Anomaly Detection for Industrial Control Systems using Process Mining

David Myers[a], Suriadi Suriadi[a], Kenneth Radke[a], Ernest Foo[a]

[a]*Queensland University of Technology (QUT), Australia*

Abstract

Industrial control systems (ICS) are moving from dedicated communications to switched and routed corporate networks, exposing them to the Internet and placing them at risk of cyber-attacks. Existing methods of detecting cyber-attacks, such as intrusion detection systems (IDSs), are commonly implemented in ICS and SCADA networks. However, these devices do not detect more complex threats that manifest themselves gradually over a period of time through a combination of unusual sequencing of activities, such as process-related attacks. During the normal operation of ICSs, ICS devices record device logs, capturing their industrial processes over time. These logs are a rich source of information that should be analysed in order to detect such process-related attacks.

In this paper, we present a novel process mining anomaly detection method for identifying anomalous behaviour and cyber-attacks using ICS data logs and the conformance checking analysis technique from the process mining discipline. A conformance checking analysis uses logs captured from production systems with a process model (which captures the expected behaviours of a system) to determine the extent to which real behaviours (captured in the logs) matches the expected behaviours (captured in the process model). The contributions of this paper include an experimentally derived recommendation for logging practices on ICS devices, for the purpose of process mining-based analysis; a formalised approach for pre-processing and transforming device logs from ICS systems into event logs suitable for process mining analysis; guidance on how to create a process model for ICSs and how to apply the created process model through a conformance checking analysis to identify anomalous behaviours. Our anomaly detection method has been successfully applied in detecting ICS cyber-attacks, which the widely used IDS Snort does not detect, using logs derived from industry standard ICS devices.

Keywords: ICS, SCADA, Critical Infrastructure, Security, Cyber Attack, Process Mining

1. Introduction

Industrial control systems (ICSs) include supervisory control and data acquisition (SCADA) systems. A typical ICS or SCADA network consists of devices such as programmable logic controllers (PLCs), which are controlled through the use of human-machine interfaces (HMIs) [1, 2]. In addition to PLCs and HMIs, ICS and SCADA networks can use remote terminal units (RTUs) to control devices such as sensors in the network, which are in turn operated by master terminal units (MTU) [2]. As SCADA networks can be spread over large geographical areas, SCADA devices are moving from dedicated communication equipment such as serial links to Ethernet based switched and routed networks, connected to corporate networks through specialised gateways [3]. This allows SCADA and ICS devices to be controlled through one central location and simplifies management of the devices [3]. However, connecting these SCADA devices to corporate networks potentially exposes them to the Internet, placing the devices at risk of cyber attacks.

There have been several cyber attacks on critical infrastructure, including ICS and SCADA networks [4]. ICS and SCADA networks typically have security measures implemented to detect these cyber attacks, most commonly by using intrusion detection systems (IDS). While real-time detection is the ultimate goal of IDS, a system which provides near real-time detection is still very useful. This is shown by the 290 reported ICS-CERT incidents in

Email addresses: d2.myers@qut.edu.au (David Myers), s.suriadi@qut.edu.au (Suriadi Suriadi), k.radke@qut.edu.au (Kenneth Radke), e.foo@qut.edu.au (Ernest Foo)

2016 [5]. Any IDS that allows the detection of intrusion, even on a daily basis, is a significant improvement, as only 2% of attacks were detected by an IDS [6]. These IDS devices come in two major types, first signature-based, and second statistical anomaly-based IDS. However, these devices do not detect process-related threats [7], which our research does. Therefore, our research would sit beside, and add to, existing IDS to detect a greater range of attacks.

In this paper, we define a process as a sequence of events, from start to finish of the process being conducted. An industrial process is a series of events involving physical industrial equipment to produce a product or provide a service. An example of an industrial process may be a conveyor belt sorting ore in a mining facility, where the industrial process is the series of events to complete this process; such as starting conveyor, detecting the type of ore, then directing the ore to the appropriate conveyor belt. This is in contrast with commonly used terms in literature, such as "process-related", or "process-oriented". Rather than simply capturing and monitoring one process variable (such as temperature) related to an ICS device as what has been done in existing work [8, 9], our approach proposed in this article looks beyond changes in a particular variable; instead, it looks at the way in which various sequences of "events" were executed and detect those deviant sequences. This approach allows a more comprehensive assessment of the potential problems (and by extension, security attacks) within the system.

In this paper, we define a process model as a representation of the sequence of events in a process, and their temporal relationships, visually constucting the overall process being conducted by the ICS devices from start to finish of the industrial process. Process models in the area of Process Mining are typically represented in Petri-net form. Process mining is a series of techniques used to discover and improve business processes, consisting of process discovery, conformance checking, and process enhancement. In this paper we use the process discovery and conformance checking to model an ICS process, and identify anomalies within the modelled process.

To detect anomalous behaviour, and potentially cyber attacks, on ICS and SCADA devices, we have developed a process mining anomaly detection method, which uses the conformance checking process mining analysis to identify anomalous events in ICS data logs. This anomaly detection method is conducted through gathering and pre-processing of ICS device logs, developing a model of expected behaviour, and using both the newly pre-processed event logs and a developed model as input for a process mining conformance checking activity. This method allows us to detect *control flow* (i.e sequences of events) deviations in industrial processes, leading to identification of potential cyber attacks. Such a control-flow-based detection method for cyber attacks would not have been possible using traditional signature- or statistical-based IDSs.

The significance of our research thus lies in its focus on the *overall process* of a system for detecting threats, such as the types of events that occurred and the *ordering* of events. This is in contrast to signature-based IDSs, which detect threats based on known cyber attack patterns, and statistic-based IDSs, which identify anomalous behaviour by statistical modelling. Currently, no IDS used for ICS systems focuses on the *overall industrial process* for detecting attacks. Furthermore, we have assessed the process mining anomaly detection method we present in this paper using widely used Siemens S7-1200 PLCs controlling scaled-down physical ICS systems. These ICS systems were deployed on a process control network, designed to replicate a typical ICS network. The industrial process was automated and Cyber attacks on these ICS systems were conducted.

The contributions of our paper are as follows: (1) a recommendation for ICS devices logging methods that is best-suited for process-mining analysis based on our experiments with various ICS logging methods methods; (2) a formalized approach for pre-processing and transforming typical device logs produced by ICS devices to *event logs* that are suitable for process mining based analysis; and (3) guidances on the generation of process models capturing the expected industrial processes of ICS systems and on how to apply conformance checking analysis, together with the derived event logs, to identify potential cyber attacks. We also present a case study to evaluate our process mining-based anomaly (including cyber attacks) detection method using device logs obtained from industry standard ICS devices.

The remainder of the paper is structured as follows: In Section 2, we outline the current landscape of ICS and SCADA Security, provide an overview of process mining techniques, and summarize the current use of process mining and log mining to detect anomalous behaviour in cyber security domains. In Section 3, we define our process mining anomaly detection method. In Section 4, we evaluate our process mining anomaly detection method by conducting a case study involving device logs containing cyber attacks on industry standard ICS devices. Section 5 discusses our results. Finally, conclusions and future works are provided in Section 6.

2. Background and Related Work

In this section we provide an overview of industrial control system security, process mining, and the detection of anomalies using process mining and log mining.

2.1. Industrial Control System Security

ICS devices are typically connected to a dedicated network, connected to a corporate network through the use of a specialised gateway [3]. These corporate networks can be broken into two separate networks; first the internal network, consisting of workstations and local severs, and a DMZ or external network, consisting of servers exposed to the Internet, such as public web servers. In addition to these corporate networks, ICS devices can be located in remote networks, connected via external virtual private networks (VPN), the Internet or dedicated communication paths [2].

There have been several cyber attacks on ICS and SCADA devices and infrastructure in the past, a recent notable example of this is the December 23, 2015 attack on three Ukrainian power companies causing power disruption to 225,000 customers [10]. In addition to this recent attack, Miller and Rowe list and describe 15 cyber attacks on ICS and SCADA devices between 1982 and 2012; which include critical infrastructure attacks such as the Siberian gas pipeline in 1982; Worchester, MA Airport, USA in 1997; the Russian Gazprom gas company in 1999; Maroochy Water System in Queensland, Australia in 2000; and the Davis-Besse Nuclear Power Plant in Ohio, USA in 2003 [4]. Traditional IDSs cannot detect process-related threats [7], such as deviation from the control flow (events conducted within a process and their ordering) of an industrial process. This paper thus demonstrates how process mining techniques (that is the *conformance checking* analysis in particular) can be productively applied to identify anomalous behaviour and cyber attacks that manifest themselves within the control flow of industrial processes.

2.2. Process Mining

Process mining techniques have traditionally been applied in business contexts (e.g. [11, 12, 13] in order to "discover, monitor, and improve real processes by extracting knowledge from event logs" [14]. Three main components of process mining techniques include *process discovery, conformance checking, and enhancement*. Process discovery is the most commonly-used type of process mining analysis; it involves the derivation of a process model, typically a Petri-net model, from pre-processed event logs extracted from organisations' information systems. A process model captures the *control flow* of a business, whereby business events and their ordering are explicitly expressed graphically. The second type of process mining analysis, conformance checking, also referred to as "Delta analysis" [15], is used to compare existing process models with event logs to determine the extent to which real behaviours captured in logs agree with the expeected behaviours [14]. Finally, the third process mining analysis, enhancement, is conducted to extract other important insights related to processes, such as performance analysis [16], resources interaction analysis [17], and root-cause analysis [18].

Case ID	Activity	Timestamp.
1	Pump On	2015-08-15:10:00:00.000+01:00
1	Check Temperature	2015-08-15:10:01:00.000+01:00
1	Check Water Level	2015-08-15:10:02:00.000+01:00
1	Pump Off	2015-08-15:10:03:00.000+01:00
2	Pump On	2015-08-15:10:04:00.000+01:00
2	Check Water Level	2015-08-15:10:05:00.000+01:00
2	Check Temperature	2015-08-15:10:06:00.000+01:00
2	Pump Off	2015-08-15:10:07:00.000+01:00

Table 1: Example minimal event log for process mining analysis, with Case, Event, and Timestamp attributes.

Unlike data mining, process mining does not use the concept of "features". In process mining, we develop a process model, and what could be interpreted as features would be associated with the perspective of the specific process being modelled, such as the sequence of events, and timing of events. In the case of the process mining conformance checking activity, the key feature is the sequence of events. Process mining activities start with event

logs, which are typically gathered from information systems, in our case ICS and SCADA devices. Table 1 shows an example of an event log which comprises of a set of events. An event, at minimum, should have (1) a case identifier referring to the case to which the event belongs (a case is an instance of a full sequence of events of a process) (2) an event name (where an *event* refers to a step within a case [19, 14]), and (3) ordering information, such as timestamp, to allow one to arrange events chronologically [14]. Event logs may also include other messages, including records, transaction logs and entries in a database [14]. These event logs are used as input into each of the process mining activities. The process discovery activity uses an event log as an input, then using an algorithm, generates a model representing the process mined from the event log. These output models are typically expressed in the form of Petri nets [20], but can include other modelling notations such as BPMN [21] and YAWL [22] There are several types of algorithms that can be used by the process discovery activity, including Abstraction-based, Heuristic-based, Search-based and Region-based algorithms [23].

Abstraction-based algorithms generate models by ordering relations of events in an event log. The first algorithm designed for process discovery, and the traditional algorithm used is the α-algorithm. The α-algorithm, first introduced in 2004, is designed to "rediscover" sound Workflow-nets (WF-nets), similar to petri-nets, from an event log [19]. This algorithm has been used as a base for developing new algorithms, with improvements to solve new, novel problems; including the $\alpha+$ [24], $\alpha++$ [25], $\alpha\#$ [26], $\alpha*$ [27], and β (also refered to as the tsinghua-α) [28] algorithms. Heuristic-based algorithms, such as the Heuristics Miner [29], and Flexible Heuristics Miner [30], generate models where events are ordered based on the frequency of events which occur. Due to being based on frequency of events, these algorithms are more adept to deal with noise in event logs [29], however events which happen fewer than a specified threshold may be ignored [23]. Search-based algorithms, such as the Genetic Algorithm Miner (GA) [31] are algorithms which "try to mimic the process of evolution" [28]. Genetic algorithms, like Heuristics algorithms, perform better while mining noisy event logs, however events which occur less than a specified threshold are removed in a "post-pruning" step [31].

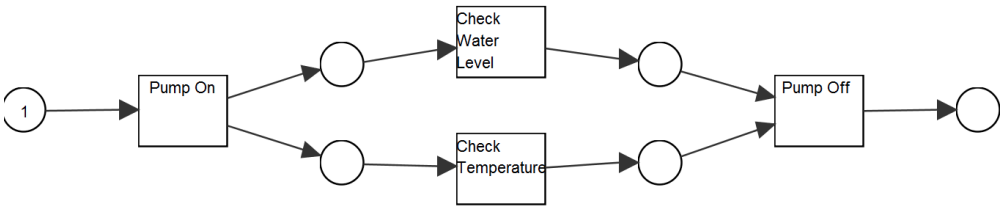

Figure 1: Example process model generated from the event log in Table 1, using the ProM Framework and the α-algorithm.

Log Pre-processing. In any process-mining analysis, it is important that one uses an event log of sufficient quality. Existing data pre-processing activities conducted within the process mining domain [15, 32, 23, 11, 13] are mainly concerned with filtering out irrelevant events or removing outlier events. Unfortunately, while existing techniques are applicable for the purpose of pre-processing ICS event logs, they are not sufficient for our purpose.

Firstly, to detect cyber attacks, we are actually interested in outlier events as they are the ones that are likely to contain anomalous behaviours indicating security breaches. Therefore, outlier events are *essential* in our analysis and are the main focus in our approach. Traditional pre-processing methods which eliminate infrequent outlier events or cases are thus inappropriate for the purpose of process mining-based analysis for anomaly detection.

Secondly, event logs generated from typical ICS systems do *not explicitly correlate* each event to a particular case or process instance (see Section2.2). In fact, ICS logs tend to only record *status* information (hence, not a process-oriented log) that is recorded every so often. For example, at time t_0, the conveyor belt identified as c_1 was moving in the left direction; some time later at time $t_0 + x$ (where x is a unit of time duration), the event log may record another entry stating whether c_1 was *still* moving in the left direction or whether the movement direction of c_1 has changed to the right. While each event is not explicitly correlated to a particular case, the nature of ICS operations are typically cyclic (that is, ICS operations typically consist of a sequence of steps that are operated repetitively with minor variations with each repetition), with each cycle being seen as a distinct process instance.

