Cross-domain Alert Correlation methodology for Industrial Control Systems

Oualid Koucham, Stéphane Mocanu, Guillaume Hiet, Jean-Marc Thiriet, Frédéric Majorczyk

Cross-domain Alert Correlation methodology for Industrial Control Systems

Oualid Koucham[a], Stéphane Mocanu[b,*], Guillaume Hiet[c], Jean-Marc Thiriet[d], Frédéric Majorczyk[e]

[a] *GIPSA-Lab, Univ. Grenoble Alpes*
[b] *LIG, Univ. Grenoble Alpes, CNRS, Inria, Grenoble-INP*
[c] *CentraleSupélec, Inria, CNRS, IRISA*
[d] *GIPSA-Lab, Univ. Grenoble Alpes*
[e] *DGA, Inria*

Abstract

In this paper we develop an alert correlation framework specifically tailored for Industrial Control Systems (ICSs). Alert correlation is a set of techniques used to process alerts raised by various intrusion detection systems in order to a eliminate redundant alerts, reduce the number of false alerts, and reconstruct attack scenarios. In ICSs the presence of a physical process and the associated specific threats has led to the heterogeneity of alerts due to the development of multi-domain detection techniques. Such that, some detection approaches rely solely on observations at the level of the cyber domain while other approaches will monitor the physical process. The two approaches are complementary but the information carried by the two types of alerts are different. In this work, we combine the alerts from physical domain intrusion detection with more classical cyber-domain intrusion detection alerts. We develop an alert correlation approach using an alert enrichment that allows mapping physical domain alerts into the cyber domain. We also propose a specific alert selection for correlation that adapts to the state of the physical process by dynamically adjusting the

*Corresponding author
Email addresses: okoucham@cisco.com (Oualid Koucham), stephane.mocanu@grenoble-inp.fr (Stéphane Mocanu), guillaume.hiet@centralesupelec.fr (Guillaume Hiet), jean-marc.thiriet@univ-grenoble-alpes.fr (Jean-Marc Thiriet), frederic.majorczyk@supelec.fr (Frédéric Majorczyk)

size of the selected alert window. We publicly released all the datasets generated and used in our results.
Keywords: Alert Correlation, Intrusion Detection, Alert Enrichment, Industrial Control Systems, Runtime Verification

1. Introduction

In this paper, we propose an alert correlation approach to link alerts from multiple intrusion detection systems in the context of industrial control systems (ICS).

Cybersecurity of Industrial Control Systems (ICS) had become an important topic in academic research after the Stuxnet incident. One of the key aspects which distinguish an ICS from classical IT systems is the presence of a physical process. This has lead to the apparition of a new class of novel threats which target the physical process and many ICS-oriented intrusion detection approaches have been explored in the research community [1, 2, 3, 4]. Since control devices have limited resources and operate under real time constraints [5], most approaches focus on network-based intrusion detection systems (NIDS) over host-based intrusion detection systems (HIDS).

In this paper we are interested in the alert correlation between IDSs interpreting information in two different domains : cyber-domain (i.e. network activity) and physical-domain (i.e. process variables monitoring). Process-aware IDSs take into account the physical process that the ICS controls. They decide if the value applied to a process variable may put the physical process in a critical state. Cyber-domain IDSs analyze the network activity at a higher level. They monitor data such as IP addresses, TCP/UDP ports, network flows or the syntax of the network protocols.

However, understanding alerts in the context of ICS remains a challenge for several reasons. On the one hand, process-aware IDSs cannot identify the source of the attacks, hence the need for cyber-domain IDSs. On the other hand, cyber-domain IDSs do not provide information about the effect of the

attacks on the physical process. Since attacks targeting the physical process can be detected in the physical and in the cyber domains, we need to correlate alerts from both domains to better understand the attack scenarios.

Alert correlation [6, 7, 8] is a set of techniques used to eliminate redundant alerts, reduce the number of false alerts, and reconstruct attack scenarios. So far, there have been few attempts at developing alert correlation in ICSs. In this work, we tackle two specificities of ICSs that challenge traditional correlation approaches : (i) the need to correlate alerts spanning both the cyber and the physical domain, (ii) the alert selection policy for online correlation in the presence of a time sensitive physical process.

1.1. Heterogeneity of multi-domain alerts.

Classical alert correlation [8] includes a *normalization* and a *pre-processing* stages which unify the attributes' values between alerts from different IDSs. However, in most of the works it is assumed that these stages were already performed or their extent is limited to filling missing values such as the time, the source or the type of attacks [8]. In an ICS, these pre-processing stages are more complex due to the heterogeneity of the alerts received by the correlator from both the physical and the cyber domains. In order to pre-process alerts in ICS, the correlation system needs to include information about the devices laying at the boundary of the physical and cyber domains, namely the controllers. Given the diversity of the network interfaces and ICS protocols supported by the controllers, *an attacker can use different vectors at the network level to achieve the same effect over the physical process.* In a conceptual view one can say that the available network protocols are mapped to the same effect on the physical process. Using an analogy with the *procedural abstraction* [9] we consider the attack effect on the physical process as an *abstraction* of the network level attack vector and, reciprocally, the network level attacks are *concretions* (or *realisations*) of the physical level attacks. By relying on the notion of the abstraction operator [10], alerts from the physical domain can be rewritten in terms of cyber attributes and correlated with cyber domain alerts. This

approach is the cornerstone of our alert normalization procedure and will be discussed in detail in section 4.

1.2. Alert selection policy.

For each new alert, the correlator needs to decide which previously received alerts will be tested for correlation. This choice is called an alert selection policy. A naive policy will memorize and test all received alerts. However, due to resource limits, most alert correlation approaches set a sliding time window with a fixed size. Each new alert is tested with all or a subset of the previous alerts in the current window. For instance, the authors in [8] heuristically set a window size of 2s. In ICS where the evolution of the physical process is hard to predict, deciding on a single optimal window size is problematic. We thus develop alert selection policies that adapt to the state of the physical process.

1.3. Contributions and paper structure

In summary, we make the following contributions :

• We propose an approach to link manifestations from both the cyber and the physical domains in an ICS by taking into account the configuration of controllers and the specifications of ICS protocols.

• We introduce an alert selection policy which captures long-term manifestations of attacks by adjusting the alert window to the runtime context of the physical process.

• We evaluate our approach using an experimental testbed in a hardware-in-the-loop setting under both attacks and legitimate operator manipulations.

The paper is organized as follows. Section 2 provides the necessary background information about industrial control systems and introduces our threat model. Section 4 presents our correlation approach. Section 5 describes our evaluation setup and implementation. Section 6 provides a discussion and analysis of the results. Section 3 reviews related work on alert correlation in ICS. Finally, Section 7 concludes the paper.

2. Background

ICSs are cyber-physical systems that control a physical process in order to achieve some industrial objectives [11]. They can be divided in different levels. At the field level, sensors perform measurements of physical quantities (temperature, pressure, level, etc.) while actuators such as pumps or valves act on the physical process. At the control level, programmable logic controllers (PLCs) regulate the physical process through an execution cycle. This cycle consists in reading new values from the sensors, transitioning to new states, executing some control logics, and sending commands to actuators. At the supervisory level, control servers collect data and send commands to the PLCs. Human-machine interfaces (HMIs) provide the means for operators to monitor and act on the physical process. Specific workstations allow engineers to change the PLC's control logics. Finally, historians store data on the evolution of the physical process to enable trend analysis and reporting.

2.1. Sequential Control Systems

In this paper, we focus on *sequential control systems* where the control logics follow a sequence of discrete steps linked by transitions with boolean guards in terms of sensor states and internal variables. Each step is associated with actions such as the manipulation of an actuator (a valve or a motor for instance). Control logics in sequential control systems are best represented using sequential function charts (SFC), a graphical language defined in the IEC 61131-3 standard [12]. In a typical SFC-based program, the fundamental sub-processes or stages in the execution of the process control are represented by linear sequences of steps and transitions. We call such sequences *activities*. These activities can be further combined in parallel or branching configurations to construct more complex logics. At any moment during the execution of a sequential control system, the set of steps reached by the control logics constitutes the *runtime context* of the physical process.

2.2. Communication protocols in ICS

Communications between the various components in an ICS are performed through a variety of open and constructor-specific protocols. We illustrate our approach in this paper using the Modbus/TCP protocol. This application layer protocol operates in a client/server setting. It is typically used for inter-PLC and supervisor/HMI to PLC communications. Two Modbus fields are of interest for our application as they determine the effect of a message on the process variables: the function code and the data fields. The function code specifies the nature of the request such as the reading or the writing of process variables. The data field depends on the function code; in case of a write request, it specifies the start address of the write operation in the PLC's memory (i.e the address of the first variable to write), the quantity of the elements to write relative to the start address, and the new written values.

2.3. Threat Model

In most real-world attack incidents [13, 14, 15, 16], the attack originates from the supervisory level since it is easier to be contaminated either through infected USB flash drives [13], spear-phishing attacks [14, 15], accessing malicious websites [14] or downloading compromised software [14]. The attack then propagates to the lower levels of the ICS. The direct modification of the traffic from sensors or towards actuators at the field level requires physical access to the ICS components. Such physical attacks fall under sabotage rather than computer attacks, and are generally solved using organisational measures and physical security. We, thus, focus on ICS attacks with both cyber and physical manifestations. The goal of the attacker is to put the physical process in a critical state through a series of commands sent to the PLCs using one or more ICS protocols as in the case of the CrashOverride malware [16]. The attacks are carried through supervisors and through ICS workstations that can access the PLCs. We assume that the attacker has knowledge about the network protocols supported by each PLC and about the memory mapping of process variables among the PLCs, possibly from a preliminary identification step [16].

With respect to MITRE ATT&CK for ICS Matrix [17] we consider threats that will "Impair the Process Control" by forcing some outputs using "Standard Application Layer Protocol" by "Manipulation of Control" with a large spectrum of "Impacts" including "Loss of Availability/Control/Protection/Safety" occurring in "Damage of Property".

