffic is unidirectional data, resulting in significantly reduced overhead. Moreover, protocol-specific acknowledgement packets are returned, which are typically small in size.

In general, network data generated by the PLC during execution does not influence the physical process under control, and state changes in the physical process do not directly generate network data. Consequently, its use in anomaly identification is limited as a dependant relationship between the data and physical process does not exist. However, we identify an absolute deviation in network metrics while a controller is under attack, thus these are used to enrich the incident classification stage of our approach. We use the *average packet length*, the *number of read acknowledgment packets*, and the *number of write acknowledgement packets* as metrics.

C. Device Logs and Diagnostics

PLC device logs and diagnostics are stored in the controller's diagnostic buffer. While it is possible to program custom PLC logs that report on a range of events, these are not consistent across PLC models, thus we only utilise default logs. Querying the PLC logs can be used to analyse system events that have occurred during the PLC execution cycle. The metric calculated on device log data (L) for a given time (T) is the *number of PLC device logs*.

The PLC device logs were acquired by either accessing the PLC's on-board diagnostic buffer containing a chronologically ordered list of logged events through the PLC's programming environment, or by reading the PLC's SD card. We observe that default configured PLC logs related more to monitoring engineering tasks (e.g., hardware changes, operation mode changes) are not affected by changes in the physical process and consequently have significant limitations in aiding PLC attack detection. In addition, an insufficient standardised structure for PLC logs, results in vendors provide contrasting event types, meta-details and formats. This makes generalising logs across PLC vendors challenging due to their differing formats. Hence, the use of PLC device logs in our anomaly identification stage is limited due to both the detachment from the physical process and lack of scalable format between PLC vendors. Despite this, our empirical analysis reveals that the diagnostic buffer is typically populated in situations that could be identified as a system fault. Therefore, PLC device logs are used to enhance the PLC incident classification process.

IV. Methodology and Detection Model

The proposed diagnosis model presented in Fig. 2 consists of the generation of a PLC run-time data set followed by two stages of binary classification; (1) Anomaly Identification, and (2) Incident Classification.

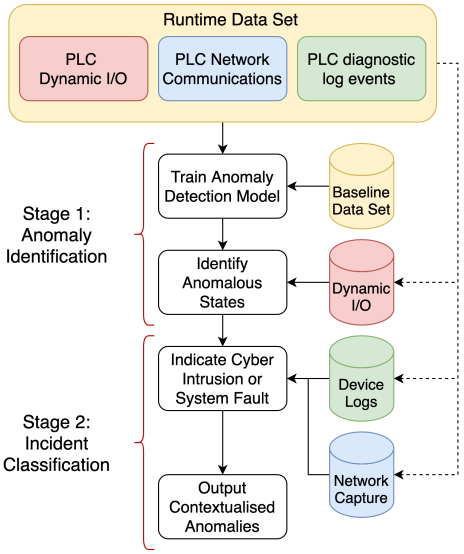


Fig. 2: PLC Run-Time Detection and Classification Model

A. Generation of PLC Run-Time Data

Algorithm 1 describes the process of generating a PLC state represented in our run-time data set. The inputs are the dynamic trace D which comprises a set of PLC dynamic I/O data, a PLC device log file that contains a total of m log events, and a collection of j network communication packets, described in section III. The following subsections describe our approach for acquiring these data sources.

Algorithm 1 Generation of PLC Runtime State

Input: Dynamic Trace $D = \{M_n, I_n, Q_n, T_n, C_n\}$, Log File $L = Log_1, ..., Log_m$, Network Packets $P = pkt_1, ..., pkt_j$
Output: PLC Run-time State

```
1: for vector ∈ D do
2:     if vector instance of T then
3:         for timer in vector do
4:             if timer is not equal to previous timer then
5:                 timer state = True;
6:                 Store timer value as previous
7:             end if
8:         end for
9:     end if
10:    if vector instance of C then
11:        for counter in vector do
12:            if counter is not equal to previous counter then
13:                counter state = True
14:                Store counter value as previous
15:            end if
16:        end for
17:    end if
18:    vector value = encode vector as Base64
19: end for
20: for packet ∈ P do
21:    if packet instance of write acknowledgment then
22:        write acknowledgment count increment by 1
23:    end if
24:    if packet instance of read acknowledgment then
25:        read acknowledgment count increment by 1
26:    end if
27:    accumulate packet length
28: end for
29: mean packet length = total packet length / |P|
30: total logs = |L|
```