The challenge within the pre-processing stage is therefore on automatically mapping each event to its distinct process instances - also known as *event correlation* [14]. Currently, there is a dearth of literature in how to do such an

event correlation within the process mining domain in general, and within the control systems community in particular.

Furthermore, the example given above also shows that a new entry will be added to the ICS device log even though nothing 'new' has happened: even if the conveyor belt c_1 has not changed its movement direction, a new status information will still be recorded stating the status of c_1. From a process mining analysis perspective, status information that does not indicate the occurrence of anything new (we denote such log entry as a *non-event record*) is redundant. For other purposes, the multiple occurrences of non-event records may be important, for example, showing the systems and logging information is healthy and functioning. Therefore, removal of non-event entries is another data pre-processing activity that needs to be performed when transforming ICS log into an event log. Again, such a situation is not commonly encountered in the logs used in the process mining domain.

2.3. Anomaly Detection using Log Data

There have been several examples of the use of process mining within a security context [15, 32, 23]. These previous works focus on the application of two of the three primary activities of process mining, the process discovery and conformance checking/delta analysis activities. The first found example of the application of process mining to the area of security was from van der Aalst and de Medeiros in 2005 [15], which introduced the concept of using Process Mining to detect anomalous process executions. In this case, van der Aalst and de Medeiros introduce an event log, containing the expected behaviour of a business process, and using the α-algorithm generate a process model representing the expected behaviour [15]. Using the generated model, with conformance checking/delta analysis, van der Aalst and de Medeiros replay additional event logs in an activity named playing the "token game" [15]. The "token game" replays the new event logs against the expected behaviour, showing the anomalous event where the log derivates from the expected behaviour [15].

Conformance checking was used again in 2012 by Accorsi and Stoker [32], expanding upon van der Aalst and de Mederios's use of the process mining activity for anomalous process detection. Using simulated logs based on a bank loan application scenario and domain expert knowledge, Accorsi and Stocker demonstrate the potential use of conformance checking for security analysis, such as security auditing [32]. Accorsi and Stoker first define a list of requirements for the business process, and outline 10 security requirements for the loan application example, used to validate results. Using the conformance checking plug-in "Conformance Checker", for the ProM Toolkit [33], an event log can be replayed, to find derivations in the event logs through determining the *fitness* of an event log. The fitness of the event log is found during the conformance check, where event log is replayed on the process model during a "token game". In a conformance check, a fitness of 1 indicates the event log matches the model, and a fitness of < 1 indicates the event log does not match the model [32]. Further, a fitness of 0.99 can indicate a log with low derivation, where a fitness of 0.50 can indicate a log with high derivation. Using additional plugins for the ProM Toolkit, including "LTL Checker", "SCIFF Checker", and the additional "CLIMB Framework", Accorsi and Stoker were able to verify 9 of their 10 listed security requirements, showing process mining can be effectively used for security auditing [32].

Expanding on their previous work, in 2012 Accorsi, et. al. use the process discovery activity to demonstrate the potential of process mining for security auditing [23]. Accorsi, et. al. describe the currently used types of algorithms used in process mining, and evaluate each of these algorithm types for use in security auditing. Accorsi, et. al. outline three key types of algorithms which can potentially be used within security auditing: Heuristic-based, Search-based, and Region-based algorithms; as well as outlining two types of algorithms which are described as not suitable for use in security auditing: Abstraction-based algorithms and fuzzy mining methods [23]. In addition to describing the types of algorithms and potential for use in security auditing, Accorsi, et. al. suggest the use of different perspectives that can be mined for valuable information that could be used in an audit, such as the use of the Timing perspective to uncover covert channels, and the Case and Data perspective to detect information leakage [23]. However Accorsi, et. al. defer further analysis of the algorithms and cases to a future paper [23].

Process mining is a research area commonly associated with data mining and log mining. Log mining has previously been used to detect process related threats in SCADA systems. In 2012, Hadžiosmanović developed MELISSA (mining Event Logs For Intrusion in SCADA Systems), a tool designed for finding "undesirable behaviour", including misconfiguration, anomalous behaviour and attacks [7]. Making use of it's"data preparator" (DP) engine, used for data aggregation and translation to ready the data for pattern mining [7], tasks similar to those in a pre-processing stage in process mining to prepare event logs for process mining based analysis; and pattern engine (PE), using the FP-Growth algorithm to mine for frequent patterns, MELISSA finds anomalous events, returning them for manual analysis [7].

Through using MELISSA on the event logs from a SCADA log of a two-week period containing 101,025 events, MELISSA found 198 events for inspection for domain experts. Out of these 198 events for inspection, 23 were found as suspicious, and 5 events were considered anomalous behaviour [7].

Existing intrusion detection systems have been used for identifying anomalous behaviours in ICS networks. In 2016, Udd, et. al. introduced a method of improving the existing intrusion detection tool Bro for the task of detecting anomalies on industrial control system networks [9]. Focusing on the IEC 60870-5-104 protocol, this method to detect anomalies uses two components, implemented in the Bro intrusion detection system; a learning component through a whitelisting system to define approved network traffic on the control system network, and a detection component to create alerts when traffic on the control system network does not match the defined whitelists [9]. Approved traffic is controlled through two whitelists, first for ARP traffic, where MAC addresses are paired with IP addresses ensuring only approved devices are allowed on the network, and second for TCP traffic, ensuring hosts only access the allowed IP addresses and ports. These whitelists are generated through listening to baseline network traffic, which are then used by the detection system to compare packets on the network to the generated whitelists [9]. If a packet does not match the whitelists, an alert is raised for further analysis.

In addition to the use of IDS for detecting anomalous behaviours on ICS networks, "process-oriented" techniques have been a recent focus for ICS and SCADA intrusion detection. In 2016, Colbert, et. al. outline a process-oriented method for intrusion detection by exploiting domain knowledge of control system operators and focusing on the "critical process variables" of the industrial process [34]. Colbert, et. al. define critical process variables as variables important to the industrial process operation, such as a sensor monitoring temperature of a device [34]. By working with the domain experts of the ICS and network security engineers, ranges of acceptable and critical values are defined and applied to the sensors being monitored. When the value of the sensor reaches one of these defined critical values, an alert is created and sent to a human analyst who receives these alerts through a web interface for further analysis and action [34].

The idea of process-oriented techniques were also used for intrusion detection in SCADA networks in 2016 by Nivethan and Papa [8]. Similar to Colbert, et. al., Nivethan and Papa used process semantics to identify anomalous behaviour in a process variable. Nivethan and Papa developed a framework of two parts; a language to identify what systems to monitor, and a compiler designed to automate the creation of rules for intrusion detection systems, including Snort and Bro [8]. Working with process engineers, ranges of acceptable values are identified, such as a temperature of a device in a process that must not exceed a specified value. The compiler uses these requirements to create a "monitoring profile" which can be rules or signatures for the IDS Snort, or scripts for the IDS Bro [8]. These rules will will create an alert when a process variable exceeds one of the defined values.

Similar to the works of both Colbert, et. al. and Nivethan and Papa [34, 8], which focus on monitoring key variables related to an ICS or SCADA process, such as the temperature of a device, and creating an alert for a human operator to interpret either through a web interface [34], or through the alerts from IDS devices such as Snort and Bro [8], Cheung et. al. [35] introduce a method of "model-based" intrusion detection for SCADA networks. Cheung et. al. create models of the modbus protocol, expected behaviour of a network, and the expected state of servers and services on the network. These models are implemented as rules for the commonly used IDS Snort, which send an alert to an operator if unexpected behaviour occurs. Caselli et. al. [36] introduces specification mining, and a three-stage method for using specification mining on an ICS network for creating rules for an IDS device, such as Bro. First, a system discovery stage identifies devices on a network being monitored. Second, documentation of the devices provided by operators or obtained from the internet is collected. Finally, the documentation is analysed and rules are create for an IDS.

These approaches by Colbert, et. al. [34], Nivethan and Papa [8], Cheung et. al. [35], and Caselli et. al. [36] are process-oriented techniques, which focus on monitoring *the process variables* of specific devices, or parts of the overall industrial process, such as the network or status of devices. In our paper, rather than focusing on a violation of a process variable, such as the temperature or status of a device, we focus on the violation of the control flow, or the *overall sequence of events from the start to finish of an industrial process*. The control flow of an ICS process is a critical element of the process being conducted, and changing the sequence of events can result in the disruption of the ICS process. We contend that the examination of the control-flow of an ICS network for IDS purposes has not received enough attention within the community. A closely related work is the work of Alizadeh, et. al. [37] which introduces an approach to identify "non-conforming user behaviour", which uses both the data perspective, and the process perspective, together for security auditing. This approach has the goal of finding deviations of the data usage

and *control-flow* of a process, by comparing the expected behaviour of a system (process perspective) with the user behaviour (data perspective), for security auditing purposes. This method, similar to our paper, focuses on the *control flow* of a process, and shows the further potential for process mining to be used for anomaly detection for industrial control systems.

Like the previous examples of the application of process mining methods to security contexts, such as security auditing, our paper focuses on the process discovery and conformance checking activities of process mining. Techniques such as log mining, which are comparable to process mining, have previously been used to successfully detect anomalous behaviour in SCADA event logs, using methods such as pattern mining. This method of detecting anomalous behaviour using pattern mining works under the assumption that events in the event log which occur frequently over an extended period of time are normal or expected events [7]. In this case, there is the potential that anomalous behaviour which has occurred over a long period of time can be considered normal behaviour. In contrast, the process mining conformance checking activity compares the events within an event log to a *model of the expected behaviour* of the system, having the potential for process mining to be used to detect anomalous behaviour in ICS event logs without being tainted by noise or other threat-related behaviours seen in the logs.

3. A New Process Mining Anomaly Detection Method

In this section we introduce our method for identifying anomalies in ICS and SCADA device logs. This process mining anomaly detection method is broken into four distinct stages. The first stage outlines ICS device log generation methods, which are used to record device logs suitable for pre-processing. The second stage is the device log to event log transformation stage, where ICS device logs are pre-processed, transforming them to event logs suitable for use in process mining activities. The third stage is the creation of a model of expected behaviour for the ICS system, where a model of the acceptable behaviour of the ICS system is modelled in Petri-net form. Finally, there is the Conformance Checking stage, where the newly pre-processed event logs are replayed upon the created model of expected behaviour, identifying anomalous cases and inserted event classes for further analysis. Our presented process mining anomaly detection method, as shown in Figure 2, is similar to existing methods for anomaly detection, such as anomaly detection using machine learning techniques. In our anomaly detection method, we adopt established process mining methods to identify anomalies on the control flow of an ICS process. ICS devices lend themselves to a process mining approach, as the ICS devices typically follow a more regimented order, than traditional IT systems.

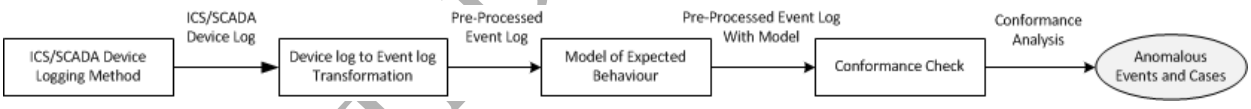

Figure 2: Process Mining Anomaly Detection Method.

3.1. Stage 1: Industrial Control System Device Logs

ICS and SCADA device logs can typically be generated and stored in two ways; firstly, locally on the device, and secondly, through an HMI, which sends commands to and receives information, such as device logs, from PLCs. HMIs typically store logs either on the system running the HMI, or on a Historian, a central repository for device logs.

There are several different methods commonly used for generating device logs. Firstly, a logging method, or how the logs are generated, recorded and stored. This logging method can commonly include “Circular log”, “Segmented circular log”, “Display system event at” and “Trigger event” methods. Circular log is a method that stores all logged events in one singular file with a specified file size. When that file size is reached, and the device log is full the log begins to overwrite from the beginning of the device log. Segmented circular log is similar to Circular log, however when the device log reaches the file size limit, it creates a new log up to a specified limit of logs. When the limit is reached, like Circular log, the log beings to overwrite from the beginning of the first device log. Display system event at device log is a Circular log, however when the log reaches a specified “fill level”, a user specified system event is triggered. Finally, a Trigger Event device log is a Circular log which triggers a specified system event when the log is full.

Secondly, there are several acquisition modes that can be used for individual device logs. These acquisition modes are methods to gather data from tags on the PLC, becoming the contents of the device log. There are three acquisition modes commonly available for use, "Cyclic", "On Change" and "On Demand" modes. The Cyclic acquisition mode polls the specified tags simultaneously on a specified time interval, logging the events in the device log. The On Change acquisition mode will log events to the device log when the value of the tag changes. Finally, the On Demand mode will log events when a script, code block or HMI receives a request to log data. Deadband constraints can be placed on individual tags in all acquisition modes, where a tag is only logged if it is either within, or outside of a specified value range. This can be used in conjunction with On Change to reduce log sizes.

To determine the most appropriate ICS device logging methods for process mining based analysis, we used four Siemens S7-1200 PLCs with six separate logging methods, including: one log with on change, one log with on change with deadbands, one log with cyclic in 5 second intervals, one log with cyclic with deadband in 5 second intervals, one log with cyclic acquisition mode in 1 second intervals, and finally one log with a combination of both on change and deadband acquisition modes in 5 second intervals. From this testing, further outlined in 5.2, we observe several differences between the data logging methods. First, we find that the Cyclic device logging method records far more information than the On Change device logging method. In addition to this, the On Change device logging method does not log events as they change. On Change device logs which record a tag which would have a very high variance, such as the a pressure level or water level, does not log as the values change, but instead logs events every 1 to 3 seconds, as shown by the in Table 13. Due to On Change device logs only recording values every 1 to 3 seconds, the configuration of a Cyclic device log with a low interval period, such as a one second interval will record the changing values more frequently than On Change logs.