2.4. Illustrative example

We present a sub-process (Figure 1) that will be used in the following sections as a running example to illustrate our approach. This sub-process represents a single processing stage in a multi-stage chemical plant. Product incoming from the physical process is first put in tank *TK2* through valve *VTP1*. Using the cart *CH1*, a quantity of product *P5* is also added to the content of tank *TK2*. Then, the products are mixed using motor *M2* for a fixed amount of time. Finally, tank *TK2* is emptied using valve *VT2*.

These control operations are performed by a PLC and correspond, in terms of control logics, to a single activity (a linear sequence of step-transitions). In the supervisory domain, a supervisor HMI along with an OPC server allow the operators to perform manual interventions on the process. In our example, the supervisor HMI can communicate with the PLC using either Modbus or SOAP web-services through the PLC's internal web server. The OPC server only uses Modbus. The engineering workstation is solely used to reprogram the PLC and is never used for any process interventions.

Two types of IDS are deployed : (i) a process-aware IDS which surveys the state of the sensors/actuators through cyclical observation of the signals and reports any process specification violation such as the opening of a valve in the wrong step, and (ii) a protocol-specific payload-based IDS which reports any unknown attempt to manipulate an actuator on the PLC. For instance, unknown attempts might be due to a Modbus write command sent from an unauthorized host (engineering workstation), or by using a previously unseen protocol (for instance, a SOAP request).

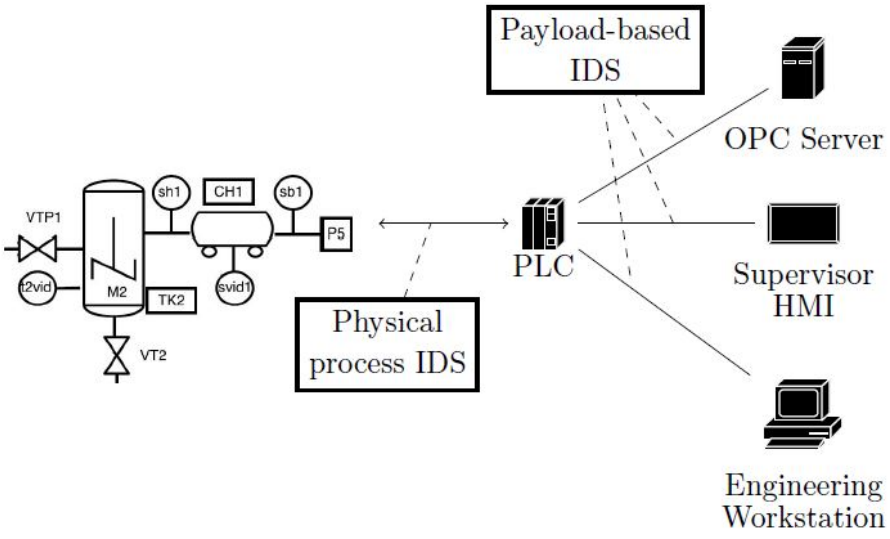

Figure 1: Example of a subprocess

A possible process-aware attack on this sub-process is an attempt to overflow the tank *TK2* through the malicious manipulation of valve *VTP1*. In a typical run, this valve is only manipulated at the beginning of the activity until tank *TK2* is filled. To overflow *TK2*, an attacker can keep valve *VTP1* open throughout the activity. If the attacker sends the command to open valve *VTP1* from an unauthorized host such as the engineering workstation, then both the process-aware IDS and the payload-based IDS will raise alerts. Our goal is to group such alerts so that they can be displayed together to a security operator. The alert from the payload-based IDS would allow the operator to incriminate the engineering workstation and further investigate the source of the attack (by examining the workstation's logs for instance). However, the payload-based IDS alert can also be a false positive (a message not observed during learning). Similarly, the opening of valve *VTP1* might be a legitimate action or an action with low incidence on the physical process. Using the alert raised by the physical process IDS, the operator can recognize that it is indeed an attack, and that given the current context of the physical process, opening valve *VTP1* is critical. Thus, the association of information from both the payload-based IDS

and the physical process IDS allows the operator to gain precious time in the characterization of the alerts.

3. Related Work

Our work focuses on correlating alerts issued by heterogeneous IDSs monitoring an ICS. Thus, we do not contribute to the detection phase, and we do not evaluate the relative detection performances of the IDS. However, our correlation approach uses data from various IDSs, relying on existing detection algorithms. We develop a specific testbed to evaluate our correlation approach, which allows us to monitor both the network traffic and the physical process.

This section first presents the related work on alert correlation, which are the closest works. Then we introduce the intrusion detection approaches that can benefit from our correlation approach. Finally, we briefly mention the works that proposed testbeds and datasets to evaluate intrusion detection and correlation approaches in the ICS domain.

3.1. Alert correlation

Overall, the main objectives of alert correlation approaches consist in :

- reducing the number of alerts by removing false positives and redundancies,
- grouping low-level alerts by reconstructing attack scenarios that afford operators a higher-level view,
- giving scores of criticality to alert groups.

In this article, we are interested in the first objective.

There is a significant amount of literature on alert correlations in traditional IT systems, and several works proposed different classification models of correlation approaches [18, 8, 19]. We can identify the following four-step general architecture:

1. preprocessing, i.e., alert normalization and enrichment;

2. verification;

3. aggregation;

4. impact and priority analysis.

Raw alerts raised by the IDS often require some preprocessing to be analyzed and compared in the following correlation steps. For instance, alerts emitted by different IDS often use different formats and conventions. Thus, a **normalization** step is required to unify the syntax (i.e., the structure of the attributes) [20, 21] and semantics (i.e., the meaning of the attributes) [20, 22, 21] of the alerts. For instance, the works in [21, 20] rely on the IDMEF format [1] which defines a unified format that enables syntactic normalization.

In general, alert correlation approaches implicitly consider the normalization step [6]. However, this step is crucial for the correlation process [23]. In ICS, normalization is even more challenging because alerts from the cyber and the physical domains are characterized by radically different attributes. Additionally, these alerts from different domains can carry complementary information, as in the case of a process-oriented attack originating from the cyber domain. This heterogeneity limits the applicability of existing normalization approaches, which cater either for syntactic variations in the representation of information on the same attribute or the representation of the same information in different attributes.

The previous remarks on the heterogeneity and complementarity of the information carried by alerts coming from different domains of an ICS suggest that our main objective is to enrich the information gathered from one domain with information from the other domain. In traditional alert correlation approaches [7, 24, 25], **alert enrichment** objective is to add some missing contextual information. Enrichment approaches often rely on knowledge bases [26]

[1] `http://tools.ietf.org/html/rfc4765`

which might include information such as the system's topology [24] and assets [7]. In the same vein, the approach in [25] uses honeypot databases for contextual information on malware propagation activity or the profile of web servers in order to enrich IDS alerts.

However, these classical enrichment approaches suffer from the same bane as classical normalization approaches. Such enrichment solutions are possible due to their confinement to a single domain where attributes are mainly homogeneous. For instance, while the approach in [25] aims at enriching local alerts raised by IDS with global threat information available in honeypot databases, all information belongs to the cyber domain and is represented using similar attributes (IP addresses, detection time, classification of attacks). In contrast, information carried by physical domain alerts is represented in terms of widely different attributes (actuator/sensor events and states) in comparison to information within cyber domain alerts.

Alert **verification** consists in verifying and discarding alerts that are not pertinent for the system configuration [27]. For instance, alerts might refer to attacks exploiting vulnerabilities not present in the system. Following the classification in [27], verification approaches can be classified as either active or passive. Active alert verification [27] dynamically looks for information to determine the pertinence of an alert. On the other hand, passive alert verification [24] relies on a priori information gathered about the system, which is possibly stored in formal knowledge bases such as M4D4 [26]. So far, verification approaches geared towards ICS remain rare. As a preliminary step, the development of knowledge bases for ICS that might support passive alert verification has been studied in the literature [28, 29].

Aggregation consists in reducing the number of alerts by grouping alerts together into *meta-alerts*. For instance, alerts referring to the same attack can be grouped into meta-alerts. The approaches which look for similarity between alerts [30, 6] often rely on a direct comparison between common attributes using different metrics. The aggregation of alerts can serve as a preliminary stage to identifying attack scenarios through the association of meta-alerts with

elementary attack steps. This association can be based on a specification of the expected attacks using, for instance, correlation rules [31], attack graphs [32] or trees [33]. Attack reconstruction can also be performed by reasoning on pre-conditions and post-conditions of elementary attack steps [34]. The above approaches assume that a subset of common attributes characterizes alerts to define their similarity metrics. As argued previously, this means that a direct application of these approaches in the context of ICS, where alerts coming from different domains do not share attributes, is complex without a proper pre-treatment step.

Finally, the **impact and priority analysis** rank alert depending on criteria such as the potential impact of attacks or the confidence in the IDS's verdicts. For instance, Briesemeister et al. [35] evaluate the priority of alerts in ICS depending on the nature of the targeted components and the ICS zone where the attacks are detected.

In ICS, work on alert correlation remains scarce. Many works focus on the last steps of correlation, i.e., identifying attack scenarios and prioritizing alerts. For example, Briesemeister et al. proposed an approach to detect, correlate and visualize multistep attacks [35]. Lanoe et al. [36] also proposed a rule-based approach for attack-scenario reconstruction. Bayesian classification can also be used to identify predefined scenario attacks in the smart-grid domain [37].

Those approaches complement our work since they focus on reconstructing multistep attacks. In contrast, we focus more on the first stage of the correlation process, i.e., correlating single-step manifestations of attacks spanning the cyber and physical domains. Normalizing and aggregating heterogeneous alerts is a crucial step before trying to identify more complex attack scenarios. Moreover, our approach aggregates information from both the process and cyber domains, which is crucial to better classify and prioritize those alerts in further correlation steps.

Other approaches are limited in the types of alerts they can correlate [38, 39] or the information taken into account during the correlation process [40]. Feng et al. combine different anomaly-based approaches to detect intrusions cite-

Feng2017. However, they only combine physical-domain approaches. In [39], an alert correlation for ICSs is proposed, but limited only to Windows logs and network pattern-based alerts. In [40], the authors develop a statistical anomaly alert classification approach based on a hidden Markov model (HMM). Alerts from different anomaly-based IDSs are reduced to a tuple of Boolean values, where each position in the tuple is associated with an IDS. The HMM estimates whether a tuple corresponds to an attack, a fault, or a normal behavior. In contrast, our fusion approach takes full advantage of the information available in each alert's attribute using a pre-processing phase.