1) PLC Dynamic I/O Data: We take an active approach to acquiring dynamic I/O data values from a PLC by querying the controller register states using open-source communication libraries. Table I shows the libraries used and the different network communication properties between the two PLC models explored in our evaluation. The acquired PLC dynamic I/O values are pre-processed by converting each value into a binary representation given some condition related to its behaviour. Lines 1 to 15 in Algorithm 1 show this process. Converting these values to a binary representation allows for analogous analysis of both discrete and continuous data types. The pre-processing conditions are outlined as follows:

- **Inputs and Outputs**: on = 1, off = 0
- **Holding Bits**: on = 1, off = 0
- **Timer**: incrementing/decrementing = 1, stationary = 0
- **Counter**: incremented/decremented = 1, no change = 0

TABLE I: Comparison of PLC Network Communication Properties

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	Pylogix	CIP	0xc1	0xcd

PLC counters and timers are not naturally represented as binary values. Therefore, we process a timer or counter register by comparing the current value to its value in the previous state. If changed, the state of the timer or counter register is set to be true. We obtain five binary vectors each representing an area of dynamic PLC memory. Every binary vector is converted to a Base64 representation and used as a feature within our PLC run-time data set. Converting the binary vectors to Base64 greatly improves the scalability by reducing the dimentionality of the feature set, as large scale industrial systems normally have more than 50 sensors/actuators [10].

2) PLC Network Communications: In parallel with our dynamic I/O acquisition, we execute a packet sniffer module using the Scapy framework [11] to capture all packets sent by a PLC. We also implement a Berkeley Packet Filter (BPF) rule to only receive PLC network responses.

Subsequently, we calculate our defined network metrics. First, the number of dynamic I/O register read/write acknowledgements are obtained through packet inspection using byte identifiers in the application layer as shown in Table I. Second, the mean packet length is calculated for all the captured packets, which can also be indicative of the type of response.

3) PLC Device Log Events: System events are stored within the PLC's CPU diagnostic buffer. The ability to access this data is restricted to methods provided by the PLC's vendor, meaning that a general extraction method could not be implemented in our model. Hence, PLC system events are parsed and accumulated post-acquisition, and represent the total number of device logs generated by the PLC.

B. Stage One: Anomaly Identification

The first stage of our proposed model exploits novelty detection to identify subtle I/O anomalies in the physical process. Since CPS are driven by an underlying physical process with consistent states, they generally have a deterministic operating behaviour. Evidently, this makes CPS well-suited to baseline data set generation under normal operation, which is a fundamental aspect of novelty detection. We selected two novelty detection machine learning algorithms for comparison. The first algorithm is One Class Support Vector Machines (OCSVM) [12] which attempts to fit a hyper-sphere around normal data and classifies anything outwith the hyper-sphere as abnormal. The second novelty detection approach is Local Outlier Factor (LOF) [13]. LOF is described as the anomaly score of each point in the data set which is determined by how isolated it is with respect to data points around it. For use in a novelty detection setting, the LOF score of previously seen samples influences new unseen data points.

C. Stage Two: Incident Classification

The second stage of our model calculates a deviation in network and device log metrics for each anomaly identified in stage one. Each anomaly is diagnosed by classifying the incident as *cyber intrusion* or *system fault*.

Our approach adopts the notion that PLC device logs are typically generated during a situation that resembles a *system fault*, as device logs have been implemented by vendors for this type of event [14]. Conversely, a *cyber intrusion* 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. During our experiments, it is observed that a higher deviation in the PLC network data metrics, indicates a probable cause of a *cyber intrusion*. Conversely, a higher deviation in device log metrics dictates a probable cause of a *system fault*.

In order to classify every corresponding deviation, we first calculate averages from network and log metrics in the baseline PLC run-time data set. Via 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:

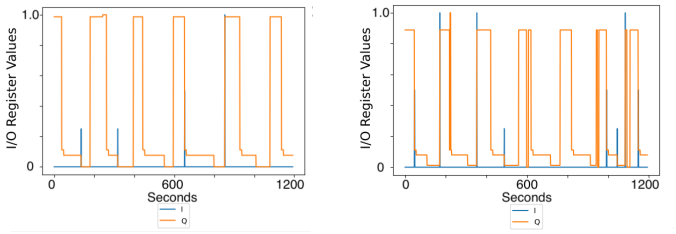
$$Dev_t = n(\tfrac{1}{m}) \sum_i^n |x_i - Baseline\_Mean(x)|,$$
$$\text{where } Dev \in \{network, logs\}$$

V. Evaluation

A. Evaluation Methodology

We develop and assess three scenarios, each of which is generalised and launched across two different PLC vendors. All of our data sets for the diagnosis framework were generated through the Glasgow University Liquid Purification (GULP) testbed. Although GULP was designed to simulate stages of the water treatment process, it is important to note that the data collected for our methodology and the devices we use can be generalised to other ICS architectures and wider OT environments. The distribution of PLC I/O values during our evaluation scenarios is depicted in Fig. 3. Considering normal operation, there is a consistent flow of states, with a repeated pattern seen as the physical process completes its naturally deterministic cycle. However, during our scenarios there is significantly more variance in the captured states shown in Fig. 3b, which is what the diagnosis framework attempts to identify and further classify.