As such, device logs using a Cyclic acquisition mode recorded at a frequent time interval, such as in one second intervals, should be used for process mining based analysis. This Cyclic acquisition mode records the largest amount of information, including events which are potentially excluded from On Change logs, and is the most useful as an input to a pre-processing stage. In addition, we do not recommend the application of deadbands on ICS device logs, as deadbands potentially remove information which can be useful for detecting anomalous behaviour. Deadbands have the potential to remove some pre-processing efforts, however it is expected that pre-processing will involve the removal of events which are not necessary for the process mining based analysis, potentially including events with deadbands applied, depending on the goal of the process mining activity.

3.2. Stage 2: Transforming Device Log to Event Log

Logs produced by ICS/SCADA systems (that is device logs) are, most of the time, not readily usable for process mining analysis. Therefore, some data pre-processing activities need to be conducted. These data pre-processing activities typically involved cleaning and/or filtering of events, as well as removing outlier events (such as those events that occur infrequently) such that only relevant records are retained [14]. Furthermore, as stated in Section 2, in addition to common filtering of events, device logs also require specific types of pre-processing that are not quite commonly performed in traditional process mining case studies.

This section provides the details of our data pre-processing approach to derive a usable event log from raw logs produced by ICS/SCADA systems. A short informal description of each of the data pre-processing activity will be provided, followed by its formal definition.

3.2.1. Industrial Control Systems Device Logs

We first introduce the concept of *device status record*, which is an *individual* entry (or row) that is produced and recorded by ICS/SCADA devices. Typically, a device status record is expressed in a key-value pair with the 'key' being the name of the status variable being recorded. Each record in a device log also has a timestamp which states the time at which the status was recorded. A *device log* is therefore comprised of a set of device status record.

Definition 1 (Device Status Record). Let $\mathscr{S}$ be the *status record universe*, that is the set of all possible status record identifiers. A status record consists of a number of *status attributes*, such as status *variable names*, the corresponding status *variable values*, the *timestamp* at which the status variable is logged, and other data attributes.

Let $\mathcal{SA}$ be the set of all possible *status attributes*. For any status record $s \in \mathscr{S}$ and for any status attribute $a_s \in \mathcal{SA}$, $\#_{a_s}(s)$ refers to the value of the status attribute a_s for the status record s.

VarName	VarValue	Timestamp
Conv_Run_Status	0	2015-03-19 15:34:29
HMI_Tank_Pump_Running	-1	2015-03-19 15:35:02
...	...	...

Table 2: An Example of a Simple Device Log

Let $\mathcal{D}_{varName}$ be the set of possible status variable names and $\mathcal{D}_{time}$ be the set of possible timestamps. For each status variable name $n \in \mathcal{D}_{varName}$, $\mathcal{D}_{varValue\_n}$ is the set of the possible values for the status variable n.

We define a minimum of the following three attributes that each status record $s \in \mathcal{S}$ should have:

- $\#_{varName}(s) \in \mathcal{D}_{varName}$ is the *variable name* for the status record s
- $\#_{varValue}(s, n) \in \mathcal{D}_{varValue\_n}$ is the *variable value* for the status record s where $n = \#_{varName}(s)$
- $\#_{time}(s) \in \mathcal{D}_{time}$ is the timestamp of the status record s

□

Definition 2 (Device Log). An device log $\mathcal{L}_{\mathcal{D}} \subseteq \mathcal{S}$ is a set of device status records. □

Table 2 shows an example of a device log where each status record contains the minimum required attributes.

3.2.2. Device Log Filtering by Attributes

A typical step in any log pre-processing activity is the filtering of records that are not relevant for analysis. Filtering a device log based on the value of a status record attribute is one of the most commonly-performed log filtering. For example, if one is not interested in the status records related to the speed of a conveyer belt, one may remove all status record whose *varName* attribute contains the value of `BELT_SPEED`. A filtered-by-attribute device log is therefore a device log that contains no status records with unwanted status attribute values.

Definition 3 (Filtered-By-Attribute Device Log). $\mathcal{L}_{\mathcal{D}}$ be a device log, $\mathbb{P}(\mathcal{L}_{\mathcal{D}})$ be the powerset of $\mathcal{L}_{\mathcal{D}}$, $\mathcal{SA}$ be the set of status record attributes, and $\mathcal{D}_{a_f}$ be the set of all possible values for an attribute $a_f \in \mathcal{SA}$.

We define a filter-by-attribute function f_{att}: $\mathbb{P}(\mathcal{L}_{\mathcal{D}}) \times SA \times \mathcal{D}_{a_f} \rightarrow \mathbb{P}(\mathcal{L}_{\mathcal{D}})$ which takes, as inputs, a device log $\mathcal{L}_{\mathcal{D}}$, a device log attribute a_f, the value of the log attribute $val_{a_f} \in \mathcal{D}_{a_f}$, and produces a filtered-by-attribute device log $\mathcal{L}_{\mathcal{D}f_{att}} \in \mathbb{P}(\mathcal{L}_{\mathcal{D}})$ where $\mathcal{L}_{\mathcal{D}f_{att}} = \{\ s \in \mathcal{L}_{\mathcal{D}} \mid \nexists_{s' \in \mathcal{L}_{\mathcal{D}}} . \#_{a_f}(s') = val_{a_f}\ \}$.

□

One can imagine the application of multiple filter-by-attribute functions iteratively over a device log, each with a different combination of a status attribute and its corresponding value.

3.2.3. Non-Event-Filtered Device Log

In Section 2, we discuss about how a device log contains many non-event records which are not useful for process mining analysis. Therefore, these non-event records that exist in a device log need to be removed.

Removing non-event records involves a more elaborate sequence of operation as one needs to apply the concept of 'change over time'. That is, we will only remove a status record with timestamp t_1 if the *value* of their corresponding status *variable name* has *not* changed as compared to the last recorded value of the same *variable name* at time $t_1 - x$ (where x is a unit of time duration). If at a later time $t_1 + x$ we see another status record entry with the same *variable name* but with a different value, that status record will be kept. Obviously, the status record in which a *variable name* is seen for the first time will also be kept. A non-event-filtered device log is therefore a device log that contains only those status records indicating changes in the values of various status variable names.

Definition 4 (Non-event Filtered Device Log). $\mathcal{L}_{\mathcal{D}f_{att}}$ be an attribute-filtered device log, $\mathbb{P}(\mathcal{L}_{\mathcal{D}f_{att}})$ be the powerset of $\mathcal{L}_{\mathcal{D}f_{att}}$, $\mathcal{SA}$ be the set of status record attributes, and $\mathcal{D}_{varName}$ be the set of all possible values for an attribute $varName \in SA$.

We define a non-event removal function $f_{nonEvent}$: $\mathbb{P}(\mathcal{L}_{\mathcal{D}f_{att}}) \rightarrow \mathbb{P}(\mathcal{L}_{\mathcal{D}f_{att}})$ which takes an attribute-filtered device log and produces a non-event-filtered log $\mathcal{L}_{\mathcal{D}f_{nonEvent}} \in \mathbb{P}\mathcal{L}_{\mathcal{D}f_{att}}$ where $\mathcal{L}_{\mathcal{D}f_{nonEvent}} = \{\ s \in \mathcal{L}_{\mathcal{D}} \mid \forall_{s_1,s_2,s_3 \in \mathcal{L}_{\mathcal{D}}}$:

- $\#_{time}(s_1) \leq \#_{time}(s_2) \leq \#_{time}(s_3) \wedge$
- $\#_{varName}(s_1) = \#_{varName}(s_2) = \#_{varName}(s_3) \wedge$
- $\#_{varValue}(s_1, \#_{varName}(s_1)) \neq \#_{varValue}(s_2, \#_{varName}(s_2)) \wedge$
- $\#_{varValue}(s_2, \#_{varName}(s_3)) \neq \#_{varValue}(s_2, \#_{varName}(s_3)) \wedge$
- $\nexists_{s'_1 \in \mathcal{L}_{\mathcal{D}}}: (\#_{time}(s_1) \leq \#_{time}(s'_1) \leq \#_{time}(s_2) \wedge \#_{varName}(s_1) = \#_{varName}(s'_1)) \wedge$
- $\nexists_{s'_2 \in \mathcal{L}_{\mathcal{D}}}: (\#_{time}(s_2) \leq \#_{time}(s'_2) \leq \#_{time}(s_3) \wedge \#_{varName}(s_2) = \#_{varName}(s'_2))$

} □

3.2.4. Activity Name Derivation

An *event* implies that "something has happened" (e.g. a tank *starts* pumping water). In contrast, a *status record* describes the state of a particular element (e.g. the tank *is pumping* water). Section 3.2.3 describes how we remove all non-event records such that all status records that remain in the log are those that indicate changes. In other words, all status records that exist in a non-event-filtered device log are indeed events.

However, a status record is described by status variable names and their values; two different values for the same variable name will mean two different activities. For example, a status variable name `TANK_PUMP` may contain two different values: `0, 1` with the former indicating that the tank is not pumping and the later indicating that the tank is pumping. Therefore, each unique combination of a status variable name and a status variable value would indicate a different activity. To obtain an event log, we therefore need to map each unique combination of status variable name and its value to a particular activity name.

Definition 5 (Activity Name Derivation). Let $\mathcal{D}_{varName}$ be the set of all possible status variable names, $\mathcal{D}_{varValue\_n}$ be the set of possible status variable values for a given status variable name $n \in \mathcal{D}_{varName}$, and $\mathcal{D}_{act}$ be the set of possible activity names.

For each unique combination of a status variable name $n \in \mathcal{D}_{varName}$ and one of its values $val_n \in \mathcal{D}_{varName\_n}$, we define an *activity name derivation* function $f_{act}^{(n,val_n)}: \mathcal{D}_{varName} \times \mathcal{D}_{varValue\_n} \rightarrow \mathcal{D}_{act}$ which will map the given combination of n and val_n to a particular activity name $act \in \mathcal{D}_{act}$. □

3.2.5. Case Identifier Derivation

As described in Section 2, an event is always associated with a particular case identifier. A device log obtained from a ICS/SCADA system typically does not have such information available. As part of the data pre-processing activity, we need to be able to define what constitutes a case. Fortunately, asssuming that the device log extracted was produced using the cyclic logging mode (Section 3.1), it is possible to correlate each event to a particular case.

To do so, a domain expert will need to define the characteristics of the status record that indicates the start of a cycle. This is normally defined as a combination of a particular status variable name $varName_0$ and one of its values $varValue_0$. Assume that status records s_{c_0}, s_{c_1}, and s_{c_2} are the three earliest status records seen in the device log whose status variable names are $varName_0$ and whose variable values are $varValue_0$ (s_{c_0} occurred before s_{c_1}). The status record s_{c_0}, along with any subsequent status records that occurred *after* s_{c_0} up until, but excluding, s_{c_1} should be grouped together as one case, and as such, these records should be correlated to the same case identifier. Since these records belong to the very first cycle, for simplicity, we assign the value of '0' as the case identifier for these events.

Subsequent records from s_{c_1} up until, but excluding s_{c_2} should be grouped into another case with a different case identifier that is derived by incrementing the previous case identifier by 1. We should continue this process for all status records in the event log such that each status record is mapped to a particular case identifier.[1]

[1] Note that any status records that occurred before s_{c_0} will be discarded.

Definition 6 (Case Identifier Derivation). Let $\mathcal{L}_{\mathcal{D}f_{nonEvent}}$ be a non-event filtered device log, $init \in \mathcal{D}_{varName}$ be a status variable name, $val_{init} \in \mathcal{D}_{varValue\_init}$ be one of the possible values for the status variable $init$, and $\mathcal{D}_{id} \subseteq \mathbb{Z}_0^+$ be a set of all possible case identifier. The pair $(init, val_{init})$ represents the combination of status variable name and value that indicates the start of a device log cycle.

The function $precede$: $\mathcal{L}_{\mathcal{D}f_{nonEvent}} \rightarrow \mathcal{L}_{\mathcal{D}f_{nonEvent}}$ takes a status record $s_j \in \mathcal{L}_{\mathcal{D}f_{nonEvent}}$ and returns s_{j-1}, that is the *immediate preceding status records* of s_j such that $\nexists_{s' \in \mathcal{L}_{\mathcal{D}f_{nonEvent}}} : \#_{time}(s_{j-1}) \leq \#_{time}(s') \leq \#_{time}(s_j)$.

We define a *case identifier derivation* function $f_{id} : \mathcal{L}_{\mathcal{D}f_{nonEvent}} \rightarrow \mathcal{D}_{id}$ where for any status record $s_k \in \mathcal{L}_{\mathcal{D}f_{nonEvent}}$:

$$f_{id}(s_k) = \begin{cases} 0, & \text{if } \nexists_{s'_k \in \mathcal{L}_{\mathcal{D}}} : \#_{time}(s'_k) \leq \#_{time}(s_k) \ \wedge \ \#_{varName}(s'_k) = init \ \wedge \\ & \qquad \#_{varValue}(s'_k, init) = val_{init} \\ \#_{id}(precede(s_k)), & \text{if } \#_{id}(precede(s_k)) \neq \perp \ \wedge \ \#_{varName}(precede(s_k)) \neq init \ \wedge \\ & \quad \#_{varValue}(precede(s_k), init) \neq val_{init} \\ \#_{id}(precede(s_k)) + 1, & \text{if } \#_{id}(precede(s_k)) \neq \perp \ \wedge \ \#_{varName}(precede(s_k)) = init \ \wedge \\ & \quad \#_{varValue}(precede(s_k), init) = val_{init} \\ \perp, & \text{otherwise} \end{cases}$$

□

3.2.6. Deriving Event Log

Having defined a number of functions in the preceding sections, we are now in the position to derive an event log. An event log consists of a set of *events*. An event is defined by a number of attributes. At minimum, an event should have (1) a case identifier, (2) an activity name, and (3) a timestamp.

Each status record that exists in a non-event filtered device log can be mapped into an *event* by applying the activity name derivation function and the case identifier derivation function.

Definition 7 (Event Log). Let $\mathcal{L}_{\mathcal{D}f_{nonEvent}}$ be a non-event filtered device log, $\mathcal{D}_{varName}$ be the set of possible status variable name, $\mathcal{D}_{varValue\_n}$ be the set of possible values for a status variable name n, $\mathcal{D}_{id}$ be the set of case identifiers, $\mathcal{D}_{act}$ be the set of activity names, $\mathcal{D}_{time}$ be the set of possible timestamps, f_{act}^{n,val_n} be an activity name derivation function for a particular combination of status variable name n and one of its values $val_n \in \mathcal{D}_{varValue\_n}$, and f_{id} be a case derivation function. The timestamp value for each status record can be directly mapped to the timestamp of the derived event.