3.2. Intrusion detection

Our correlation approach uses data from various intrusion detection systems (IDS), relying on existing detection algorithms. We can identify two major detection methods: anomaly-based and misuse-based. Anomaly-based approaches suppose that an intrusion can be detected by observing deviations from the normal behavior of the monitored system. Misuse-based approaches rely on a knowledge base of abnormal behavior, usually represented by attack signatures. In general, misuse-base IDS use pattern matching algorithms to recognize suspicious actions.

Compared to misuse-based intrusion detection, anomaly-based approaches have been the focus of research efforts on intrusion detection in ICS. Indeed, there is a widespread belief that ICS exhibit relatively stable behaviors due to their fixed topologies and regular communication patterns [5]. This stable behavior limits the numbers of false positives due to legitimate anomalies, i.e., legitimate behaviors that were not present in the reference model. Moreover, anomaly-based approaches can detect novel attacks. Misuse-based intrusion detection efforts in the ICS domain have consisted mainly of building signature databases such as those provided by Digital Bond [2].

The literature contains several examples of ICS-oriented IDS taxonomies [41,

[2] `https://github.com/digitalbond/Quickdraw-Snort`

42, 43, 40, 44]. By synthesizing these different taxonomies, we propose to classify ICS-specific IDS approaches based on the degree of knowledge that the IDS has of the system's interaction with the physical process. Thus, we distinguish between *cyber-domain* intrusion detection approaches that only focus on the cyber aspect of the ICS and *physical-domain* approaches that focus on the physical process.

3.2.1. Physical-domain approaches

Process-aware IDSs [45, 46, 47, 48, 49, 50] cover the physical domain of the ICS. Given a model of the correct behavior of the physical process, a process-aware IDS monitors the communications between sensors/actuators and Programmable Logic Controllers (PLCs) while reporting anomalous deviations in the evolution of process variables. They decide if the value applied to a process variable may put the physical process in a critical state. For instance, an attacker might issue commands to overflow a tank by opening a valve when the tank is already full.

Hadziosmanovic et al. associate a model for each variable depending on its type in order to predict its next value [45]: an autoregressive model with maximal and minimal bounds for continuous variables and a set of observed values for discrete variables and constants. The IDS then raises an alert when variables manifest abnormal behavior. Other works extract forbidden states [49] or legitimate behaviors [50] from manual specifications of the physical process. Such specifications are often expressed in terms of interval values on measure vectors. The IDS raises an alert when the current state reaches or approaches a forbidden state.

In general, such approaches express constraints as simple conjunctions of assignments to PLC registers with no notion of temporal order. Thus, the model does not allow expressing rules over events, i.e., changes in the state of the actuators or sensors, or over temporal behaviors involving states and events. For instance, one cannot express a rule that raises an alert if a valve is opened more than 10 seconds after shutting down a motor. Expressing such rules requires

more expressive models based on trajectories instead of states. Moreover, those approaches assume that the set of critical states can be manually determined in advance. While such an assumption can be valid for small systems, manually determining critical states for large systems can be daunting, with the risk of missing some critical states. As a result, some approaches explored more expressive formalisms, such as temporal logics [48, 47] or state-aware models [51]. For example, Koucham et al. infer temporal safety properties over the states of sensors and actuators from legitimate traces [47]. Then, their IDS detects any violation of such process specifications.

More recently, some works proposed to infer control invariants, i.e., relations between sensor readings and the concomitantly triggered PLC commands [52, 53]. Similarly, Aoudi et al. proposed an anomaly-based technique that is capable of detecting stealthy attacks by monitoring time series of sensor measurements for structural changes in their behavior [54]. Those approaches rely on a sufficiently expressive formalism to cover the sequential dynamics of an ICS. They automatically build the detection models through an inference phase to alleviate the operators from the burden of manually specifying the models.

In contrast to the approaches that rely solely on cyber domain observations to detect attacks targeting the physical process [3], process-oriented approaches directly monitor the evolution of process variables. They are better suited to detect attacks targeting the physical process, such as Stuxnet [13] or CrashOverride [55]. However, they cannot identify the cyber artifacts related to the attack (e.g., the IP address of the station the attacker used to reconfigure the physical process).

3.2.2. Cyber-domain approaches

Cyber-domain IDSs analyze the network activity at a higher level. They monitor data such as IP addresses, TCP/UDP ports, network flows or the syntax of the network protocols.

Network monitoring approaches used in ICSs usually detect suspicious network activities by analyzing network flows [1, 56]. The IDS first learns a safelist

of flows from legitimate network traffic, and then alerts any flow absent from the safelist during detection. However, dynamic port allocation can lead to false positives [1]. Moreover, an attack can correspond to a legitimate flow, leading to false negatives if the attacker exploits devices and protocols that operators also use for legitimate configuration of the physical process. Finally, an unknown flow does not provide enough information for a security operator to understand its impact on the ICS physical process.

Telemetry-oriented approaches [57, 58] focus on building a base profile of network exchanges using statistical measures or classification models. Such approaches target a limited threat model, similarly to the flow-based approaches. They focus on attacks that lead to a significant observable deviation in the flow statistics, such as denial of service [57, 58] or reconnaissance attacks [58]. Thus, a telemetric approach would fail against more subtle and advanced attacks, such as process-oriented attacks that only use a few commands to put the physical process in a critical state.

Payload-based IDS [2, 4, 59, 60, 3, 61] are protocol-specific anomaly-based IDS that detect abnormal ICS protocol messages or sequences of messages.

Some of these approaches [2, 4] focus on the field values of each message. For example, Düssel et al. create a reference model of each protocol message using an n-gram approach [4]. Yuksel et al. build a more precise base profile by recording the distribution of the set of values taken by pre-selected message fields (see Section 2 for a discussion of Modbus fields) [2]. Then, any message whose field value differs significantly from the base profile is flagged as anomalous during the detection phase. Contrary to flow-based and telemetry-based approaches, those payload-based approaches can detect attacks using legitimate flows, e.g., a configuration command issued by a legitimate workstation. However, they can only detect significant deviations in the distribution of message values (e.g., the occurrence of infrequent messages). They cannot detect attacks relying on frequent messages. In such attacks, it is not the value of the message that identifies the attack, but the fact that this message arrives in a particular context. Detecting those attacks implies to reason about the state of the system

or the sequence of messages.

Some approaches also take into account dependencies between messages. Cheung et al. focus on the Modbus protocol and explore an anomaly-based approach that constrains the values taken by some fields (e.g., function code or protocol identifier) in a Modbus message [56]. Similarly, Lin et al. develop an analyzer for the DNP3 protocol [62] which checks dependencies: (i) within a single DNP3 message such as field lengths, and (ii) between different DNP3 messages such as the occurrence of a response after each request. While the dependencies on the protocol grammar in these works are limited to simple request-response constraints, some approaches go farther by inferring an automaton model of the message exchanges[59, 60, 3, 61]. For example, the detection method in [3] builds models of the network exchanges between a supervisor and a controller to detect unknown command sequences, which might reveal attacks on the physical process. However, to keep the models tractable, these approaches apply abstractions to the network messages, such as ignoring sensor values. This simplification leads to inaccurate models that fail to capture the state of the physical process and may generate false positives and missed attacks.

A global issue of protocol-based approaches is the lack of information provided by the alerts concerning the behavior at the physical process level. Since no context is given about the state of the physical process when the variable is accessed, the process operator faces difficulties in identifying whether the alert corresponds to a false or true positive. However, these approaches are essential to identify cyber-domain artifacts. Such artifacts could then be used to better analyze and respond to attacks. For example, the IP address or domain name that issues the malicious commands can be correlated with a previous malware infection and used to block any further attacks.

3.2.3. Cross-domain approaches

Some works [63, 40] cover multiple aspects of the above taxonomy. Such a broad coverage is motivated by the need to detect sophisticated attacks, identify

accidental deviations, and reduce false positives. While the above approaches explore the joint use of different intrusion detection approaches, the treatment of alerts remains simplistic, leading to increased awareness of the need for alert correlation. For example, the solution proposed by Zhou et al. includes knowledge of the physical process in terms of critical states [40]. However, the reduction performed on the alerts discards information that would otherwise help the operator understand the suspicious manifestations (e.g., impacted actuators or IP addresses). Thus, an important and still unresolved issue facing approaches covering multiple aspects is the need to associate suspicious manifestations coming from both the cyber and the physical domains despite their disparity in terms of attributes (actuator/sensor states and events in the physical domain, protocol-based attributes in the cyber domain).

Our correlation approach receives inputs from separate IDSs, each one specific to a domain, and the resulting meta-alerts summarize the information from both domains. Our experimental evaluation used the following IDSs found among the literature concerning intrusion detection: network flow IDS [1, 56], payload-based IDS [2, 4], and process-aware IDS [47].

3.3. Testbeds and datasets

Conti et al. provide a comprehensive survey on the testbeds, and related datasets, developed to test IDS performances on ICS environments. [64]. However, we could not have directly exploited those datasets for our work because we would have needed access to the control logic and the initial state of the logic in the captures to perform the activity tracking required by the process-aware IDS [47].

4. Approach

We first present the workflow of our correlation approach and then discuss each correlation stage using an illustrative example.