1) Scenario 1: Cyber Intrusion: Fig. 4 illustrates our threat model and the experiment setup used in our evaluation. The adversary of our threat model has escalated remote access through the engineering workstation as this has been a common entry point of attack in previous ICS cyber incidents [15]. It is assumed that the attacker can perform some initial reconnaissance to identify the manufacturer and the PLC model



(a) Normal Distribution of Dynamic I/O States

(b) Abnormal Distribution of Dynamic I/O States

Fig. 3: Normal and Anomalous Distribution of Dynamic I/O States

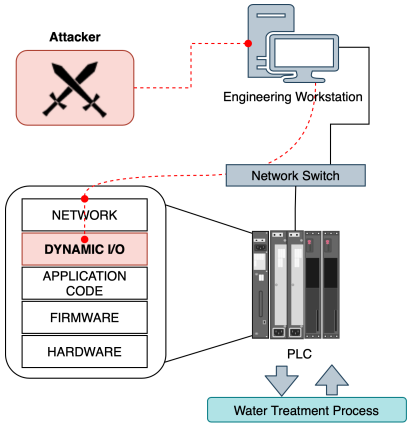


Fig. 4: PLC Threat Model targeting Dynamic Memory (*Red dotted line illustrates attacker's path*)

under attack. As also reported in past events (e.g., [1], [4]) the adversary in our threat model targets the dynamic memory of the PLC. As dynamic values are not often encrypted, they are easier to manipulate, thus a common attack vector for adversaries. Our implemented attack module initially selects a random subset of generalised PLC dynamic I/O memory areas. A random set of register addresses are subsequently generated for each selected dynamic I/O memory area. Finally, a random value is injected at each register address to execute the attack. Two single-register injections are performed, followed by a multi-register attack to evaluate the detection performance on both subtle and more obvious state fluctuations.

2) Scenario 2: System Fault: In this scenario, a PLC key-switch fault was introduced changing the PLC operating mode from START to STOP with the on-device switch. This was the primary fault adopted for evaluation as if the PLC encounters a system fault, it will often default into STOP mode.

3) Scenario 3: Cyber Intrusion and System Fault: This scenario assesses the model's ability to distinguish between cyber intrusions and system faults. Our attack module is executed twice, performing one single-register and one multi-register attack. A PLC key-switch fault is also introduced.

B. Detection Performance

In order to examine the detection performance of our diagnosis model, we adapt the Isolation Forest (IF) formulation which a typical outlier detection approach [16]. 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. The implementation of all classifiers includes an automated hyper-parameter tuning using a Grid Search approach as presented in table II. We utilise a sample size of 2000 normal data points to train the novelty detection classifiers, and each test scenario comprised 400 samples.

TABLE II: Classifier Hyper-Parameters from Grid Search

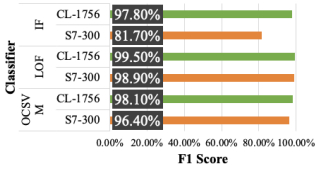
Classifier	Parameters
OCSVM	**Kernel**: RBF, **nu**: 0.001, **Gamma**: Scale
LOF	**Novelty**: True, **Algorithm**: Auto
Isolation Forest	**Estimators**: 100, **Max Samples**: Auto, **Contamination**: 0.1

To evaluate our anomaly identification method, we use a confusion matrix to generate *precision* and *recall* metrics for each classifier, which are presented as F1 scores in Fig. 5. The F1 score is the harmonic mean of both the precision and recall values. Across all three classifiers, LOF performed the best on both PLCs with an average F1 score of 99.3% for the S7-300, and 99.4% for the CL-1756 models. The second best performing classifier was OCSVM, with an average F1 score of 96.8% for the S7-300, and 97.0% CL-1756 PLC models. Conversely, the Isolated Forest classifier did not perform as well, resulting in mean F1 scores of 89.0% for the S7-300 and 96.7% for the CL-1756. One reason for this drop in score is because when detecting outliers, data points with low frequency of occurrence are identified as abnormal. In comparison, these have already been identified as normal points when training the novelty 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.