Given a case identifier $id \in \mathcal{D}_{id}$, an activity name $act \in \mathcal{D}_{act}$, and a timestamp $t \in \mathcal{D}_{time}$, $\mathcal{L}_{\mathcal{E}}^{\mathcal{L}_{\mathcal{D}f_{nonEvent}}}$ is an event log that is derived from a non-event filtered device log where $\mathcal{L}_{\mathcal{E}}^{\mathcal{L}_{\mathcal{D}f_{nonEvent}}} = \{ (id, act, time) \mid \exists_{s \in \mathcal{L}_{\mathcal{D}f_{nonEvent}}} :$

- $id = f_{id}(s) \wedge$
- $act = f_{act}^{n,val_n}(n, val_n)$ where $n = \#_{varName}(s)$ and $val_n = \#_{varValue}(s, n) \wedge$
- $time = \#_{time}(s)$

$\}$

□

To provide an overview of the pre-processing activity, we have outlined an simplified algorithm to summarise the steps involved in the transformation of ICS device logs to pre-processed event logs, shown below in Algorithm 1. Each of the functions in the algorithm relates to a definition described above. First, the "removeFilteredAttributes" function refers to Defintion 3, where events not relevant for process mining analysis are removed. The function "removeNon-eventRecords" refers to Definition 4, where non-event records are removed. The function "addHumanNames" refers to Definition 5, where a human readable variable name is derived. The function "addCaseIdentifiers" refers to Definition 6, where the case identifier is associated with each row of the event log. Finally, the function "collateIntoLog" refers to Definition 7, which derives the event log from the previously defined functions.

Algorithm 1: Simplified Algorithm for Transforming ICS Device Logs to Event Logs

```
Data: Industrial Control System Device Log.
Result: Pre-processed Event Log for Process Mining activities.
initialization;
while not at end of device log do
    read rows;
    for rows in device log do
        removeFilteredAttributes();
        removeNon-eventRecords();
        addHumanNames();
        addCaseIdentifiers();
        collateIntoLog();
    end
end
```

3.3. Stage 3: Model of Expected Behaviour

A process model, typically represented in WF-Net (Workflow Net) [19] or Petri-net [20], is a model which represents the life-cycle of a case, or process instance [14]. This model of process instance is used as part of the process mining conformance checking analysis, where pre-processed event logs are replayed on a process model. Each of these process models will be unique to the ICS system, and is intended to be a model of an entire process instance from start to finish with only acceptable events. Typically, an ICS network will not have a model of the expected behaviour in this form, but the function of the ICS systems are recorded in alternative forms, such as programming logic of the ICS devices. The design of this model is traditionally a manual process, and will generally require significant domain knowledge of the ICS system and industrial process being modelled.

Previous work has been conducted by Myers, et. al. [38] describing the use of process mining process discovery techniques to discover a process model of ICS devices for use in conformance checking activities. Myers, et. al. found that it was possible using process mining algorithms, to accurately discover a process model representing an ICS process, without the requirement of significant domain knowledge. Myers, et. al. compared the results of several algorithms, and found that the most suitable algorithm for creating a model of an ICS process was the Inductive Miner configured with Perfect fitness. Throughout our paper, we adopt this method of generating a process model of an ICS process.

3.4. Stage 4: Conformance Checking

To identify anomalies in the ICS based event logs, we use the conformance checking process mining activity. The conformance checking activity compares an existing process model, typically in Petri-net form, to a given event log by replaying the event log on the process model to calculate the fit of the model. The fitness of the model, between 0 and 1 determines how well the event log matches the given process model, both overall and on a per-case basis. The closer the fitness is to 1, the closer the match. To calculate the fit of our pre-processed event logs using conformance checking, we design a model of expected behaviour. As previously discussed, this technique has been successfully used by van der Aalst and de Medeiros in 2005 [15], and Accorsi and Stoker in 2012 [32], to identify anomalies in an information security context.

Figure 3: Process Mining Anomaly Detection Method Conformance Checking activity.

To conduct this conformance checking activity, pre-processed event logs as an output from Stage 2, in Section 3.2, are replayed with the event log upon the model of expected behaviour of the ICS system, modelled as an output from Stage 3, in Section 3.3, using a conformance checking capability of the process mining (ProM) toolkit Conducting this conformance check results in first, a series of all event cases. These cases have associated fitness values, where a fitness of 1 denotes that case matches the model of expected behaviour. Cases which have a fitness lower than 1 contain anomalous events which are not part of the expected behaviour of the device, and are the focus of the conformance analysis. Within these cases, these events are split in to two areas; first inserted events and second, missing events. Inserted events are when an event has occurred when it should not have, and missing events occur when an event should have happened but did not. Using these cases, and events, further analysis can be conducted by an operator with knowledge of the ICS system, to identify if these anomalous events are cyber attacks. Both the identification of inserted events and removed event classes should be indication of further analysis. Control system operators are needed to conduct further analysis to distinguish device malfunction, human error, and other legitimate actions such as maintenance from a cyber attack. For example, in the case of a device malfunction, the running industrial process will deviate from the model of expected behaviour, and will appear as anomalous behaviour by our anomaly detection method.

4. Evaluation

To evaluate our anomaly detection method outlined in Section 3, we conduct a case study of the application of our method on industry standard ICS devices. In Section 4.1, we outline our experimental set-up, including ICS systems, and ICS network topology. In Section 4.2.2 we describe our logging method and datasets gathered from the ICS systems, to be used as input to our event log pre-processing method. In Section 4.2.4, we pre-process our event logs in preparation for our conformance checking activity. In Section 4.2.5 we generate a model of expected behaviour using the process mining process discovery activity to be used in our conformance checking activity. Finally, in Section 4.2.6 we conduct our conformance check and outline our experimental results.

4.1. Experimental Setup

Our experimental equipment represents two separate industrial processes, which we use to generate two datasets and their associated models of expected behaviour. The first industrial process consists of four scale systems, shown in Figure 4, include; a bi-directional conveyor belt system, a water pump system and a “reactor” pressure vessel system. These devices are connected to a master power meter, where HMI commands and device logs flow through. Each of these four devices are connected to a separate Siemens S7-1200 PLC, an industry standard ICS system, which we control using a HMI. These three systems and master power meter form an industrial process, used to record ICS device logs and build our model of expected behaviour.

The industrial process begins with the bi-directional conveyor belt system, shown in Figure 5. The bi-directional conveyor system is a scaled down physically moving conveyor belt system, with two loops travelling different directions marked in Figure 5 as “1”, with two coloured pucks travelling on the conveyor belt. Two sensors, marked in Figure 5 as “2” control the conveyor belt. The first sensor which detects the colour of a passing puck, the second sensor detects the presence of a puck. Depending on the colour of the puck, a paddle, marked in Figure 5 as “3”, will change direction to ensure the puck is travelling on the correct side of the conveyor belt. In the case of Figure 5, the white pucks will be directed to the left loop, and black pucks directed to the right loop.

Each of these devices, shown in Figures 5,7,8, are controlled using a Siemens S7-1200, an industry standard PLC, shown in Figure 5. These four Siemens S7-1200 PLCs, and HMI, were connected to a multi-level network built

Figure 4: Scale water, bi-directional conveyor, "reactor" system, and master power meter used to represent an industrial process for the first dataset.

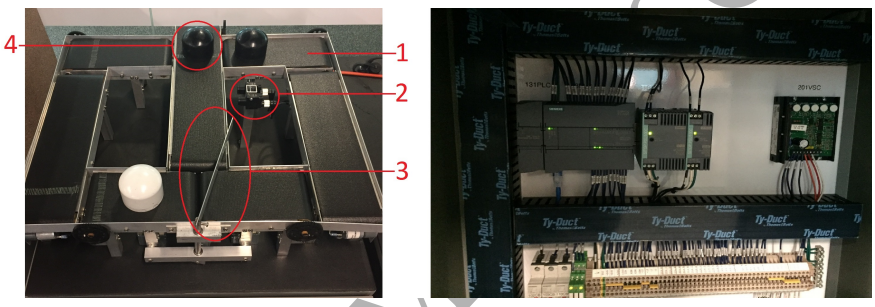

Figure 5: Scale bi-directional conveyor system, and controlling Siemens S7-1200 PLC used as part of our industrial process for the first dataset.

to represent a typical ICS network. Each of the PLCs as shown in Figure 5, are controlled via a HMI, and were directly connected to a "Process Control Network", outlined in Figure 6. This industrial process and process control network have been designed to represent a typical ICS or SCADA network as outlined in Section 2.1. The Siemens S7-1200 PLCs controlling this industrial process were programmed using STEP7, and the HMI, including all logging configurations, were designed and complied using WinCC V11 SP2, both part of the Siemens Totally Integrated Automation (TIA) portal V11, SP2.

On the completion of the conveyor belt system, the water tank system shown in Figure 7 is activated. This water tank system consists of a water pump used to transfer water from a lower reservoir to an upper reservoir. The upper reservoir, marked in Figure 7 as "1", contains a sensor which measures the current water level of the tank. While the water is below a specified level in the upper reservoir, the water pump moves water from the lower reservoir to the upper reservoir. Both the water pump and lower reservoir are located underneath the upper reservoir, marked in Figure 7 as "2". Once the specified water level has been met, the upper reservoir then slowly releases water back into the lower reservoir.

The final stage of the industrial process is the "reactor", a pressure vessel shown in Figure 8. This reactor is connected to an air compressor, which pressurises the reactor until it reaches a specified pressure level, measured in pounds per square inch (PSI). Once this pressure is reached, a solenoid, marked in Figure 8 as "1" activates, opening a valve to release air and lower the pressure to a second specified lower pressure level. Once that level is reached the solenoid closes and the reactor begins to build pressure again. This process repeats for a designated period of time, at which the reactor is emptied of any remaining air pressure.

The second industrial process consists of a separate industrial control system, a scale water tank system. This second industrial process is controlled by a National Instruments NI cRIO-9074 PLC, and is on a separate, contained network and is isolated from the first industrial process. The industrial process conducted by the National Instruments

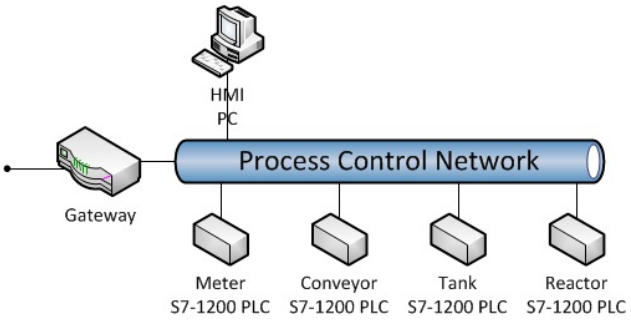

Figure 6: Siemens S7-1200 PLCs and HMI connected to the "Process Control Network" for the first dataset.

PLC begins by a water pump, connected to a water tank, pumping water into a second water tank. Once the second water tank has reached a specified water level, measured by a pressure sensor, the pump stops, and water begins to flow back into the first water tank. In addition to this water pump and pressure sensor, the first tank contains an emergency cut-off switch which stops the water pump if the water level of the first tank drops below a certain level. Finally, the first water tank contains a water heater, which maintains the temperature of water in the tank to a specified temperature. Once started, this process repeats until stopped. The National Instruments PLC controlling this industrial process was monitored and controlled by a HMI, which was used to record device logs.

While our experiments and two datasets were all conducted on device logs generated from two separate industrial processes, consisting of scale physical systems, which were a bi-directional conveyor belt system, a water pump system, a "reactor" system, and a master power meter for the first industrial process, and a scale water pump system for the second industrial process, they were not production industrial control systems. This was because our cyber attacks disrupted the control process of both running industrial processes, which is unacceptable on real systems, such as conveyor belt systems at airports, or real nuclear reactor control systems. Each of these scale physical systems was connected to a industry-standards ICS device, either the Siemens S7-1200 PLC, or the National Instruments PLC. All of the ICS devices were automated and monitored to complete the industrial processes, and the attacks were conducted by and monitored by humans throughout each of the exercises, outlined further in Section 4.2.2, making our test as realistic as possible.

4.2. Experiment

In the following sections we apply our process mining anomaly detection method to identify anomalous events found in ICS logs gathered from industry standard ICS devices.

4.2.1. Threat Model

The experimental setup, as previously outlined in Section 4.1, consists of two separate networks of ICS devices. The first dataset consists of a process control network containing ICS devices connected to a multi-level corporate network. The second dataset consists of a closed network of ICS devices. We assume for both datasets that the HMI and Historian record accurate device logs reflecting the current state of the PLCs, and their respective industrial processes. In addition, we assume the network packet captures record all network traffic including the actions of the attackers, and network traffic to and from ICS devices. This assumption is supported by our experimental setup, where all ICS devices for both datasets are connected to a network using a hub, allowing a network capture to record traffic from all connected devices.

The attackers have the ultimate goal of disrupting the industrial process. We adopt a similar threat model to the Dolev Yao model [39]. We assume that the attackers have access to the ICS network, however, do not have direct, physical access to any ICS devices, including all PLCs, HMI's, and Historians on the network. We assume that the attackers have no ability to modify ICS devices directly such as the ability to modify, or re-program any of the PLCs to change or affect the process. In addition, the attackers do not have the ability to modify or edit sensor data, or device logs recorded by the Historian. We assume the attacks conducted on the devices contain legitimate commands, including sending crafted packets, or flooding devices with packets to change a running industrial process. Using

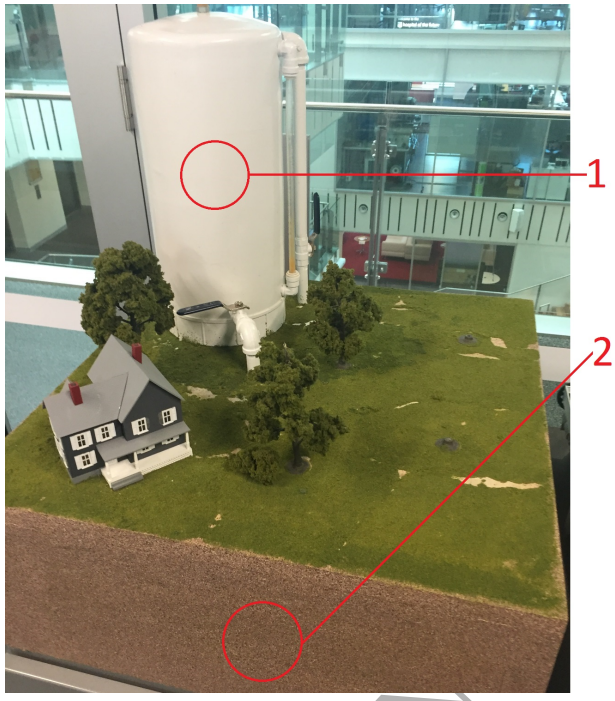

Figure 7: Scale water tank system used as part of our industrial process the first dataset.

these methods, the attackers can send messages to the PLCs or HMI in order to change and disrupt the currently running industrial process.