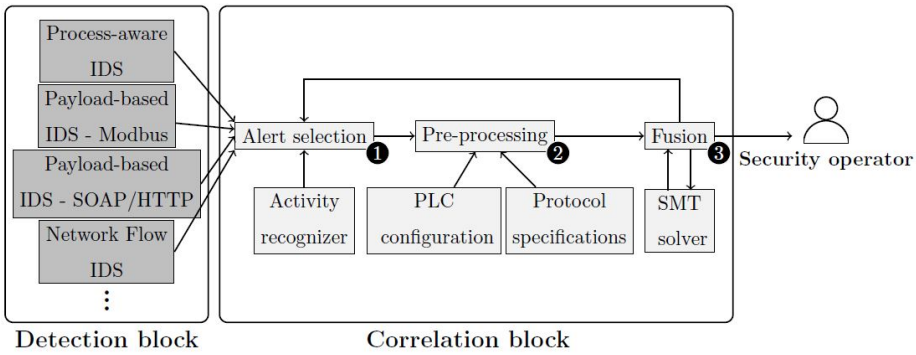

Figure 2: Workflow of the correlation approach

4.1. System overview

Figure 2 shows the workflow of our proposed online correlation approach. At a high level, we distinguish between two blocks : a detection block and a correlation block. The detection block contains a cyber-domain IDS and a process-aware IDS. At runtime, these IDSs detect suspicious manifestations and send alerts to the correlation block, which is the focus of this paper.

The heart of the correlation block consists of a loop in three stages.

In the *alert selection* stage (❶), the received alerts are inserted into activity-specific alert windows. Alert windows limit the number of alerts considered for correlation, which is particularly useful in online correlation. In our approach, these alert windows are instantiated and terminated depending on the evolution of the activities at runtime. The *activity recognizer* is responsible for notifying the correlator about the runtime context of the physical process (i.e, the active steps within the activities). It tracks the current active steps using the knowledge of the SFCs implemented in the PLCs and the evolution of the actuator and the sensor states. The activity recognizer is provided with the SFCs executed by the PLCs and does not need to connect to the PLCs in order to obtain them. Similarily, it can directly monitor the evolution of the actuator and the sensor signals without interacting with the PLCs. Thus, through the activity recognizer, the correlator can be notified about the runtime context of the physical process independently of the PLCs.

In the *pre-processing* stage (❷), the correlator transforms the alerts coming from different IDSs in first-order logic formulae so that they can be compared for correlation. In particular, alerts from the physical domain are mapped to the cyber domain given knowledge such as ICS protocol specifications and the memory layout of process variables.

Finally, the *fusion* stage (❸) tests whether alerts are correlated and keeps track of the correlations. Since alerts and their transformations are expressed as first-order logic formulae, the correlator tests the compatibility of their attributes by querying a satisfiability modulo theory (SMT) solver. Correlated alerts are grouped into meta-alerts which can be presented to the security operator through a SIEM (Security Information and Event Management) software for instance. The meta-alerts contain references to the correlated alerts. Feedback about successful correlations is reported back to the *alert selection policy*. This feedback can be used to select the next alerts to test for correlation (edge from ❸ to ❶).

4.2. Alert selection

The alert selection phase relies on an alert window and some predefined alert selection policies to decide which alerts will be provided to the subsequent stages. As the same attack may trigger alerts from different IDS at different times, an important issue is the choice of the time windows length during which alerts are selected for correlation. Since we perform correlation online, this window length cannot be bounded a priori. For instance, a network flow IDS will raise an alert instantaneously when an unknown flow is detected, while a process-aware IDS may wait for another event to decide the violation of security property. Due to the limitations of the memory resources and the ever-increasing number of old alerts to test against every new alert, our correlator cannot keep all the received alerts indefinitely. Classical online correlation approaches often rely on a time sliding window whose size is heuristically selected in response to this issue. The choice of a single optimal alert window size is difficult since an activity's (i.e., a subprocess) duration can vary over time. If the alert window's size is not

adjusted to the activity, the correlator can miss activity-long manifestations of an attack.

In our example sub-process (Figure 1), maintaining the valve *VTP1* open causes violations to be raised by the physical process IDS throughout the activity, since *VTP1* should only be manipulated at the beginning of this activity, i.e when filling the tank. A sliding time window-based correlator would need to adjust the window's size to match the activity's duration in order to collect all the alerts relative to the attack. However, the transitions in an activity can depend on other parts of the physical process (the influx of product *P5* from another subprocess) as well as on specific time conditions (such as running motor *M2* for a certain amount of time). Moreover, an operator can also manipulate *VT2* to interrupt the flow of output product and thus impact the activity's duration. Consequently, the time spent in each activity is hard to predict.

To solve this issue, instead of setting a fixed alert window size, we adjust the size of the window at runtime by monitoring the activation of steps in an activity (recall from Section 2 that sequential control systems can be represented as combinations of activities). Depending on the number of steps taken into account in the alert window, we introduce three alert selection policies:

- *All alerts.* In this simple policy, all the alerts from the activation of the first step of the activity until the end of the activity are selected for correlation.
- *Alerts in adjacent steps.* This policy takes into account the intermediate steps within an activity. The correlator allocates a slot in the window for each step in the activity. During runtime, the correlator inserts new alerts in the current slot corresponding to the current active step in the activity. Only alerts that occur within the last 2 slots are selected for correlation. The policy rests on the assumption that alerts that occur in adjacent slots are more likely to be correlated than alerts belonging to distant slots.
- *Adaptive alert selection.* Instead of fixing a parameter for the number of relevant neighboring slots, this policy uses the result of the previous correlations to decide on the number of past neighboring slots to add. At first, the new

alert is tested against the alerts in the current slot. Then, correlations are iteratively tested against earlier slots. If a correlation cannot be found within a slot in an iteration, no alerts in earlier slots are tested. This policy allows finding activity-long attacks by linking alerts from adjacent slots until no further correlation is possible.

4.3. Alert pre-processing

The alert pre-processing phase maps alerts from the physical process IDS into cyber alerts. To illustrate the need for such a mapping, consider the attack on the example subprocess in Figure 1. The process monitor ([47]) reports an alert A_{pm} concerning the forbidden opening ($\uparrow$) of valve *VTP1*. The alert A_{pm} is expressed using the following list of attribute key-value pairs:

$$\mathbf{A_{pm}} : [(\textit{component}, \textit{VTP1}), (\textit{event}, \uparrow)], \tag{1}$$

where *component* and *event* are the *attributes* through which the monitor reports the attack manifestation. If the attacker opens valve *VTP1* by sending a Modbus command from an engineering workstation with address H_{ew} to the PLC with address H_{plc}, then the Modbus actuator access monitor could raise an alert A_{am}:

$$\begin{aligned}\mathbf{A_{am}} : [&(\textit{source IP}, H_{ew}), (\textit{destination IP}, H_{plc}), (\textit{unit id}, 0),\\ &(\textit{protocol id}, 0), (\textit{function code}, 5), (\textit{address}, 41), (\textit{data}, 1)]\end{aligned} \tag{2}$$

In this alert, the *function code* has the value 5 which refers to the modification of a single coil (1-bit variable). The value of the *address* attribute corresponds to the mapping of actuator *VTP1* in the PLC's memory. For the purpose of our example, *VTP1* is mapped to address 41 and can be accessed either through Modbus or SOAP/HTTP. Both information are given by the PLC's configuration in terms of supported interfaces and memory mapping of process variables which are specified during development time by an engineer.

We note that there are no common attributes between A_{pm} and A_{am}. Each alert comes from a monitor operating in a different domain, and they consequently report alerts in terms of widely different attributes. It is impossible to

directly compare the alerts without a pre-processing stage. The next observation is that A_{am} is only one possible manifestation of the attack in the cyber domain. To achieve his objective, the attacker can select between different requests that command the opening of *VTP1*. These requests can be detected by the actuator access monitors but will result in alerts that vary in terms of attributes and values. For example, consider the following two possible alerts :

$$\begin{aligned}
\mathbf{A}^{'}_{\mathbf{am}} : & [(source\ IP, H_{ew}), (destination\ IP, H_{plc}), (unit\ id, 0), \\
& (protocol\ id, 0), (function\ code, 15), (start\ address, 40), \\
& (quantity, 3), (values, 010)] \\
\mathbf{A}^{''}_{\mathbf{am}} : & [(source\ IP, H_{ew}), (destination\ IP, H_{plc}), (unit\ id, 0), \\
& (protocol\ id, 0), (function\ code, 15), (start\ address, 41), \\
& (quantity, 2), (values, 10)]
\end{aligned} \tag{3}$$

In this case, the attacker uses *function code* 15 which allows the modification of multiple coils at a time. The *start address*, *quantity* and *values* give respectively the offset, the quantity of coils to be modified, and the values to write. In each of $A^{'}_{am}$ and $A^{''}_{am}$, address 41 (corresponding to *VTP1*) is assigned the value 1. It is clear that, by further varying the *start address*, *quantity* and *values* attributes, the attacker can generate a greater variety of Modbus commands which lead to the opening of valve *VTP1*. The attacker could also have used another protocol such as a SOAP/HTTP request, which further increases the possibilities. Thus, a straightforward transformation of alerts in attribute-value pairs of a generic format like IDWG [65] is intractable due to the amount of possibilities.

In general, we observe that, due to the possibilities through which an operator or an attacker can interact with a PLC to affect any given process variable, the physical domain and the network domain are linked through an *abstraction* relation. In the following we use the *abstraction* concept in the sense of *Knowledge Reformulation and Abstraction* ($\mathcal{KRA}$) [10]. For instance, an actuator event such as the opening of valve *VTP1 abstracts* away from the partic-

ular PLC interface (Modbus or SOAP/HTTP) or protocol variation (Modbus function code 5 or 15) that was used to modify the PLC's memory variable corresponding to *VTP1*.

The $\mathcal{KRA}$ model is a general abstract knowledge representation model not specially designed for computer science (although partly inspired from procedural abstraction and concretion model presented in [9]). Also, the $\mathcal{KRA}$ model primarily aims for a bottom-up use i.e., abstracting concrete observations to generic patterns. Roughly, this is achieved by hiding some attributes of the observation or making the description less detailed. Using such an abstraction process, we can abstract network actuator alerts described by formulas (2) and (3) to the process-level alert described by formula (1). If $\mathcal{O}$ is an abstraction operator, then we can write $A_{am} = \mathcal{O}_E(A'_{am}) = \mathcal{O}_E(A''_{am})$ and $A_{pm} = \mathcal{O}_H(A_{am})$, i.e., network alerts abstracts to Modbus alerts by *attribute equivalence* operator $\mathcal{O}_E$ and Modbus alerts abstracts to process alerts by *attribute hiding* operator $\mathcal{O}_H$. However, abstracting all alerts at the process level is not suitable for correlation purposes since we lose essential information concerning the source of attacks and application protocol used.