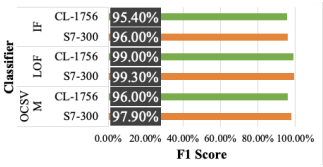
It is evident that both OCSVM and LOF are the more effective classifiers used for the detection model when identifying anomalous behaviour. Given that a CPS can contain many sporadic system states, it follows that a classification method which can cope with this variance is the most suited. Hence, this strengthens the case for using novelty detection machine learning approaches for PLC-specific anomalies.

C. Diagnosis Performance

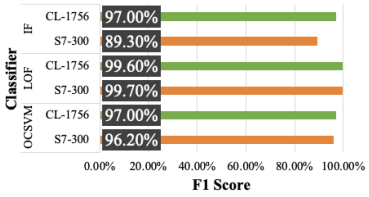
The second stage in the proposed model aimed to classify the identified anomalies in stage one as *cyber intrusion* or *system fault* by using deviations in network and log metrics. Fig 6 illustrates the deviations in PLC write acknowledgements packets (green line), device logs (purple line), and diagnosed anomalies for the S7-300 and CL-1756 PLCs. These graphs illustrate that an increase in network traffic write acknowledgements greatly correlates with the diagnosis of a cyber intrusion anomaly (red square), which is understandable as acknowledgments would be responses to manipulations of data on the PLC. Conversely, deviations in the amount of PLC device logs show significant increase at the time where



(a) F1 Scores for Scenario 1 Anomaly Identification

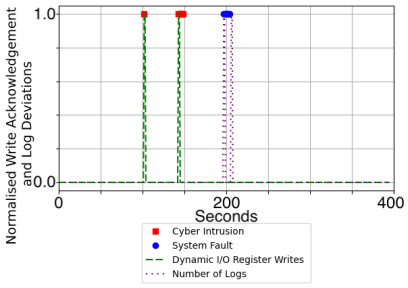


(b) F1 Scores for Scenario 2 Anomaly Identification

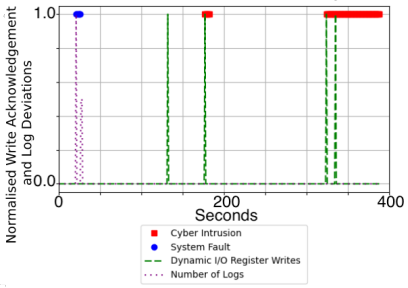


(c) F1 Scores for Scenario 3 Anomaly Identification

Fig. 5: PLC Detection F1 Scores for Classifiers under each Scenario



(a) CL-1756 Writes and Logs



(b) S7-300 Writes and Logs

Fig. 6: Incident Classification

an incident is classified as a system fault (blue circle). This promotes the use of PLC logs for identifying faults rather than intrusions, and key to distinguishing between the two.

Fig 6b shows a deviation in PLC write acknowledgments at around 130 seconds. At this point, the PLC was forced into a legitimate physical process state by the attack module and not an anomalous state. Consequently, By solely using dynamic I/O values for anomaly detection, this type of attack was not identified. The deviation at around 180 seconds in Fig 6b demonstrates that when we include write requests as a feature in the initial anomaly detection, the detection module was able to identify and subsequently classify this attack vector. In general, attacks of this type emphasise an inherit need for synthesised data sources for comprehensive detection.

VI. Conclusion

There are challenges in distinguishing whether CPS anomalous behaviour results from a cyber intrusion or a system fault as both types of event can produce similar effects in the physical process. This paper introduces a novel 2-stage PLC anomaly detection and diagnosis model. We generate a synthesised set of PLC run-time data that comprises dynamic I/O register states, outbound network packets, and PLC device logs. PLC dynamic I/O register states were used to perform anomaly detection through a novelty detection machine learning approach, achieving an average F1 score of 99.35%. Once an anomaly has been detected, our model calculates a weighted deviation in network and device log metrics to infer a classification of cyber intrusion or system fault. We identified that network metrics, specifically PLC write acknowledgement packets, demonstrate a likely cyber intrusion, whereas an increase in PLC device logs suggests a likely system fault. In general, the presented framework establishes strong foundations for the development of vendor-independent intrusion detection components for CPS.

References

[1] 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.

[2] 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.

[3] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious scada communications," in *2013 12th International Conference on Machine Learning and Applications*, vol. 2, 2013.

[4] 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.

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] P. Biondi and the Scapy community. (2021) Scapy: Packet crafting.

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

[13] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000.

[14] 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.

[15] B. Johnson, D. Caban, M. Krotofil, D. Scali, N. Brubaker, and C. Glyer, "Attackers deploy new ics attack framework "triton" and cause operational disruption to critical infrastructure," Tech. Rep., 2017.

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