4.2.2. Industrial Control System Device Logs

A control system operator uses a HMI to send commands to PLCs, which store events that occur as device logs. ICS device logs are can be collected from PLCs and stored in one location through the use of a database server called a "Historian. PLCs are typically spread out over large geographical areas, and the internal memory of a PLC is small, thus Historians are used as a centralised server to collect and store device logs.

Throughout our experiment, two datasets were created. These datasets contained device logs which were recorded by the HMI, acting as a Historian, which was controlling each of the ICS devices, the Siemens S7-1200 PLCs and the National Instruments NI cRIO-9074 PLCs. For the first dataset, Each PLC controlling each system recorded six (6) separate device logs, as outlined in Table 3. These recorded device logs, which are logs of events which have occurred on a particular ICS device, such as PLCs, include the following: Cyclic device log with a 5 second interval; Cyclic device log with a 1 second interval; Cyclic device log in a 5 second interval configured with deadbands; On change log device log; On change device log with deadbands, and finally the initial pre-configured device logging method initially on the HMI, which was a combination of cyclic, on change and deadband device logging methods. These device logging methods were selected from our analysis of ICS device logging methods, conducted in Section 3. Device logs using the circular logging method were recorded using the segmented circular logging method. These device logging methods represent different combinations of the possible logging and acquisition methods available using the Siemens S7-1200 programming software WinCC V11 SP2 and STEP 7 V11 SP2, part of the Siemens TIA Portal V11 SP2. Throughout our experimentation, we use a Historian, through the use of our HMI, to gather device logs from each of the Siemens S7-1200 PLCs. In our second dataset, the HMI recorded device logs from the NI cRIO-9074 PLC in the same method, as a Cyclic device log, to be consistent with our pre-processing methods.

Device Log	Logging Method	Acquisition Mode
Conv_Log	Circular	On Change, Cyclic in 5 second intervals
Conv_Log_Cyclic	Circular	Cyclic in 5 second intervals
Conv_Log_Cyclic1s	Segmented Circular	Cyclic in 1 second intervals
Conv_Log_CyclicDeadband	Circular	Cyclic in 5 second intervals with Deadband
Conv_Log_OnChange	Circular	On Change
Conv_Log_OnChangeDeadband	Circular	On Change with Deadband
Master_Log	Circular	On Change, Cyclic in 5 second intervals
Master_Log_Cyclic	Circular	Cyclic in 5 second intervals
Master_Log_Cyclic1s	Segmented Circular	Cyclic in 1 second intervals
Master_Log_CyclicDeadband	Circular	Cyclic in 5 second intervals with Deadband
Master_Log_OnChange	Circular	On Change
Master_Log_OnChangeDeadband	Circular	On Change with Deadband
Tank_Log	Circular	On Change, Cyclic in 5 second intervals
Tank_Log_Cyclic	Circular	Cyclic in 5 second intervals
Tank_Log_Cyclic1s	Segmented Circular	Cyclic in 1 second intervals
Tank_Log_CyclicDeadband	Circular	Cyclic in 5 second intervals with Deadband
Tank_Log_OnChange	Circular	On Change
Tank_Log_OnChangeDeadband	Circular	On Change with Deadband
Reactor_Log	Circular	On Change, Cyclic in 5 second intervals
Reactor_Log_Cyclic	Circular	Cyclic in 5 second intervals
Reactor_Log_Cyclic1s	Segmented Circular	Cyclic in 1 second intervals
Reactor_Log_CyclicDeadband	Circular	Cyclic in 5 second intervals with Deadband
Reactor_Log_OnChange	Circular	On Change
Reactor_Log_OnChangeDeadband	Circular	On Change with Deadband

Table 3: Complete List of device logs recorded by HMI, including Logging Method, Acquisition Mode and Log Format.

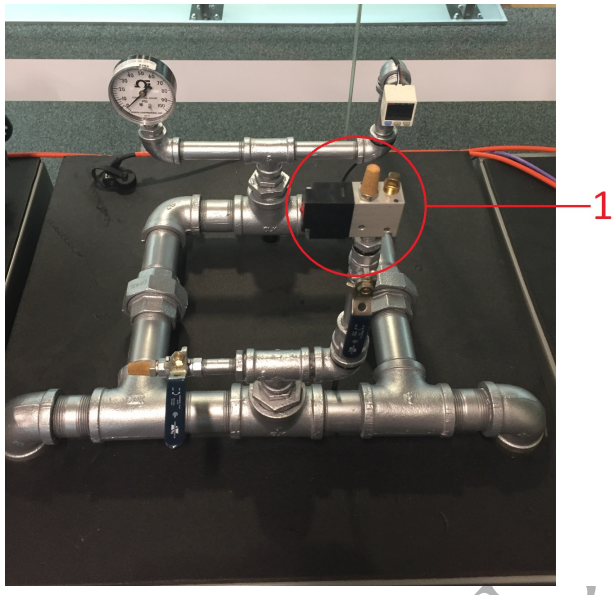

Figure 8: Scale "reactor" system used as part of our industrial process for the first dataset.

4.2.3. Datasets

Throughout our experiments we use two separate datasets[2], recorded from industry standard ICS devices, the Siemens S7-1200 PLCs, and the National Instruments NI cRIO-9074 PLC. Each datasets was comprised of two components; device logs recorded by a HMI, as outlined as outlined in Section 4.2.2, and network traffic captures recorded in PCAP format.

The first dataset was created on our process control network, consisting of four Siemens S7-1200 PLCs, each controlling a scale industrial system as described in Section 4.1. This first dataset contained control data, consisting of device logs and captures of the network traffic in PCAP files of the normal behaviour of the industrial control systems and the industrial process, and an attack dataset. The attack dataset, like the control dataset, consists of device logs and captures of the network traffic in PCAP files. The attack dataset contains attacks described in Rodofile et. al.[40]. A total of 21 cyber attacks that were conducted on the Siemens S7-1200 PLCs. These attacks comprised of 17 unique attacks, consisting of two major types; injection attacks, and flooding attacks, both with the goal of changing or disrupting the running industrial process. The attacks are outlined in Table 4.

The second dataset was created on the National Instruments PLC, controlling a individual scale industrial control system as outlined in Section 4.1. Similar to the first dataset, the second dataset contained control data, and attack data, consisting of device logs and captures of the network traffic in PCAP files. The attack data in the second dataset comprised of 9 cyber attacks conducted on the National Instruments PLC. These 9 cyber attacks comprised of 3 unique attacks, which were flooding attacks used to send commands to the National Instruments PLC, with the goal of disrupting the running industrial process. The 3 attacks in 9 variations are outlined in Table 5.

The attacks in the first dataset were conducted upon the scale industrial control systems, which were controlled by the Siemens S7-1200 PLCs, and varied in severity. As an example of an attack, the first attack conducted was to disrupt the conveyor belt system, by turning the conveyor belt system off while the conveyor belt process was running. In addition to these attacks which target individual systems, there were more severe attacks conducted such as attack 8, where a "master reset" was conducted, resetting all devices in the industrial process, and attack 14, where an "emergency stop" was conducted, halting the industrial process. These attacks all successfully disrupted the industrial process being conducted, and were recorded in the device logs gathered by the HMI. The attacks in the second dataset were conducted upon the scale system controlled by the National Instruments PLC. As an example of an attack, the first attack conducted was to disrupt the water pump, by flooding a message to turn off the water pump to the PLC

[2]The datasets are available from `https://cloudstor.aarnet.edu.au/plus/index.php/s/9qFfeVmfX7K5IDH`

Attack	Attack Description	Case ID
1	While Conveyor Running, Turn Conveyor Off	2
2	While Water Tank Running, Turn Wash Tank Off	3
3	While Water Tank Running, Turn Wash Tank Off	3
4	While Conveyor Running, Turn Pipeline Reactor On	5
5	While Reactor Running, Turn Pipeline Reactor Off	7
6	While Reactor Running, Turn Pipeline Reactor Off	7
7	While Reactor Running, Change Direction of Conveyor Belt	8, 9
8	While Any Device Running, Reset All Devices	12
9	While Water Tank Running Turn Conveyor Belt On	14
10	While Water Tank Running Turn Conveyor Belt On	14
11	While Conveyor Running Change Direction of Conveyor Belt	15
12	While Conveyor Running, Turn Conveyor Belt Off	16
13	While Reactor Running, Change Upper Threshold of Pipeline Reactor	19
14	While Any Device Running, Emergency Stop	20
15	While Reactor Running, Turn Water Tank On Manual Mode	25
16	While Conveyor Running, Turn Water Tank On Automatic Mode	25
17	While Conveyor Running, Turn Water Tank On Automatic Mode	30
18	While Conveyor Running, Change Direction of Conveyor Belt	31
19	While Water Tank Running, Change Lower Threshold of Pipeline Reactor	32
20	While Reactor Running, Turn Pipeline Reactor Off	35
21	While Water Tank, Running Turn Wash tank Off	36

Table 4: Description of attacks conducted on the Siemens S7-1200 PLCs throughout the first dataset.

Attack	Attack Description	Case ID
1	While water pump on, flood messages to turn water pump off.	4
2	While water pump off, flood messages turn water pump on.	6
3	While process is running, flood messages to turn on heater.	8
4	Over two cases, flood messages to turn water pump off.	11
5	Over two cases, flood messages turn water pump on.	13
6	Over two cases, flood messages to turn on heater.	15
7	While water pump on, flood messages to turn water pump off.	17
8	While water pump off, flood messages turn water pump on.	17
9	While process is running, flood messages to turn on heater.	18

Table 5: Description of attacks conducted on the National Instruments PLC throughout the second dataset.

Metric	Script 1	Script 2	Script 3	Script 4	Total
CPU Time	6.36s	2.14s	0.03s	0.03s	8.56s
Memory Usage	13.25MiB	13.23MiB	12.82MiB	12.63MiB	130MB
Storage Usage	84.70MB	335KB	424KB	413KB	85.87MB

Table 6: Calculated cost for pre-processing ICS device logs from the first dataset, recorded over an 8 hour period.

while the water pump was running. Out of the 9 attacks conducted, three of them were run over two instances of the industrial process. All 9 attacks conducted successfully disrupted the industrial process, and were recorded in the device logs gathered by the HMI and network packet captures.

4.2.4. Transforming Device Log to Event Log

To pre-process the event logs gathered from the HMI in both the first and second datasets, we developed a series of Python scripts to conduct pre-processing activities based from our event log pre-processing method, outlined in Section 3.2. Using these developed Python scripts, we pre-processed both the control and attack datasets Cyclic event log in 1 second intervals, a log snippet of which shown in Table 7, which we determined was the most suitable for process mining based analysis, as outlined in Section 3.1. Once these device logs were pre-processed for process mining based activities, the attack dataset was labelled with attack information.

To calculate the cost of the pre-processing activity, we have measured both the CPU-time and memory consumed by the python scripts used during the pre-processing of the first dataset. In addition, we have calculated the storage requirements of both the device log used as input to the pre-processing activity, and the pre-processed event logs. The CPU time used to execute python scripts was measured using the Python "cProfiler" module, and the memory usage of each of the python scripts was recorded using the Python "memory_profiler" module. The pre-processing activities were conducted on a Dell OptiPlex 9020 with an Intel i7-4770 3.4GHz CPU, 16GB Ram, and a 256GB SSD HDD running Windows 7 Enterprise SP1; and were implemented in Python version 3.6.2. The results of this cost analysis are shown in Table 6.

As can be seen in Table 6, the total CPU-time used by the four Python scripts as part of the pre-processing stage is 8.56 seconds. Each python script output a CSV formatted file, to be used as input for the next script, with the final script resulting in a pre-processed event log ready to be converted to XES format. The total storage space includes the device log used as input to the pre-processing steps, and all device logs generated during the pre-processing activity, which totals 85.87MB of storage space used. As a test of the scalability of our pre-processing method, we took the Cyclic event log and cut the total number of rows in half. We then measured the CPU-time of the scripts, using the Python cProfiler module as previously outlined. The log with one half of the total rows took approximately one half of the time to process, with Script 1 taking 3.677 seconds, Script 2 taking 1.291 seconds, Script 3 taking 0.0182 seconds, and Script 4 taking 0.0147 seconds. This indicates that the time required for pre-processing device logs will scale linearly with time.

After pre-processing, our event logs from both the first and second datasets were converted to XES format in preparation for use with the Process Mining toolkit (ProM) for our conformance checking activity. To convert our logs to XES format, the event logs need to be converted from their existing log format CSV (comma separated values), into the XES format. Previous work in process mining has used the tool Nitro [41] to convert logs from different formats into the XES format, however the software Nitro has been discontinued and adapted in to the Disco tool. For ease of use we have used the Disco tool to import the existing CSV event logs, and export in the XES format. These XES format logs are then imported into the ProM Toolkit version 6.7 for our conformance checking activity.