For our alert correlation needs, pre-processing has to map process-level alerts to network level alerts. This alert enrichment process corresponds to a de-abstraction or to a concretion process [9].

In the following, we extend the $\mathcal{KRA}$ model with concretion operators that will enrich the alerts. A concretion operator $\mathcal{O}_\Pi^{-1}$ will concrete an abstract alert A_{abs} to an alert that subsumes all the concrete alerts A_{con} that can abstract to A_{abs} given the relation Π. For instance $\mathcal{O}_E^{-1}(A_{am}) = A'_{am} \vee A''_{am}$ i.e. Modbus alert A_{am} concrete to network alert A'_{am} or A''_{am}.

4.4. Alert enrichment and concretion

The enrichment process is depicted in Figure 3. The process takes as input a physical process domain alert, expressed in terms of process domain attributes, and outputs an enriched cyber domain alert expressed in terms of cyber domain attributes. The heart of the enrichment consists of the successive application of

three concretion operators: two attributes equivalence concretion operators (❶ , ❷) and one attributes hiding concretion operator (❸).

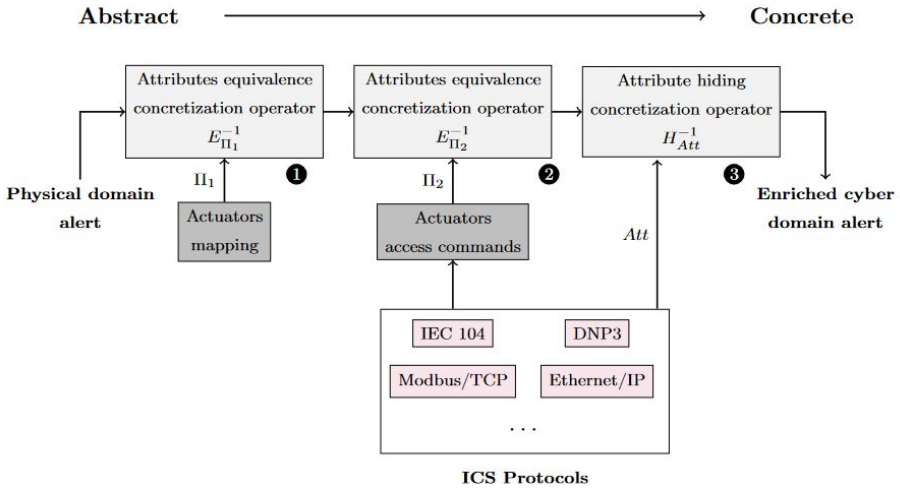

Figure 3: Overview of the alert enrichment process

Each concretion operator uses information from the ICS to perform its mapping. We now discuss each step in detail and we introduce the three concretion operators. In order to illustrate the enrichment process, we use the process-domain alert example described by formula (1).

4.4.1. Step 1: Actuator mapping

The first step in the alert enrichment process is to map each actuator to (i) the set of network addresses of the PLCs which control it, (ii) the protocols which can be used to access the actuator on each PLC identified in (i), and (iii) the address or identifier which is associated with the actuator for each protocol and PLC identified in (ii). Let Π_1 be an application that associates each actuator with information (i), (ii) and (iii) above, i.e. :

$$\{network\_address, protocol, variable\_address\} \xrightarrow{\Pi_1} \{component\}. \quad (4)$$

Then, the **attributes equivalence concretion operator** E_{Π}^{-1} will map a process-domain alert to the sum of all cyber-domain alerts that abstracts to the process-domain alert by network attributes hiding.

In the case of example 2.4, the actuator $VTP1$ can be accessed via two different network interfaces with network addresses $H1_{PLC}$ and $H2_{PLC}$ using Modbus/TCP, respectively SOAP/HTTP protocols. Thus, we can enrich the process-domain alert from formula (1) as follow:

$$E_{\Pi_1}^{-1}(A_{pm}) : [(event, \uparrow) \wedge (((dest_ip, H1_{PLC}) \wedge (protocol, Mbus) \wedge (addr, M41)) \\ \vee((dest_ip, H2_{PLC}) \wedge (protocol, SOAP) \wedge (var, bit41)))]$$

The mapping Π_1 can be obtained from implementation specifications, PLC programs, network traffic capture, and system cartography.

4.4.2. Step 2: Application protocol equivalence

The second step of the enrichment process is to associate events on actuators with ICS protocol commands. For instance, to cause a rising edge ($\uparrow$) of a discrete actuator VTP1 using Modbus, an attacker can use either function code 5 (writing a single coil) or 15 (writing multiple coils). This mapping is performed through an attribute equivalence concretion operator $E_{\Pi_2}^{-1}$.

To define Π_2, we propose to characterize the actuator access commands using three features: (i) the operation type, (ii) the variable type, and (iii) the variable access mode. We discuss the details for the Modbus protocol but the procedure easily adapts to any other industrial protocols.

- Operation type: *Specific* or *Generic*. This corresponds to a type of operation to be performed on the actuator such as a *read* or a *write* operation. Modbus allocates a specific field, called *function code*, to characterize the operation (for instance, function code 5 for discrete outputs or 15 for multiples writes). However generic types may be encountered in other protocols like Ethernet/IP.

- Variable type: *Explicit* or *Implicit*. The Modbus variable type is implied by the operation (function code) but in other cases variable's access location is used (Ethernet/IP), or described by explicit fields like in the case of Manufacturing Message Specification (MMS).

- Variable access mode: *Address-based* or *Identifier-based*. To refer to variables, Modbus uses addresses, while complex combinations of class, object and attribute identifiers or symbolic identifiers are possible like in MMS. In the case of address-based access, a variable address may be *Enumerated* (i.e. explicit address list) or identified by a *Starting address* and a *Count* or an *End address*.

Note that the enrichment model is generic and expressive enough in order to handle cases which are not covered by the above classification.

In our example, while Modbus used specific operations with implicit variable types and an enumerated or start-and-count addressing, the application protocols fields combination that maps via Π_2 to the Modbus event "set bit number c" satisfies[3]:

$$((func, 5) \wedge (addr, c) \wedge (data, 1)) \vee$$
$$((func, 15) \wedge (start_{addr} \leq c \leq start_{addr} + count) \wedge (data[c - start_{addr}], 1))$$

4.4.3. Step 3: Cyber-domain protocol field completion

Eventually, we have to handle cyber-domain attributes which are irrelevant in the physical-domain like protocol specific fields (*transaction_id* and *protocol_id* for Modbus/TCP) and generic cyber-domain attributes like source network address. The concretion operator H^{-1}_{Att} is the concretion counter-pair of and attribute hiding abstraction operation. The mapping *Att* is simply the list of the protocol fields.

4.5. Alert fusion

The alert fusion phase decides, for each new alert A_{new} and a set of previous alerts $\mathcal{A}_{old}$ from the alert selection stage, the subset of $\mathcal{A}_{old}$ which are correlated with A_{new}. The decision is performed for each pair of (pre-processed) alerts

[3]To simplify, we do not discuss in this paper the cases of 16 bits access operations which can also be used to write process variables (i.e, function codes 6, 16, 22, 23). However, the same approach still applies as in the case of function codes 5 and 15.

(A_{new}, A_{prev}), $A_{prev} \in \mathcal{A}_{old}$. On the basis of these pairwise correlations, the correlator constructs meta-alerts (clusters of alerts). Two alerts A_i and A_j belong to the same meta-alert if either A_i, A_j are correlated, or if there exists an alert A_k such that A_i, A_k and A_k, A_j are respectively correlated. Upon the reception of a new alert, the correlator notifies the security operator about the updated set of meta-alerts and the alerts belonging to each meta-alert. A meta-alert can be updated as long as one of its constituent alerts is present in an alert window, i.e as long as it is selectable by an alert selection policy. The security operator is informed about the lifetime of each meta-alert. Note that meta-alerts can contain alerts from different alert windows.

To decide whether A_{new} and A_{prev} are correlated, we test, using a Satisfiability Modulo Theories (SMT) solver, whether there is an assignment to all or a subset of their common attributes which makes $A_{new} \wedge A_{prev}$ true. However, this constraint can be quite strong. For instance, if two Modbus payload-based IDS alerts A_1 and A_2 report the same source, same destination, and are both commands to override coils albeit with different function codes (5 and 15), we would still like to fuse them since they might be symptomatic of an attacker who uses different function codes for consecutive commands. Thus, a weaker yet still interesting constraint might be:

$$A_1.source\ IP = A_2.source\ IP \wedge A_1.dest.\ IP = A_2.dest.\ IP$$
$$\wedge\ A_1.function\ code \in WFC \wedge A_2.function\ code \in WFC$$

where WFC is the set of Modbus *write* function codes.

In general, weaker constraints are obtained by either: (i) partitioning the set of values of a common attribute and checking whether the attribute's values in the pair of tested alerts belong to the same partition, (ii) ignoring a common attribute. Examples of the first type of weak constraint include checking whether source/destination addresses belong to the same network zone (control, supervisory, etc.) or whether the function codes refer to the same type of operations on process variables (write, read, etc.). Information about the network's topology can be obtained either through the plant's asset inventory databases or using

passive asset discovery techniques [66], while the operation classes specific to each protocol can be gathered from the protocol specifications if available, or through reverse engineering [67].

By default, the correlation is tested using the strongest constraint (equality between all common attributes). If the correlation fails, we test with increasingly weaker constraints by applying (i) then (ii) on the attributes shared by the alerts that are compared. Note that the number of tested weak constraints is bounded since the set of common attributes is finite and the number of ways a constraint on each attribute can be weakened is finite (using either (i) or (ii) above). To avoid too weak correlations, the user can adjust two parameters: (a) the maximum number of tested weak constraints, (b) a list of attributes which cannot be ignored. Weak correlations are tested using a breadth-first search strategy starting from the strongest possible constraint and stopping the search when no more constraint satisfying both (a) and (b) can be further produced. The type of correlation (strong, weak through partitioning, weak through hiding) is specified in the alert correlations received by the security operator.