4.2.5. Model of Expected Behaviour

To conduct conformance checking on our two datasets of pre-processed event logs, in addition to the pre-processed event log, we must have a model of the expected behaviour of the ICS systems. To create a model of the expected behaviour of our ICS systems used in our experiments, as outlined in Section 4.1, we used the technique outlined in Section 3.3, by creating process models through a process mining process discovery activity. For our first dataset, we used our control dataset containing 8 hours of normal behaviour of the ICS devices, which we pre-processed following

VarName	VarValue	Case ID	VarName HR
Conv_Read_Conv_Present_PE	-1	3	Conv_Read_Conv_Present_PE(5) Detect Puck
Conv_Read_Solenoid_Left_Direction	-1	3	Conv_Read_Solenoid_Left_Direction(5) Check Direction
Conv_Read_Solenoid_Right_Direction	-1	3	Conv_Read_Solenoid_Right_Direction(5) Check Direction
Conv_Run_Status	0	3	Conv_Run_Status(5)Stopped
HMI_Conv_Finish	-1	3	HMI_Conv_Finish(5) Activated
Tank_Read_Pump_In_Auto	-1	3	Tank_Read_Pump_In_Auto(5) Activated
Tank_Read_Pump_Running	-1	3	Tank_Read_Pump_Running(5) Activated
HMI_Tank_Finish	-1	3	HMI_Tank_Finish(5) Activated
Tank_Read_Pump_In_Auto	0	3	Tank_Read_Pump_In_Auto(5) Unactivated
Tank_Read_Pump_Running	0	3	Tank_Read_Pump_Running(5) Unactivated
HMI_Pipe_Solenoid_Mode	-1	3	HMI_Pipe_Solenoid_Mode(5) Unactivated
Pipe_Read_Solenoid_Mode	-1	3	Pipe_Read_Solenoid_Mode(5) Open
HMI_Pipe_Solenoid_Mode	0	3	HMI_Pipe_Solenoid_Mode(5) Unactivated

Table 7: Snippet of pre-processed event log, transformed from ICS generated from the Cyclic data log.

our method outlined in Section 3.2, as input to the process discovery activity. For our second dataset, similar to the first dataset, we used control data containing the normal behaviour of the ICS devices. Using the Inductive Miner algorithm configured with perfect fitness, we created two process models, one for each of the datasets, containing the expected behaviour of the ICS systems. As an example, the process model for the first dataset is shown in Figure 9. As can be seen in Figure 9, the models for industrial processes can be complex. The process model generated from the first dataset consists of 69 Places, and 194 Arcs connecting events through 79 transitions.

4.2.6. Conformance Checking

To detect anomalies in our labelled, pre-processed event logs from each dataset, we use the conformance checking process mining activity. The conformance checking activity was conducted in the Process Mining toolkit (ProM) version 6.7. Using the *Replay a log on Petri net for Conformance Analysis* plugin for ProM 6.7 shown in Figure 10, we replayed the event log generated using our pre-processing method described in Section 3.1 on our model of expected behaviour. This activity was conducted for both datasets described in Section 4.2.3.

There were a total of 38 cases in our labelled, pre-processed event log for the first dataset. Out of the total number of cases, 22 contain a cyber attack, as shown in Table 4. All attacks, with exception to attack 7, occur within one case. The attack conducted in attack 7, spanned across both case 8 and case 9, where the direction of a conveyor belt was changed in case 8, then re-set to the proper direction in case 9. The results from the ProM conformance check are measured in fitness, in three categories; "Move-Log" fitness "Trace" fitness, and "Move-model" fitness. Move-Log fitness assumes the process model that has been used in the conformance checking activity is correct, and shows where the trace in the event log does not fit the process model. Conversely, Move-Model fitness shows where the model is incorrect in comparison to the event log. Trace fitness shows the overall fitness of the model and the log, which accounts for both Move-Log and Move-Model fitness. In our results, as our process model contains the expected behaviour of the ICS devices, we use the Move-Log fitness, and compare to the Trace Fitness, to analyse our results.

Using our labelled event log generated from our first dataset, we compare the results of the conformance check using the ProM process mining tool. The results of our conformance checking activity on the model of expected behaviour for the first dataset is summarised in Table 8, showing both the Move-Log and Trace Fitness results. Out of the total 38 cases in the first dataset, the Move-Log Fitness results identified 19 cases as anomalous, 14 of which are true-positives (66.66% of attacks), and the Trace Fitness results identified 23 cases as anomalous, 16 which are true-positives (76.14% of attacks). The move-log fitness shows that the process mining conformance check missed four of our attacks, the 2nd, 4th, 5th, and 21st attacks. The Trace Fitness results only missed two attacks, the 2nd and 5th attacks, however identified several incorrect cases as anomalous, or false positives, a total of 7 false positives compared to the Move-Log Fitness, which identified 5 cases incorrectly as anomalous.

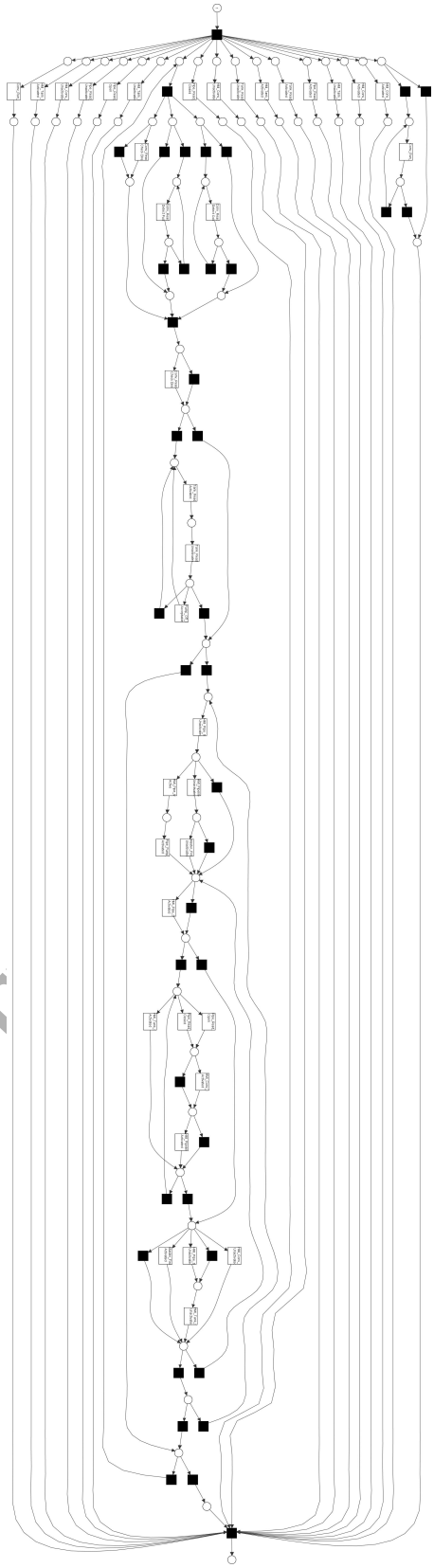

Figure 9: Petri-net model of expected behaviour, describing control flow process of our ICS.

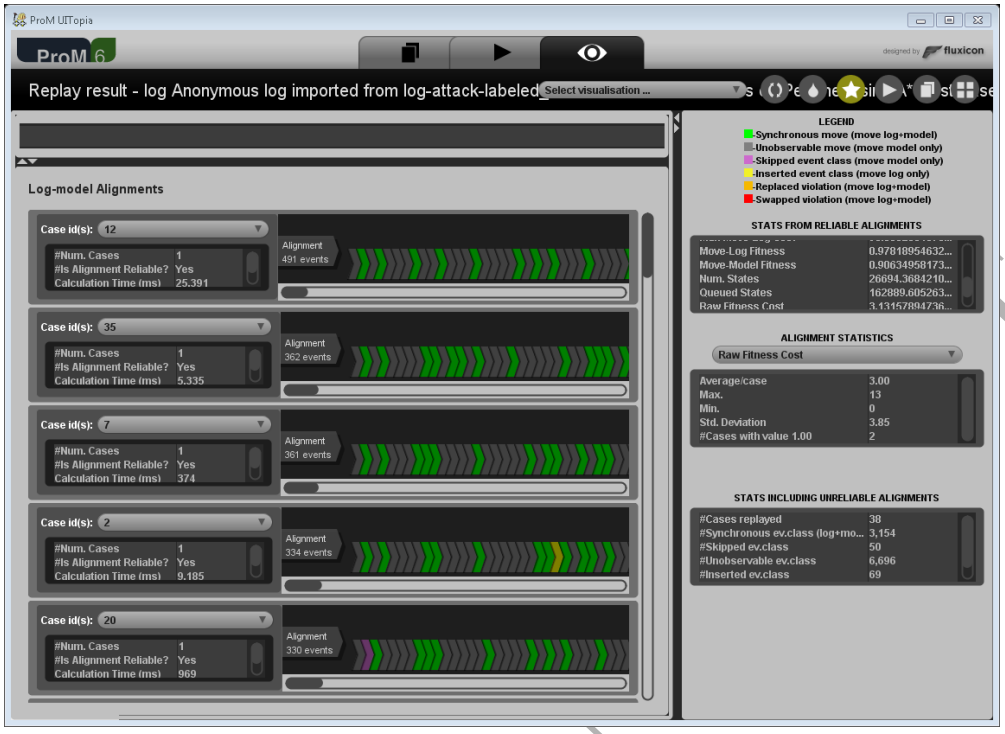

Figure 10: Example Screenshot results from conformance checking log replay in ProM 6.7 using the Replay a log on Petri net for Conformance Analysis plug-in.

	Move-Log Fitness		Trace Fitness	
	Positive	Negative	Positive	Negative
Positive	14	5	16	7
Negative	4	15	2	13

Table 8: Results of Conformance Check of the first Labelled Attack Dataset on its Model of Expected Behaviour.

We conduct the same anomaly detection process for our second dataset. There were a total of 20 cases in our labelled, pre-processed event log for the second dataset. Out of the total 20 cases, 9 of them contained attacks, as shown in Table 5. Three of the attacks occurred across multiple cases, cases . Similar to the first dataset, we compare the results of the conformance checking activity using the ProM process mining tool with our labelled event log for the second dataset. The results of our conformance checking activity on the model of expected behaviour for the second dataset is summarised in Table 9. As can be seen, both the Move-Log and Trace Fitness results were the same. Out of the 20 cases in the second dataset, both results identified 12 cases as anomalous, 9 of these are true positives (100% of attacks). No attacks remained undetected, however two cases were identified as false positives. Both the Move-Log and Trace Fitness results identified the same cases, case 9 and case 19, as false positives.

	Move-Log Fitness		Trace Fitness	
	Positive	Negative	Positive	Negative
Positive	9	3	9	3
Negative	0	8	0	8

Table 9: Results of Conformance Check of the second Labelled Attack Dataset on its Model of Expected Behaviour.

5. Discussion

In this section we discuss various areas of our process mining anomaly detection method, including the generation of ICS device logs, the transformation of device logs to event logs for process mining analysis, the development of a model of expected behaviour and conformance checking to detect anomalies in the ICS device logs.

5.1. Conformance Checking

The results of our conformance checking activity were primarily focused upon the detection of attacks conducted upon two datasets with commonly used ICS equipment (the Siemens S7-1200 PLCs and National Instruments NI cRIO-9074 PLC), by measuring the fitness of the pre-processed event log on the model of expected behaviour, and observing the anomalous behaviour as shown by inserted event classes, through using the *Replay a log on Petri net for Conformance Analysis* plugin for ProM 6.7. We found when conducting the conformance checking activity using the plugin in the ProM toolkit that there were limited options for algorithms choice, with the algorithm chosen being the α* algorithm. Using our process mining anomaly detection method, we successfully found cyber attacks which were conducted on the ICS equipment which were conducted on the PLCs throughout the experiment.

In the first dataset, We successfully found 14 of the 21 conducted attacks with Move-Log Fitness, and 16 out of the 21 attacks with Trace Fitness. The attacks which were not identified by the Move-Log Fitness were cases 3, 5, 7, and 36, and the cases that were not detected by the Trace Fitness were cases 3, and 36. In the second dataset, we successfully found all 9 of the attacks conducted, with both Move-Log and Trace Fitness. However, in this dataset, we found three false-positive cases, case 1, case 9, and case 19.

There were several cases which were identified as anomalous cases, however did not contain any of the cyber attacks which were conducted, outlined in Section 4.2.3. These detected anomalous cases are false-positives, where the case contained no attack, but were detected by our anomaly detection method. In the first dataset, The Move-Log fitness identified a total of 5 false-positives, cases 4, 13, 26, 28 and 33. The Trace fitness identifying 6 false positives, cases 4, 13, 21, 26, 27, 28, and 33. On further analysis of these false positive cases, we found that several of these false positives were caused not directly by the attacks, but due to the result of an attack. For example; cases 13, 26,and 33, detected as false positive by both the Move-Log Fitness and the Trace Fitness, have events changed due to the previous attack. Case 13 had an event where the “master reset” was unactivated, where in case 12, attack 7 was conducted, triggering a reset of all ICS devices. In case 26, the water tank system running status was set to “unactivated”, where in case 25, attack 15 was conducted in which the water tank was set to run in manual mode. Finally in case 33, the value at which the solenoid releases pressure was reset, after in case 32, attack 19 was conducted, where this value was changed. This indicates that some attacks may impact over one case, and as such false positives may be useful for further investigation.

Some of the false positives were not related to previous attacks, however. The false positives in cases 4, 21, and 27 were all unrelated to attacks. Case 4, for example, had an instance of one coloured pucks travelling around the wrong direction of the conveyor belt. This could have occurred due to the paddle controlling the direction of the pucks becoming jammed, or one of the coloured pucks travelling too close to a different coloured puck, and travelling through together. While this is anomalous behaviour, it was not caused by a cyber attack. Any “case”, or complete process instance, of this ICS system which contains events that does not fit the model of expected behaviour is a case which contains anomalous events for further manual investigation.

There were three cases that were detected as false-positives in the second dataset. These were cases 1, 9 and 19. These cases were detected as false positive in both the Move-log Fitness and the Trace Fitness. Case 1 was a false-positive that was unrelated to any attacks. Case 1 involved powering-on the PLC and HMI, which had some behaviour not accounted for in the model of expected behaviour, such as toggling the water pump and heating element before the industrial process started. Cases 9 and 19 were both detected as false-positives, which similar to the first dataset, may be related to previous attacks. The false positive in case 9 could be related to a previous attack, where case 8 involved an attack where the water heater was turned on. As the water temperature rose, above normal levels during the attack, over the next case (case 9) the water temperature lowered, which was not part of the modelled expected behaviour of the system. In case 19, a “pump on” event was detected, indicating that during the case the water pump was turned on while the pump was already running. This behaviour could be an impact from a previous attack which turned the water pump.