5. Evaluation

To evaluate our approach, we need a sufficiently complex ICS. However, due to privacy concerns, real data from operational plants is hard to obtain. As a substitute, we develop a testbed with a complex physical process and a realistic ICS architecture. This allows us to make our datasets, the system's description and our implementation publicly available. Compared to most experimental setups with publicly available datasets that can be found in the literature, our testbed is either comparable[4] or significantly more complex[5].

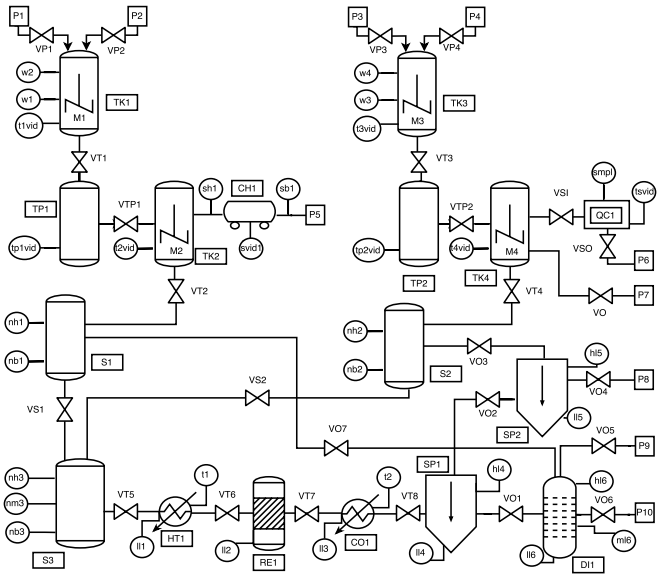

Figure 4: Physical process used in our evaluation

5.1. Testbed Description

Our evaluation testbed consists of a hardware-in-the-loop setting including a simulation of the physical process shown in Figure 4. The process consists in a chemical reaction to synthesize a product P10 from reactants in the silos S1 and S2. These reactants are manufactured from initial products P1, P2, P3, and P4 in several stages. The chemical reaction and the production of reactants span 15 activities organized in sequential and parallel configurations. The physical process involves 71 sensors and actuators and has two modes: manual and automatic. The manual mode allows the operators to carry interventions on the process. Upon receiving a command for manual mode, the PLCs puts the physical process in a stable condition by manipulating actuators. This physical process is simulated using OpenModelica. Its parameters (tank dimensions, heating temperatures, mixture timing, etc.) are set so that the physical process

[4]https://itrust.sutd.edu.sg/dataset

[5]https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets

undergoes several cycles during our simulations.

An ICS architecture with both control and supervisory levels, as shown in Figure 5, steers the physical process. The architecture involves real components found in operational ICS (i.e, PLCs, control servers, HMI, etc.). The control is distributed using three PLCs (Schneider M340, Schneider M580, and Wago IPC-C6 with additional RTU 750-873). The control logics are implemented in SFC. The Schneider M340 and Wago IPC-C6 PLCs support Modbus, while the Schneider M580 PLC supports both Modbus and SOAP/HTTP. At the supervisory level, an OPC-UA server polls data and relays commands to the PLCs. A supervisor HMI provides a global view of the state of the physical process to the operators. Engineering workstations allow engineers to change the control logics of the PLCs. Operators are not allowed to perform manual operations from the engineering workstations.

Figure 5 also depicts the IDS placement within the ICS. The network flow IDS [1] covers the whole ICS traffic through a mirror port at the level of the switch. The payload-based IDS [2] (PB1, PB2, and PB3) monitor the ICS trafic (Modbus/TCP and SOAP/HTTP) reaching each PLC. Finally, the physical process IDS [47] (PP1, PP2, and PP3) operate on the link between each PLC and its local HMI. For practical purposes, we monitor the actuators/sensors signals on each PLC using the supervisory HMI traffic. However, the physical process IDS can also directly monitor the communications between the sensors/actuators and the PLC using an adequate tap.

5.2. Attacks and operator interventions

To evaluate our approach, we need to generate attack traffic. However, attacks depend strongly on the monitored system and, contrary to traditional IT domains, no dataset with significant attacks targeting ICS is available. We, thus, identify possible attacks on sequential control systems, taking into account real attack cases [68, 16]. In particular, we are interested in process-oriented attacks that depend on the state of the physical process. Examples of such attacks include *sequence attacks* where a malicious ordering of individually legitimate

commands is used to put the physical process in a critical state.

We start by identifying critical states which the physical process must not reach. This includes, for instance, overflowing tanks or injecting a bad product mix. Then, we identify the sequence of actuator manipulations sufficient to put the physical process in each critical state. For example, we identify which valve need to be opened to overflow a given tank. We then associate these actuator manipulations with steps within the control logics executed by the PLCs. For instance, opening a valve is performed in a step where the tank is full.

To allow for a diverse set of attacks, we specify three dimensions along which the attacks can vary : (i) the source of the attack, (ii) the duration of the attack, and (iii) the protocols used to carry the attack.

The source of the attack can either be a host which is allowed to send commands to PLCs (such as an HMI or an OPC server), or a host which is not used for process operations (such as an engineering workstation). The duration of the attack varies depending on the number of steps in which actuator manipulations are performed.

Finally, the attacker can alternates between several protocols (SOAP/HTTP or Modbus) and use different operations within each protocol (Modbus with function codes 5 or 15 for instance). To simulate manual legitimate operator interventions, we also associate command sequences to control steps. By contrast to the attacks, these command sequences do not put the physical process in a critical state and are carried from authorized hosts.

5.3. Implementation

The IDS is implemented in C++/Python following their descriptions in the original papers [1, 47, 2]. We use Argus to recognize network flows, and rely on Zeek to extract Modbus/TCP data from network traffic. The correlator is implemented in Python. To test for satisfiability when applying the mapping rules to alerts, the correlator queries CVC4, an SMT solver, through the PySMT library. The communication between PySMT and the SMT solver uses the SMTLib standard. Thus, any SMT solver supporting the standard

and implementing the required theories can be used. To carry interventions, we use either : (i) the Modbus client module in Metasploit which is queried through its RPC interface, (ii) the OPC-UA server by means of the OPC-UA Pythonopcua client, or (iii) SOAP/HTTP requests by means of Zeep SOAP client. All our correlation performance analysis is performed on an Intel Dual Core i7 2.6 Ghz machine with 16 GB of RAM running Linux kernel 4.4.0. All the developed code is open source (GPL licence) and available at http://lig-g-ics.imag.fr/mediawiki/index.php/SourceCode

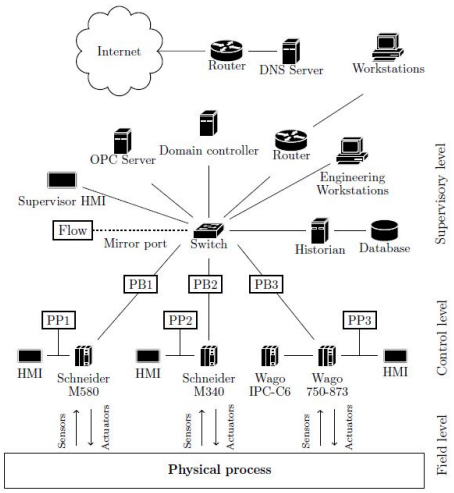

Figure 5: Simplified ICS architecture.

5.4. Datasets

We base our evaluation on 5 network captures spanning a total of 62 hours. To train the IDS base profiles, we use a 14 hour-capture free of any malicious actions and containing 27 legitimate operator interventions. This dataset reflects realistic conditions where certain legitimate behaviors are absent due to the limited training window. The number of activities' executions in the training data ranges from 28 to 155 cycles. We also generate 4 network captures which contain attacks as described in Section 5.2. These network captures span a total of 48 hours during which 26 attacks and 98 legitimate op-

erator interventions are carried. The datasets are available at http://lig-g-ics.imag.fr/mediawiki/index.php/Datasets.

6. Analysis

In this section, we present and analyze the results of our approach on the traces described in Section 5.4. We first describe the metrics we use for the evaluation and then we compare the results against a classical temporal window correlation approach.

Metrics. Our correlator links two alerts which are considered to be manifestations of an operator or attacker's intervention (i.e, sequence of commands). Correlation errors happen either when two alerts which are not from the same intervention are linked (*incorrect correlation*), or when two alerts which are from the same intervention are not linked (*missing correlation*). To evaluate our correlation approach with respect to these possible errors, we use the following two metrics :

$$\textit{False correlation rate} = \frac{\#\ \textit{incorrect correlations}}{\#\ \textit{produced correlations}}$$

$$\textit{Missing correlation rate} = \frac{\#\ \textit{missing correlations}}{\#\ \textit{expected correlations}}.$$

The *false correlation rate (FCR)* reflects the proportion of incorrect correlations in the correlations produced by the correlator. The *missing correlation rate (MCR)* represents the proportion of missing correlations in the number of correlations expected to be produced by the correlator given the alerts. The expected correlations can be computed since we have access to detailed information (time, protocols, target actuators, etc.) about the attacks and operator interventions carried in the evaluation traces. When specifying these correlations, we expect the correlator to group all side-effects of attacks. Such side effects include commands sent by the attacker to put the PLCs in manual/automatic mode and any process violations that might be indirectly caused by the attacks. For both metrics, lower values reflect better correlation performance.

In addition, we also evaluate the reduction in the number of alerts which are sent to the operator after the correlation. This reduction is measured through the following metric :

$$Reduction = 1 - \frac{\#meta\text{-}alerts + \#uncorrelated\ alerts}{\#total\ alerts}.$$

While higher values of the reduction metric mean less alerts to handle for the security operators, the reduction should still allow to distinguish between different operator/attacker interventions, i.e by keeping a low FCR value.

Note that our performance metrics are classical for the evaluation of alert fusion performance as presented in [8], for instance.