There are several challenges to implementing our anomaly detection technique in industry. Firstly, typical ICS devices do not log events using the "on change" method prescribed in our paper, and instead log events when an error event occurs to reduce device log size. This would result in our process mining based anomaly detection method being unable to be used. Our technique requires the generation of a model of expected behaviour of the ICS system. In large and complex ICS networks, such as power generation and transmission networks, while the model can be generated using process mining process discovery techniques, this model may become extremely complex, and difficult to interpret without domain knowledge of the systems. In addition, industrial systems can be comprised of large numbers of PLCs which have limited on-board storage, and are typically spread over large geographical areas. Our anomaly detection technique would need to be deployed in a centralised location, such as on a Historian, to gather and pre-process ICS device logs and identify anomalies. The anomaly detection technique presented in this paper is not conducted in real-time, and requires additional manual analysis to differentiate between cyber attacks and other anomalous events which do not fit the model of expected behaviour.

Using results from our first dataset, we found the program we used to conduct our experiment, ProM, reports that the conformance check of the pre-processed event log took 15516ms, or 15 seconds, to process the 3672 rows of data from our pre-processed event log from our attack dataset, consisting 8 hours of data. If we have any datasets that grow less than the approximate 3672 rows processed in 15 seconds, our anomaly detection method has the potential to run in semi-real time. As shown in Section 4.2.4, our method has been shown to scale linearly comparing data with time.

5.2. Industrial Control System Device Logs

To determine the most appropriate ICS device logging methods in Section 3.1, we used four commonly used programmable logic controller (PLC), the Siemens S7-1200. Theses PLCs were programmed and configured using STEP7 V11 SP2, and controlled by a HMI created using the WinCC V11 SP2, both part of the Siemens TIA portal V11 SP2. Each of these PLCs control an individual system which is part of an industrial process, described in Section 4.1. The HMI was configured with a total of six logging methods of various types, as outlined in Section 10, which were chosen to compare key features of logging methods and acquisition modes as outlined in Table 3. These logging methods include: one log with on change, one log with on change with deadbands, one log with cyclic in 5 second intervals, one log with cyclic with deadband in 5 second intervals, one log with cyclic acquisition mode in 1 second intervals, and finally one log with a combination of combination of both on change and deadband acquisition modes in 5 second intervals.

Logging Method	Acquisition Mode	Format
Circular	On Change	CSV
Circular	On Change with deadband	CSV
Ciruclar	Cyclic in 5 second intervals	CSV
Circular	Cyclic in 5 second intervals with deadband	CSV
Segmented Circular	Cyclic in 1 second intervals	CSV
Circular	Both in 5 second intervals with deadband	CSV

Table 10: Configured logging recorded by HMI for analysis of device logging methods.

Deadbands, which are constrains upon logging either by logging data within or outside of a specified range, were configured on three of the six logs, as outlined in Table 3. These deadbands are applied to tags which are expected to have high variance and produce large amount of logging entries, and are typically used to reduce logging sizes. The tags and assigned deadbands which were configured are outlined in Table 11.

The main differences between the device logs recorded by the HMI are the two major acquisition modes; Cyclic and On Change, and the use of deadbands to only record events which occur outside of the deadband limits described in 11. As described in Section 3.1, the Cyclic device log as shown in Table 12, records all tags simultaneously, showing the current value of the tags once every pre-defined interval. The On Change device log, as shown in Table 13, records far less information, only recording values as they change. This method of logging potentially does not log data which can be found in Cyclic device log, and may miss some valuable or useful information for conducting an analysis on the device logs. In addition, as can be seen Table 13, the On Change acquisition mode does not truly log data on

Configured Tag	Method	Low Limit	High Limit
Pipe_Read_Pipeline_Pressure	Outside	0.3	0.5
Pressure_Int	Outside	30	40
Tank_Level_1	Outside	0	81
Tank_Read_Tank_Level	Outside	0	81
Tank_Usage_Level	Outside	0	81

Table 11: Configured logging recorded by HMI for analysis of device logging methods.

change. It was expected that there would be a very large amount of data produced by a tag which has a high variance, for example, the Pipe_Read_Pipeline_Pressure(1) tag, which measures the current pressure of the "Reactor" system, where the value measured is highly accurate and would change frequently. However, by observing the device log, we see that the tag has actually been recorded once every one to three seconds, as can be seen under the "Timestring" and "Time_ms" columns. This means an device log recorded using the Cyclic data acquisition mode configured to record in one second intervals would produce more accurate data than a device log recorded using the On Change acquisition mode. One downfall to the Cyclic acquisition mode is the size of device logs. Due to logging all tags on an interval, the file size of the device logs are several times larger than a device log recorded with the On Change data acquisition mode. In a small system the file size differences may not be of concern, however in a large ICS network the management and storage of large device logs may factor in to the choice of logging method and acquisition mode. For our use, a device log using the Cyclic data acquisition mode configured with a 1 second interval, without any configured deadbands provides the most complete event logs, and would be the most suitable for use in process mining based activities. For our second datset, which used the National Instruments NI cRIO-9074 PLC, we captured data in the same format as the Siemens S7-1200 PLC so we can conduct our pre-processing and anomaly detection activities.

VarName	TimeString	VarValue	Validity	Time_ms
Pressure_Int(2)	7/16/15 9:31:11	0	1	42201396659
Pump_Off_SP_Int(2)	7/16/15 9:31:11	40	1	42201396659
Pipe_Solenoid _Open_Status(2)	7/16/15 9:31:11	-1	1	42201396659
Pipe_Read_Solenoid_Open_Cmd(2)	7/16/15 9:31:11	-1	1	42201396659
HMI_Pipe_Master_Mode(2)	7/16/15 9:31:11	-1	1	42201396659
Solenoid_On_SP_Int(2)	7/16/15 9:31:11	40	1	42201396659
Pump_On_SP_Int(2)	7/16/15 9:31:11	5	1	42201396659
Solenoid_Off_SP_Int(2)	7/16/15 9:31:11	30	1	42201396659
Pipe_Read_Solenoid_Mode(2)	7/16/15 9:31:11	0	1	42201396659
HMI_Pipe_Solenoid_Off_SP(2)	7/16/15 9:31:11	30	1	42201396659
HMI_Pipe_Solenoid_On_SP(2)	7/16/15 9:31:11	40	1	42201396659
HMI_Pipe_Pump_Off_SP(2)	7/16/15 9:31:11	40	1	42201396659
HMI_Pipe_Pump_On_SP(2)	7/16/15 9:31:11	5	1	42201396659
Pipe_Read_Pump_Mode(2)	7/16/15 9:31:11	0	1	42201396659
Pipe_Read_Pump_Run_Cmd(2)	7/16/15 9:31:11	0	1	42201396659
Pipe_Pump_Run_Status(2)	7/16/15 9:31:11	-1	1	42201396659
Pipe_Read_Pipeline_Pressure(2)	7/16/15 9:31:11	0.3130435	1	42201396659

Table 12: A section of a Cyclic device log recorded in 5 second intervals from a Siemens S7-1200 PLC controlling a "Reactor" system.

Several of the event logs recorded by the HMI, as listed in Table 3, were recorded with deadbands configured to record events which occur outside of the deadband high and low values outlined in 11. As discussed in Section 3.1, deadband are limits which can be placed on individual task recorded by the HMI. These values are typically used to reduce log sizes. We found during the experimentation that several of the tags continued to record events even

VarName	TimeString	VarValue	Validity	Time_ms
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:00	0.4173913	1	42201397922
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:02	0.3521739	1	42201397945
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:04	0.3	1	42201397968
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:06	0.3391304	1	42201397991
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:08	0.3782609	1	42201398015
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:10	0.4434783	1	42201398038
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:12	0.3521739	1	42201398061
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:15	0.3	1	42201398085
Pipe_Read_Pipeline_Pressure(1)	7/16/15 9:33:17	0.3652174	1	42201398108

Table 13: A section of an On Change device log without deadbands from a Siemens S7-1200 PLC controlling a "Reactor" system.

if the values were within the defined acceptable ranges. The use of deadbands can also be considered useful for process mining, where the removal of events which are considered acceptable, as in within the high and low values of a deadband, can potentially reduce the amount of pre-processing needed to conduct process mining based analysis. However, we found that even with the deadbands, there is still the requirement for a pre-processing stage involving these tags. With these issues, in combination with the requirement for logging events with as much information as possible, we came to a decision to not use any logs with deadbanding for pre-processing and the process mining based analysis. An additional common method of logging event in ICS and SCADA systems is error-based logging, where event logs are only recording events where an error is detected. Our event log pre-processing method would not be applicable to event logs which use this method of recording error events only. From testing and our case study, we have validated that the Cyclic data log with a low interval period records richest and most complete information, and is the most appropriate for use in process mining based analysis, and as such, should be used as the primary method of generating ICS device logs.

There are some challenges to implementing our presented anomaly detection method on logs from industry. First, ICS devices in industry may not have the level of verbose logging needed to be used for process mining based analysis. Second, to test the accuracy of our anomaly detection method, devices logs which have pre-known or labelled cyber-attacks or other anomalous behaviour are required. Finally, conducting this experiment on logs obtained from industry will require us to conduct attacks on running, production industrial control systems, or through injecting synthetic attacks in these industry logs.

5.3. Transforming Device Logs to Event Logs

ICS data logs initially are not suitable for process mining based analysis, and a pre-processing stage must be conducted. In our first dataset, Our ICS data logs transformation method removed data log status records which contained no value for the process mining based analysis. As outlined previously, These values included tags which were included in the deadbands, status records such as the changing of pressure inside of the reactor, or level of water in the water tank reservoir. By eliminating these during the pre-processing stage, there is the potential to lose supporting information related to any cyber attacks conducted on the ICS system, however, once an anomalous case or event has been found for further analysis, data log status records can be further examined.

5.3.1. Comparison with Intrusion Detection System

Typically, ICS and SCADA networks do not have IDS systems located on production networks, rather having an IDS located on a corporate network, which the ICS systems connect to through the use of specialised gateways. There has been some work extending existing IDS to the task of detecting anomalies on ICS networks, however these do not focus on the overall industrial process. To compare our anomaly detection method with a traditional IDS system, we have chosen the widely used IDS, "Snort". The Snort ICS was configured with the latest Community and Emerging Threats rule sets, as well as the rule set for S7 devices from the Digitalbond Quickdraw project. This Snort IDS was configured in a virtual machine, running on Windows 7 Enterprise SP1. The virtual machine was configured with

Ubuntu Linux, and had the latest release of Snort, version 2.9.9.0 installed and configured. The virtual machine was configured with 2 CPU cores, 4GB of RAM, and a 20GB HDD.

To conduct the test with the configured Snort IDS, we used the first dataset outlined in 4. The attack dataset consisted of PCAP files containing the network traffic of the ICS network with the conducted attacks. We replayed these PCAP files using Snort, and found that Snort configured with the latest in community, emerging threats and S7 rule sets, was unable to detect any of the process related attacks that were conducted on the S7-1200 PLCs. This is due to firstly, the attacks consisting of legitimate commands received by the IDS systems; and secondly, the Snort IDS being unable to interpret Profinet traffic used by the ICS devices.

6. Conclusion

Industrial control systems (ICS) and supervisory control and data acquisition (SCADA) systems are moving from dedicated communications to switched and routed corporate networks, exposing them to the internet and placing them at risk of cyber attack. The detection time for security breaches can be from months to years, where 66% of these breaches were discovered months after the attack was conducted [6]. Non-real time analysis, such as our presented anomaly detection method, as an additional defence for ICS networks has the potential to greatly reduce the time for discovery of an attack. To detect anomalous events, and potentially cyber attacks, we introduced a novel process mining anomaly detection method, where we analysed and compared various typical device logging methods used by ICS and SCADA systems, demonstrating that Cyclic data logs with low polling intervals are the most appropriate logging method. We introduced our method transforming ICS device logs to be suitable for process mining based analysis. We generated two models of expected behaviour of our two ICS systems. We conduct a process mining conformance checking activity using our pre-processed event logs from both the first and second datasets, and the generated models of expected behaviour. Finally, we validate our process mining anomaly detection method by conducting a case study where we use our method to successfully identify anomalies, including several cyber attacks, in two datasets of industrial control system device logs recorded from commonly used ICS devices, the Siemens S7-1200 PLCs, and the National Instruments NI cRIO-9074 PLC. From this case study, we have shown process mining can successfully be used to detect anomalous behaviour, and detect cyber attacks, on critical infrastructure devices such as ICS and SCADA equipment.

Future Work. From our work, we have outlined several areas of potential future work. First, our method of anomaly detection uses device logs gathered from ICS systems as an input for our pre-processing stage, however there may be additional data logs on information systems such as computers or servers on the same "Process Control Network", for example a HMI or workstation, which can be used to support these ICS device logs to detect anomalies and attacks. In addition, our method of pre-processing removes what we describe as *non-event records*. The removal of non-event records may also remove information useful for identifying anomalous behaviour, such as inconsistent logging of events. An area of future work is to is the inclusion of both additional data sources, and the analysis of the non-event records, in our pre-processing stage. Second, in our paper we formalise ICS device logs, and our method for the transformation of ICS devices logs to event logs suitable for use in pre-processing based analysis. As a result of this, an area of future work is the development of an automated system to transform ICS device logs to event logs suitable for process mining based analysis. Third, our anomaly detection technique, compared with other IDS systems such as Snort, does not work in real time. An area of future work is to improve our technique to process ICS device logs in real time, allowing for faster discovery of anomalous events. As our anomaly detection technique is improved for real time analysis, a study of the scalability of our proposed method is required. Fourth, we compare our anomaly detection technique with signature-based anomaly detection, specifically the IDS Snort. An area of future work is to compare our method with more intrusion detection methods, with attack datasets containing real-world attacks. Fifth, in our paper we generate a model of expected behaviour of the ICS systems. An area of future work is to expand our model of expected behaviour to include semantics for security goals, which can then be used as an input for a conformance checking activity.

Acknowledgements

This research was funded in part by ARC Linkage Project LP120200246, *Practical Cyber Security for Next Generation Power Transmission Networks*.