Results. To evaluate our approach, we compute the FCR, MCR and reduction metrics on the traces described in Section 5.4. We compare our results with a fixed-size sliding window correlator. This correlator keeps a fixed size window of old alerts and links each new alert with all alerts in the window. We compute the metrics for different sizes of the temporal window. Table 1 summarizes the results for both the activity-based and the temporal approaches. In this table, P_{all}, P_{adj} and P_{ada} refer to our activity-based approach using respectively the *all alerts in the activity*, *alerts in adjacent steps*, and *adaptive alert selection* policies discussed in Section 4.2. For the temporal correlator, T_{10s}, T_{1m} and T_{5m} refer respectively to window sizes of 10 s, 1 min and 5 min. In comparison, an activity's average duration is around 5 min and can range anywhere between 1 and 18 min. We set the maximum temporal window size to 5 min to limit false correlations on alerts belonging to multiple activities.

Overall, the results show that the activity-based approaches, in particular the ones using policy P_{all}, achieve the best FCR and MCR scores with the highest reduction in alerts across all traces.

With regards to the FCR, since the temporal correlator links all alerts which fall within the window without taking into account the compatibility of the attributes, our approach performs better as expected.

For instance, the temporal correlator does not distinguish between alerts

relative to attacks on different PLCs. We also note that, in comparison with the temporal correlator, the activity-based policies achieve better or comparable reduction results while still maintaining a low FCR. In fact, all false correlations using the activity-based approaches involve rare cases of unknown OPC-UA flow alerts that correspond to keep-alive messages.

With respect to the MCR, we observe that the activity-based approach with policy P-1 performs the best. In contrast, the temporal approaches perform badly especially for small time windows. Even though fewer correlations are missed by the temporal approach as the temporal window size gets bigger (T-1m and T-5m), deciding on a specific time window size remains problematic. For instance, a window size of 5 minutes captures all the expected correlations in Trace 3 but misses 37% of the expected correlations in Trace 1. Instead, the results show that better performance can be achieved by following an activity-based approach. The worse relative performance of the activity-based policies P_{adj} and P_{ada} compared to P_{all} indicates that activity-long attack manifestations are not exclusively limited to adjacent slots and do not appear at each slot of the activity.

To illustrate this point, let us consider an attack which consists in overflowing tank *TK3* by manipulating valve *VP3*. The attacker uses both Modbus and SOAP commands to put the PLC in manual mode and manipulate the valve. This attack causes 6 physical process (PP) IDS alerts spread throughout the activity and 3 cyber-domain alerts from the SOAP and Modbus payload-based (PB) IDS. In response to this attack, the activity-based approach with policy P_{all} generates a meta-alert containing all of the aforementioned alerts. Figure 6 shows the correlation graph corresponding to the meta-alert. Nodes in the graph refer to alerts, while edges refer to the correlations. Edge labels specify the strength of the correlations (*strong* or *weak*), which attributes are weakened in case of a weak correlation, and the absolute reception time difference between the correlated alerts in seconds. We observe that the absolute time difference between alerts vary widely from 0 s to 546 s. Thus, depending on the selected time window size, a subset of the alerts is successfully correlated by the temporal

approach. Note that all edges connected to node A19 have large time values. In fact, A19 is recorded many steps after the initial cluster of alerts, thus the alert selection policies P_{adj} and P_{ada} will also fail to capture it.

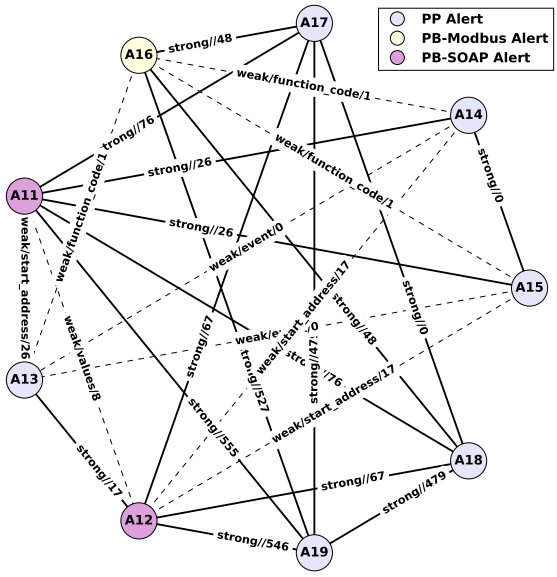

Figure 6: Correlation graph of a long-term attack using the activity-based approach with policy P-1

7. Conclusion

In this paper, we have developed an ICS-oriented correlation approach to link heterogeneous alerts from IDSs spanning both the cyber and the physical domains. The main contributions rest on two aspects: the mapping of physical domain alerts to the cyber domain through abstraction operations, and the definition of alert selection policies that take into account the runtime context of the physical process. The evaluation of our approach on a complex physical process subject to process-oriented attacks has shown good correlation metrics compared to temporal-based correlators. A natural extension of this work consists in reconstructing complex attack scenarios spanning multiple activities.

Such reconstruction could rely on a joint security-safety analysis [69] to produce the attack scenarios, and on the activity-specific meta-alerts generated by our approach to track the elementary attack steps.

References

[1] R. Barbosa, R. Sadre, A. Pras, Flow whitelisting in SCADA networks, Int. Journal of Critical Infrastructure Protection 6 (3-4) (2013) 150–158.

[2] O. Yüksel, J. den Hartog, S. Etalle, Reading between the fields: Practical, effective intrusion detection for industrial control systems, in: Proc. of the 31st Annual ACM Symp. on Applied Computing, SAC '16, ACM, NY, USA, 2016, pp. 2063–2070.

[3] M. Caselli, E. Zambon, F. Kargl, Sequence-aware Intrusion Detection in Industrial Control Systems, in: Proc. 1st ACM Workshop CPSS, 2015, pp. 13–24.

[4] P. Düssel, C. Gehl, P. Laskov, J.-U. Bußer, C. Störmann, J. Kästner, Cyber-critical infrastructure protection using real-time payload-based anomaly detection, in: E. Rome, R. Bloomfield (Eds.), Critical Information Infrastructures Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 85–97.

[5] A. Cárdenas, A. Saurabh, et al., Challenges for securing cyber physical systems, in: Workshop on Future Directions in CPS Security, 2009.

[6] A. Valdes, K. Skinner, Probabilistic alert correlation, in: Int. Workshop on Recent Advances in Intrusion Detection, Springer, 2001, pp. 54–68.

[7] H. Debar, A. Wespi, Aggregation and correlation of intrusion-detection alerts, in: RAID, Springer, Berlin, 2001, pp. 85–103.

[8] F. Valeur, G. Vigna, C. Kruegel, et al., Comprehensive approach to intrusion detection alert correlation, IEEE Transactions on Dependable and Secure Computing 1 (3) (2004) 146–169.

[9] B. Liskov, J. Guttag, Abstraction and Specification in Program Development, MIT Press, Cambridge, MA, USA, 1986.

[10] L. Saitta, J.-D. Zucker, Abstraction in artificial intelligence and complex systems, Computer Science, Springer, 2013.

[11] K. Stouffer, J. Falco, K. Scarfone, SP 800-82 Rev 2. Guide to Industrial Control Systems (ICS) Security, NIST (2015).

[12] J. Karl Heinz, M. Tiegelkamp, IEC 61131-3: Programming Industrial Automation, 2nd Edition, Springer, 2010.

[13] N. Falliere, L. O. Murchu, E. Chien, W32.stuxnet dossier, https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf, [Online, acc. : February-2022] (2011).

[14] F-Secure, Backdoor:w32/havex, www.f-secure.com/weblog/archives/00002718.html, [Online, acc. : February-2022] (2014).

[15] R. Khan, P. Maynard, K. McLaughlin, et al., Threat analysis of blackenergy malware for synchrophasor based real-time control and monitoring in smart grid, in: ICS-CSR '16, BCS Learning & Development, Swindon, UK, 2016, pp. 1–11. doi:10.14236/ewic/ics2016.7.

[16] US-CERT, Crashoverride, https://www.us-cert.gov/ncas/alerts/TA17-163A, [Online, acc. : February-2022] (2017).

[17] A. Otis, M. Belisle, J. Steele, MITRE ATT&CK for industrial control systems: Design and philosophy (2020).

[18] R. Sadoddin, A. Ghorbani, Alert correlation survey: Framework and techniques, in: Proc. of the 2006 Int. Conf. on Privacy, Security and Trust, PST '06, ACM, NY, USA, 2006, pp. 37:1–37:10.

[19] S. Salah, G. Maciá-Fernández, J. E. Díaz-Verdejo, A model-based survey of alert correlation techniques, Computer Networks 57 (5) (2013) 1289–1317. doi:https://doi.org/10.1016/j.comnet.2012.10.022.

[20] D. Kim, H. Bang, J.-C. Na, Intrusion alert normalization method using awk scripts and attack name database, in: The 7th International Conference on Advanced Communication Technology, ICACT 2005., Vol. 1, 2005, pp. 608–611 Vol. 1. doi:10.1109/ICACT.2005.245944.

[21] K. Stroeh, E. R. Mauro Madeira, S. K. Goldenstein, An approach to the correlation of security events based on machine learning techniques, Journal of Internet Services and Applications 4 (1) (2013) 7.

[22] Z. Liu, C. Wang, S. Chen, Correlating multi-step attack and constructing attack scenarios based on attack pattern modeling, in: 2008 International Conference on Information Security and Assurance (isa 2008), 2008, pp. 214–219.

[23] A. Chuvakin, Security correlation then and now: A sad truth about siem (2019).
URL https://medium.com/anton-on-security/security-correlation-then-and-now-a-sad-truth-about-siem-fc5a1afb1001

[24] U. Shankar, V. Paxson, Active mapping: resisting nids evasion without altering traffic, in: 2003 Symposium on Security and Privacy, 2003, pp. 44–61. doi:10.1109/SECPRI.2003.1199327.

[25] Y. B. Mustapha, H. Débar, G. Jacob, Limitation of honeypot/honeynet databases to enhance alert correlation, in: I. Kotenko, V. Skormin (Eds.), Computer Network Security, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 203–217.