References

[1] A. Daneels, W. Salter, What is SCADA, in: D. Bulfone, A. Daneels (Eds.), 7th International Conference on Accelerator and Large Experimental Physics Control Systems, Comitato Conferenze ELETTRA, Trieste, Italy, 1999, pp. 339–343.
URL http://accelconf.web.cern.ch/AccelConf/ica99/papers/mc1i01.pdf

[2] R. van der Knijff, Control systems/scada forensics, what's the difference?, Digital Investigation 11 (3) (2014) 160–174, special Issue: Embedded Forensics. doi:http://dx.doi.org/10.1016/j.diin.2014.06.007.
URL http://www.sciencedirect.com/science/article/pii/S1742287614000814

[3] V. M. Igure, S. A. Laughter, R. D. Williams, Security issues in SCADA networks, Computers & Security 25 (7) (2006) 498–506. doi:http://dx.doi.org/10.1016/j.cose.2006.03.001.
URL http://www.sciencedirect.com/science/article/pii/S0167404806000514

[4] B. Miller, D. Rowe, A survey SCADA of and Critical Infrastructure incidents, in: Proceedings of the 1st Annual Conference on Research in Information Technology, RIIT '12, ACM, New York, NY, USA, 2012, pp. 51–56. doi:10.1145/2380790.2380805.
URL http://doi.acm.org/10.1145/2380790.2380805

[5] ICS-CERT, ICS-CERT Year in Review, https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2016_Final_S508C.pdf, Accessed on 22/09/2017 (2016).

[6] Verizon, 2013 data breach investigations report, http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2013_en_xg.pdf, Accessed on 18/04/2016 (2013).

[7] D. Hadžiosmanović, D. Bolzoni, P. H. Hartel, A log mining approach for process monitoring in SCADA, International Journal of Information Security 11 (4) (2012) 231–251. doi:10.1007/s10207-012-0163-8.
URL http://dx.doi.org/10.1007/s10207-012-0163-8

[8] J. Nivethan, M. Papa, A scada intrusion detection framework that incorporates process semantics, in: Proceedings of the 11th Annual Cyber and Information Security Research Conference, ACM, 2016, p. 6.

[9] R. Udd, M. Asplund, S. Nadjm-Tehrani, M. Kazemtabrizi, M. Ekstedt, Exploiting bro for intrusion detection in a scada system, in: Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security, ACM, 2016, pp. 44–51.

[10] ICS-CERT, Alert (IR-ALERT-H-16-056-01) cyber-attack against ukrainian critical infrastructure, https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01, Accessed on 18/04/2016.

[11] A. Partington, M. Wynn, S. Suriadi, C. Ouyang, J. Karnon, Process mining for clinical processes: A comparative analysis of four australian hospitals, ACM Trans. Manage. Inf. Syst. 5 (4) (2015) 19:1–19:18. doi:10.1145/2629446.
URL http://doi.acm.org/10.1145/2629446

[12] R. Mans, M. Schonenberg, M. Song, W. van der Aalst, P. Bakker, Application of process mining in healthcare a case study in a dutch hospital, in: A. Fred, J. Filipe, H. Gamboa (Eds.), Biomedical Engineering Systems and Technologies, Vol. 25 of Communications in Computer and Information Science, Springer Berlin Heidelberg, 2009, pp. 425–438. doi:10.1007/978-3-540-92219-3_32.
URL http://dx.doi.org/10.1007/978-3-540-92219-3_32

[13] S. Suriadi, R. Mans, M. Wynn, A. Partington, J. Karnon, Measuring patient flow variations: A cross-organisational process mining approach, in: C. Ouyang, J.-Y. Jung (Eds.), Asia Pacific Business Process Management, Vol. 181 of Lecture Notes in Business Information Processing, Springer International Publishing, 2014, pp. 43–58. doi:10.1007/978-3-319-08222-6_4.
URL http://dx.doi.org/10.1007/978-3-319-08222-6_4

[14] W. M. P. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, R. P. J. C. Bose, P. van den Brand, R. Brandtjen, J. C. A. M. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B. F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. W. Günther, A. Guzzo, P. Harmon, A. H. M. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. L. Rosa, F. M. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. M. Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. S. Pérez, R. S. Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. D. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, M. T. Wynn, Process mining manifesto, in: Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I, 2011, pp. 169–194. doi:10.1007/978-3-642-28108-2_19.
URL http://dx.doi.org/10.1007/978-3-642-28108-2_19

[15] W. M. P. van der Aalst, A. K. A. de Medeiros, Process mining and security: Detecting anomalous process executions and checking process conformance, Electronic Notes in Theoretical Computer Science 121 (2005) 3–21. doi:10.1016/j.entcs.2004.10.013.
URL http://dx.doi.org/10.1016/j.entcs.2004.10.013

[16] S. Suriadi, C. Ouyang, W. M. van der Aalst, A. H. ter Hofstede, Event interval analysis: Why do processes take time?, Decision Support Systems 79 (2015) 77 – 98. doi:http://dx.doi.org/10.1016/j.dss.2015.07.007.
URL http://www.sciencedirect.com/science/article/pii/S0167923615001396

[17] J. Nakatumba, W. van der Aalst, Analyzing resource behavior using process mining, in: S. Rinderle-Ma, S. Sadiq, F. Leymann (Eds.), Business Process Management Workshops, Vol. 43 of Lecture Notes in Business Information Processing, Springer Berlin Heidelberg, 2010, pp. 69–80. doi:10.1007/978-3-642-12186-9_8.
URL http://dx.doi.org/10.1007/978-3-642-12186-9_8

[18] H. Nguyen, M. Dumas, M. La Rosa, F. Maggi, S. Suriadi, Mining business process deviance: A quest for accuracy, in: R. Meersman, H. Panetto, T. Dillon, M. Missikoff, L. Liu, O. Pastor, A. Cuzzocrea, T. Sellis (Eds.), On the Move to Meaningful Internet Systems: OTM 2014 Conferences, Vol. 8841 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2014, pp. 436–445. doi:10.1007/978-3-662-45563-0_25.
URL http://dx.doi.org/10.1007/978-3-662-45563-0_25

[19] W. Van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, Knowledge and Data Engineering, IEEE Transactions on 16 (9) (2004) 1128–1142. doi:10.1109/TKDE.2004.47.
URL http://dx.doi.org/10.1109/TKDE.2004.47

[20] T. Murata, Petri nets: Properties, analysis and applications, Proceedings of the IEEE 77 (4) (1989) 541–580. doi:10.1109/5.24143.

[21] Object Management Group, Inc., Business Process Model and Notation, http://www.omg.org/spec/BPMN/, Accessed: 28/04/2016.

[22] The YAWL Foundation, YAWL: Yet Another Workflow Language, http://yawlfoundation.org/, Accessed: 28/04/2016.

[23] R. Accorsi, T. Stocker, G. Müller, On the exploitation of process mining for security audits: the process discovery case, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, Coimbra, Portugal, March 18-22, 2013, ACM, 2013, pp. 1462–1468. doi:10.1145/2480362.2480634.
URL http://doi.acm.org/10.1145/2480362.2480634

[24] A. A. de Medeiros, B. F. van Dongen, W. M. Van der Aalst, A. Weijters, Process mining: Extending the α-algorithm to mine short loops, Eindhoven University of Technology, Eindhoven 19 (2004) 24.

[25] L. Wen, W. M. P. van der Aalst, J. Wang, J. Sun, Mining process models with non-free-choice constructs, Data Mining and Knowledge Discovery 15 (2) (2007) 145–180. doi:10.1007/s10618-007-0065-y.
URL http://dx.doi.org/10.1007/s10618-007-0065-y

[26] L. Wen, J. Wang, J. Sun, Mining invisible tasks from event logs, in: Advances in Data and Web Management, Joint 9th Asia-Pacific Web Conference, APWeb 2007, and 8th International Conference, on Web-Age Information Management, WAIM 2007, Huang Shan, China, June 16-18, 2007, Proceedings, 2007, pp. 358–365. doi:10.1007/978-3-540-72524-4_38.
URL http://dx.doi.org/10.1007/978-3-540-72524-4_38

[27] J. Li, D. Liu, B. Yang, Process mining: Extending *alpha* -algorithm to mine duplicate tasks in process logs, in: Advances in Web and Network Technologies, and Information Management, APWeb/WAIM 2007 International Workshops: DBMAN 2007, WebETrends 2007, PAIS 2007 and ASWAN 2007, Huang Shan, China, June 16-18, 2007, Proceedings, 2007, pp. 396–407. doi:10.1007/978-3-540-72909-9_43.
URL http://dx.doi.org/10.1007/978-3-540-72909-9_43

[28] B. F. van Dongen, A. K. A. de Medeiros, L. Wen, Process mining: Overview and outlook of petri net discovery algorithms, Trans. Petri Nets and Other Models of Concurrency 2 (2009) 225–242. doi:10.1007/978-3-642-00899-3_13.
URL http://dx.doi.org/10.1007/978-3-642-00899-3_13

[29] A. Weijters, W. M. van Der Aalst, A. A. De Medeiros, Process mining with the heuristics miner-algorithm, Technische Universiteit Eindhoven, Tech. Rep. WP 166 (2006) 1–34.

[30] A. J. M. M. Weijters, J. T. S. Ribeiro, Flexible heuristics miner (FHM), in: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France, IEEE, 2011, pp. 310–317. doi:10.1109/CIDM.2011.5949453.
URL http://dx.doi.org/10.1109/CIDM.2011.5949453

[31] A. K. A. de Medeiros, A. J. M. M. Weijters, W. M. P. van der Aalst, Genetic process mining: an experimental evaluation, Data Mining and Knowledge Discovery 14 (2) (2007) 245–304. doi:10.1007/s10618-006-0061-7.
URL http://dx.doi.org/10.1007/s10618-006-0061-7

[32] R. Accorsi, T. Stocker, On the exploitation of process mining for security audits: the conformance checking case, in: Proceedings of the ACM Symposium on Applied Computing, SAC 2012, Riva, Trento, Italy, March 26-30, 2012, ACM, 2012, pp. 1709–1716. doi:10.1145/2245276.2232051.
URL http://doi.acm.org/10.1145/2245276.2232051

[33] H. Verbeek, J. Buijs, B. Van Dongen, W. M. van der Aalst, Prom 6: The process mining toolkit, Proc. of BPM Demonstration Track 615 (2010) 34–39.

[34] E. Colbert, D. Sullivan, S. Hutchinson, K. Renard, S. Smith, A process-oriented intrusion detection method for industrial control systems, in: International Conference on Cyber Warfare and Security, Academic Conferences International Limited, 2016, p. 497.

[35] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, A. Valdes, Using model-based intrusion detection for scada networks, in: Proceedings of the SCADA security scientific symposium, Vol. 46, Citeseer, 2007, pp. 1–12.

[36] M. Caselli, E. Zambon, J. Amann, R. Sommer, F. Kargl, Specification mining for intrusion detection in networked control systems, in: T. Holz, S. Savage (Eds.), 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016., USENIX Association, 2016, pp. 791–806.
URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/caselli

[37] M. Alizadeh, X. Lu, D. Fahland, N. Zannone, W. M. van der Aalst, Linking data and process perspectives for conformance analysis, Computers & Security 73 (2018) 172 – 193. doi:https://doi.org/10.1016/j.cose.2017.10.010.
URL http://www.sciencedirect.com/science/article/pii/S0167404817302262

[38] D. Myers, K. Radke, S. Suriadi, E. Foo, Process discovery for industrial control system cyber attack detection, in: IFIP International Conference on ICT Systems Security and Privacy Protection, Springer, 2017, pp. 61–75.

[39] D. Dolev, A. Yao, On the security of public key protocols, IEEE Transactions on information theory 29 (2) (1983) 198–208.

[40] N. R. Rodofile, T. Schmidt, S. T. Sherry, C. Djamaludin, K. Radke, E. Foo, Process control cyber-attacks and labelled datasets on s7comm critical infrastructure, in: Australasian Conference on Information Security and Privacy, Springer, 2017, pp. 452–459.

[41] C. Wakup, J. Desel, Analyzing a tcp/ip-protocol with process mining techniques, in: Business Process Management Workshops, Springer, 2014, pp. 353–364. doi:10.1007/978-3-319-15895-2_30.
URL http://dx.doi.org/10.1007/978-3-319-15895-2_30

David Myers is a researcher, PhD candidate and sessional academic in the Information Security Discipline of the Queensland University of Technology. David's current research areas include Process Mining and the use of Process Mining based techniques for network and information security, and the security of Industrial Control Systems and SCADA networks.

Dr. Suriadi is a Senior Research Fellow within the Information Systems School at the Queensland University of Technology. He has over 5 years of experience in conducting industry-academic research in the domain of information security and data analytics. He has driven successful data analytics research collaborations with Suncorp Insurance and two Queensland-based hospitals. These collaborations have resulted in a number of high quality publications at top forum, such as Decision Support Systems and Information Systems, as well as two research grants (the Australasian Health Services Innovation (AusHSI) Stimulus Grant and Accelerate Partnerships grant) with a combined value of approximately $470K.

Dr. Kenneth Radke is an Adjunct Associate Professor at the Queensland University of Technology. His research has been centred around the cyber security of electrical critical infrastructure, and a blend of cryptography and human computer interaction (HCI) design. Ken currently works managing international communication networks connected to industrial control systems.

Dr Foo's research interests can be broadly grouped into the field of secure network protocols with an active interest in network security applications. These include specific applications in the areas of wireless sensor networks security and security in industrial controls systems such as SCADA and the smart grid. Dr Foo has extensive experience with computer networking having worked and taught in this area for over 15 years. Dr Foo has also been responsible for the design and development of the QUT SCADA security research laboratory. The SCADA laboratory consists of multiple vendor systems that are run from industrial PLCs.

Anomaly Detection for Industrial Control Systems using Process Mining

David Myers, Suriadi Suriadi, Kenneth Radke, Ernest Foo

PII: S0167-4048(18)30679-5
DOI: 10.1016/j.cose.2018.06.002
Reference: COSE 1355

To appear in: *Computers & Security*

Received date: 25 September 2017
Revised date: 12 May 2018
Accepted date: 5 June 2018

Please cite this article as: David Myers, Suriadi Suriadi, Kenneth Radke, Ernest Foo, Anomaly Detection for Industrial Control Systems using Process Mining, *Computers & Security* (2018), doi: 10.1016/j.cose.2018.06.002