[26] B. Morin, L. Mé, H. Debar, M. Ducassé, A logic-based model to support alert correlation in intrusion detection, Information Fusion 10 (4) (2009) 285–299.

[27] C. Kruegel, W. Robertson, G. Vigna, Using alert verification to identify successful intrusion attempts, in: Practice in Information Processing and Communication (PIK 2004), 27(4):219 – 227, October, 2004.

[28] L. Lemaire, J. Vossaert, J. Jansen, V. Naessens, Extracting vulnerabilities in industrial control systems using a knowledge-based system, in: Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research, ICS-CSR '15, British Computer Society, Swinton, UK, UK, 2015, pp. 1–10. doi:10.14236/ewic/ICS2015.1.

[29] L. Lemaire, J. Lapon, B. De Decker, V. Naessens, A sysml extension for security analysis of industrial control systems, in: Proceedings of the 2nd International Symposium on ICS & SCADA Cyber Security Research 2014, ICS-CSR 2014, BCS, Swindon, GBR, 2014, p. 1–9. doi:10.14236/ewic/ics-csr2014.1.

[30] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Trans. Inf. Syst. Secur. 6 (4) (2003) 443–471. doi:10.1145/950191.950192.

[31] B. Morin, H. Debar, Correlation of Intrusion Symptoms: An Application of Chronicles, Springer, 2003, pp. 94–112.

[32] S. Roschke, F. Cheng, C. Meinel, A new alert correlation algorithm based on attack graph, in: Á. Herrero, E. Corchado (Eds.), Computational Intelligence in Security for Information Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 58–67.

[33] E. Godefroy, E. Totel, M. Hurfin, F. Majorczyk, Generation and assessment of correlation rules to detect complex attack scenarios, in: 2015 IEEE Conference on Communications and Network Security, CNS 2015, Florence, Italy, September 28-30, 2015, IEEE, 2015, pp. 707–708. doi:10.1109/CNS.2015.7346896.

[34] P. Ning, Y. Cui, D. S. Reeves, Constructing attack scenarios through correlation of intrusion alerts, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS '02, ACM, New York, NY, USA, 2002, pp. 245–254.

[35] L. Briesemeister, S. Cheung, U. Lindqvist, et al., Detection, correlation, and visualization of attacks against critical infrastructure systems, in: 8th Annual Conf. on Privacy, Security and Trust, Ottawa, Canada, 2010, pp. 15–22.

[36] D. Lanoe, M. Hurfin, E. Totel, A scalable and efficient correlation engine to detect multi-step attacks in distributed systems, in: 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS), 2018, pp. 31–40. doi:10.1109/SRDS.2018.00014.

[37] W. Ren, T. Yu, T. Yardley, K. Nahrstedt, Captar: Causal-polytree-based anomaly reasoning for scada networks, in: 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), 2019, pp. 1–7. doi:10.1109/SmartGridComm.2019.8909766.

[38] C. Feng, T. Li, D. Chana, Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks, in: 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2017, Denver, CO, USA, June 26-29, 2017, IEEE Computer Society, 2017, pp. 261–272. doi:10.1109/DSN.2017.34.

[39] A. Lemay, A. Sadighian, J. Fernandez, Lightweight journaling for scada systems via event correlation, in: M. Rice, S. Shenoi (Eds.), Critical Infrastructure Protection X, Springer International Publishing, Cham, 2016, pp. 99–115.

[40] C. Zhou, S. Huang, N. Xiong, et al., Design and analysis of multimodel-based anomaly intrusion detection systems in industrial process automa-

tion, IEEE Trans. Systems, Man, and Cybernetics: Systems 45 (10) (2015) 1345–1360.

[41] B. Zhu, S. Sastry, SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy, in: Proceedings of the First Workshop on Secure Control Systems (SCS '10), 2010.

[42] R. Mitchell, I.-R. Chen, A survey of intrusion detection techniques for cyber-physical systems, ACM Comput. Surv. 46 (4) (2014) 55:1–55:29.

[43] L. Zhang, Multi-view approach to specify and model aerospace cyber-physical systems, in: Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on, 2013, pp. 595–602.

[44] S. Han, M. Xie, H.-H. Chen, Y. Ling, Intrusion detection in cyber-physical systems: Techniques and challenges, Systems Journal, IEEE 8 (4) (2014) 1049–1059.

[45] D. Hadziosmanovic, R. Sommer, E. Zambon, Through the Eye of the PLC: Towards Semantic Security Monitoring for Industrial Control Systems, in: Proc. ACSAC 14, 2014.

[46] S. Adepu, A. Mathur, Using process invariants to detect cyber attacks on a water treatment system, in: J.-H. Hoepman, S. Katzenbeisser (Eds.), ICT Systems Security and Privacy Protection, Springer International Publishing, 2016, pp. 91–104.

[47] O. Koucham, S. Mocanu, G. Hiet, et al., Detecting Process-Aware Attacks in Sequential Control Systems, in: NordSec 2016, Oulu, Finland, 2016, pp. 20–36.

[48] J. Schumann, P. Moosbrugger, K. Rozier, R2U2 : Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems, in: 6th Int. Conf. Runtime Verification (RV'15), Vienna, Austria, 2015, pp. 233–249.

[49] a. Carcano, a. Coletta, M. Guglielmi, M. Masera, I. Nai Fovino, a. Trombetta, A multidimensional critical state analysis for detecting intrusions in SCADA systems, IEEE Transactions on Industrial Informatics 7 (2) (2011) 179–186.

[50] R. Mitchell, I.-R. Chen, Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems, Dependable and Secure Computing, IEEE Transactions on 12 (1) (2015) 16–30. doi:10.1109/TDSC.2014.2312327.

[51] H. R. Ghaeini, D. Antonioli, F. Brasser, A.-R. Sadeghi, N. O. Tippenhauer, State-aware anomaly detection for industrial control systems, in: Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 1620–1628. doi:10.1145/3167132.3167305.

[52] Z. Yang, L. He, P. Cheng, J. Chen, D. K. Yau, L. Du, PLC-Sleuth: Detecting and localizing PLC intrusions using control invariants, in: 23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020), USENIX Association, San Sebastian, 2020, pp. 333–348.

[53] C. Feng, V. R. Palleti, A. Mathur, D. Chana, A systematic framework to generate invariants for anomaly detection in industrial control systems, in: 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019, The Internet Society, 2019.

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

[55] US-CERT, Crashoverride, https://www.us-cert.gov/ncas/alerts/TA17-163A, [Online, acc. : February-2022] (2017).

[56] S. Cheung, K. Skinner, Using Model-based Intrusion Detection for SCADA Networks, in: Proc. SCADA Security Scientific Symposium, 2007, pp. 127–134.

[57] S. Ponomarev, T. Atkison, Industrial Control System Network Intrusion Detection by Telemetry Analysis, IEEE Transactions on Dependable and Secure Computing 5971 (c) (2015) 1–1. doi:10.1109/TDSC.2015.2443793. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7122336

[58] R. R. R. Barbosa, R. Sadre, A. Pras, Difficulties in Modeling SCADA Traffic : A Comparative Analysis, in: Proceedings of the 13th international conference on Passive and Active Measurement (PAM '12), 2012, pp. 126–135.

[59] N. Goldenberg, A. Wool, Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems, International Journal of Critical Infrastructure Protection 6 (2) (2013) 63–75.

[60] A. Kleinmann, A. Wool, Automatic construction of statechart-based anomaly detection models for multi-threaded scada via spectral analysis, in: Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy, CPS-SPC '16, ACM, New York, NY, USA, 2016, pp. 1–12.

[61] M. Yoon, G. Ciocarlie, Communication Pattern Monitoring : Improving the Utility of Anomaly Detection for Industrial Control Systems, in: SENT, 2014.

[62] H. Lin, A. Slagell, C. Di Martino, et al., Adapting Bro into SCADA: building a specification-based intrusion detection system for the DNP3 protocol, in: Proc. CSIIRW '13, 2013, pp. 1–4.

[63] M. Parvania, G. Koutsandria, V. Muthukumary, S. Peisert, C. McParland, A. Scaglione, Hybrid control network intrusion detection systems for auto-

mated power distribution systems, in: Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on, 2014, pp. 774–779. doi:10.1109/DSN.2014.81.

[64] M. Conti, D. Donadel, F. Turrin, A survey on industrial control system testbeds and datasets for security research, IEEE Communications Surveys & Tutorials 23 (2021) 2248–2294.

[65] M. Wood, M. Erlinger, Intrusion detection message exchange requirements, RFC 4766 (2007) 1–25.

[66] A. Wedgbury, K. Jones, Automated asset discovery in industrial control systems: Exploring the problem, in: Proc. of the 3rd Int. Symp. for ICS & Scada Cyber Security Research, BCS, Swindon, UK, 2015, pp. 73–83.

[67] D. Nardella, Snap7, http://snap7.sourceforge.net, [Online, acc. : February-2022] (2014).

[68] C. Robert T. Marsh, Critical foundations: Protecting america's infrastructures, Tech. rep., The Report of the President's Commission on Critical Infrastructure Protection (October 1997).

[69] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, et al., A survey of approaches combining safety and security for industrial control systems, Reliability Engineering & System Safety 139 (2015) 156 – 178.

Table 1: Evaluation results

	FCR (%)					
	Activity			**Temporal**		
Trace	P_{all}	P_{adj}	P_{ada}	$10s$	$1m$	$5m$
N1	0	0	0	0	0	0
N2	0	0	0	12	9	11
N3	0	0	0	9	8	5
N4	2	2	2	1	15	21
	MCR (%)					
	Activity			**Temporal**		
Trace	P_{all}	P_{adj}	P_{ada}	$10s$	$1m$	$5m$
N1	0	21	21	75	36	37
N2	4	15	15	53	10	8
N3	0	15	34	62	34	0
N4	1	14	19	62	34	6
	Reduction (%)					
	Activity			**Temporal**		
Trace	P_{all}	P_{adj}	P_{ada}	$10s$	$1m$	$5m$
N1	32	22	88	36	61	61
N2	8	8	8	7	7	8
N3	84	81	79	65	70	78
N4	83	82	81	69	77	80