MELISSA: Towards automated detection of undesirable user actions in critical infrastructures

Citation for published version (APA):

Hadziosmanovic, D., Bolzoni, D., Hartel, P. H., & Etalle, S. (2011). MELISSA: Towards automated detection of undesirable user actions in critical infrastructures. In *Proceedings of the European Conference on Computer Network Defense (EC2ND, Gothenburg, Sweden, September 6-7, 2011)* (pp. 41-48). IEEE Computer Society. <https://doi.org/10.1109/EC2ND.2011.10>

DOI:

[10.1109/EC2ND.2011.10](https://doi.org/10.1109/EC2ND.2011.10)

Document status and date:

Published: 01/01/2011

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

MELISSA: Towards Automated Detection of Undesirable User Actions in Critical Infrastructures

Dina Hadžiosmanović*, Damiano Bolzoni*, Pieter Hartel*, Sandro Etalle*[†]

*University of Twente and [†]Eindhoven Technical University

The Netherlands

{dina.hadziosmanovic, damiano.bolzoni, pieter.hartel}@utwente.nl, sandro.etalles@tue.nl

Abstract—We address the detection of process-related threats in control systems used in critical infrastructures. Process-related threats take place when an attacker gains user access rights and performs actions, which look legitimate, but which are intended to disrupt the industrial process. We use logs to detect anomalous patterns of user actions on process control application. A preliminary case study suggests that our approach is effective in detecting anomalous events that might alter the regular process workflow.

I. INTRODUCTION

Industrial control systems (ICS) are used for controlling and monitoring industrial processes. Such systems can be found in critical infrastructures like: power plants and power grid systems, water, oil and gas distribution systems or building monitoring (e.g., airports, railway stations).

Although failures in the security or safety of critical infrastructures could impact people and thus damage industrial facilities, recent reports state that current critical infrastructures are not sufficiently protected against cyber threats. For example, according to Rantala [16], around 2,700 organisations dealing with critical infrastructures in the U.S. detected 13 million cybercrime incidents, suffered \$288 million of monetary loss and experienced around 150,000 hours of system downtime in 2005.

The increasing number of security incidents in ICS facilities is mainly due to the combination of technological and organizational weaknesses. In the past, ICS installations were separated from public networks, used proprietary software architectures and communication protocols. Due to the “security by obscurity” paradigm, the systems were less vulnerable to cyber attacks. Although keeping a segment of communication proprietary, ICS vendors nowadays increasingly use common communication protocols and commercial off-the-shelf software. Also, it is common to deploy remote connection mechanisms to ease the management during off-duty hours, and achieve nearly-unmanned operation.

Unfortunately, the stakeholders seldom enforce strong security policies. User credentials are often shared among users to ease day-to-day operations and are seldom updated, resulting in a lack of traceability. For example, a disgruntled (former) employee whose credentials were still valid has caused havoc in Queensland, Australia [18].

Due to these reasons, ICS facilities became more vulnerable to internal and external cyber attacks. Although companies reluctantly disclose incidents, there are several published cases where safety and security of ICS facilities were seriously endangered [17].

As “regular” computers systems, ICSs are susceptible to threats exploiting software vulnerabilities (e.g., protocol implementation, OS vulnerabilities). However, ICS facilities are also prone to process-related threats. These threats take place when an attacker using valid credentials performs actions, which look legitimate, but which are intended to alter the industrial process(es). Process-related threats also include situations when system users make an operational mistake at the application level, e.g. when a user inputs a wrong value (e.g., a highly oversized value) for a given device parameter and causes the failure of the process.

Traditional security countermeasures, such as intrusion detection systems, cannot detect, let alone mitigate, process-related threats. The anomalies generated by such threats are typically not reflected in communication patterns/data (e.g., injection of executable code to exploit a buffer overflow sent within network traffic data), and can only be detected by analysing data passed through the system at a higher semantic level. To understand the higher semantic level from network data, a protocol parser has to be used, such as in Bro [15]. Similarly, for host-based analyses, understanding the specific ICS application is crucial. At the moment of writing, there are no ICS protocol parsers publicly available. Thus we conclude that ICS network traffic cannot provide process semantic information. Other approaches for monitoring ICS behaviour include using field measurements or centralised ICS events as information resources. Field measurements represent raw values coming from field devices and thus are semantically too low level to provide an understanding of user actions. ICS event logs provide a complete, high-level view on the industrial process, capturing information about user activities, system changes in the field as well as system status updates.

Compared to both field measurements and network traffic, ICS events are more consistent as they represent a standard ICS output which is very similar in various vendor families. Therefore, we argue that ICS event logs can provide high

level process information and stakeholders could detect process-related threats by analysing ICS logs.

Problem: Even an ICS system used in a small installation generates thousands of potentially alarming log entries per day. Thus the size (and high dimensionality) of logs make manual inspection practically infeasible.

This is a relevant and challenging problem to tackle. It is relevant because process-related threats affect the security and safety of critical infrastructures, which in turn could endanger human life. It is challenging because in the past the analysis of system logs has been applied to other security domains (e.g., in [10]) but failed to deliver convincing results. We argue that ICS environments have, compared to other intranets, a less dynamic character and thus provide better chances for capturing relevant patterns.

Contributions: The main contributions of this paper are as follows:

- we propose an approach to detect process-related threats and build a tool to automate the analysis of ICS logs, which can be used to detect process-related threats,
- we have performed experiments to validate our approach using data from a real industrial installation.

Our tool leverages a well-known data mining algorithm to enumerate (in)frequent patterns within a given set. Despite being quite simple and straightforward, our benchmarks show that the chosen algorithm is effective in detecting anomalous user actions. We conduct our experiments on a type of ICS commonly referred to as “SCADA” (Supervisory Control and Data Acquisition). Therefore, in the rest of the paper we refer to SCADA environments only. These systems can typically be found in power, water and gas infrastructures. However, we believe that our approach can be easily adapted for the other types of ICS as well (e.g., distributed control systems - DCS).

II. PRELIMINARIES

In this section we explain how a typical SCADA system works. A SCADA system consists of two main domains: the process field and a control room (Figure 1). Large

Figure 1. SCADA system overview: control room and process field systems may have more than one control room. The network

Figure 2. SCADA system layered architecture

infrastructure binds the two domains together. SCADA users control the industrial process from the control room and are provided with a real-time overview of the process field device parameters (data about tank loads, pump statuses, temperatures, etc). Depending on the underlying process, SCADA systems differ from each other. For example, a power-related SCADA installation contains power switches and transformers while a water-related installation contains water pumps and valves. However, based on interviews with the stakeholders (who represent process engineers, operators and computer network experts from two different facilities), we argue that the computer systems controlling these processes behave in a similar way and our approach is thus transferrable between the domains.

System architecture: Despite the fact that there are different vendors, the system architectures in various SCADA systems are similar and the terminology is interchangeable. Figure 2 shows a typical SCADA layered architecture. Layer 1 consists of physical field devices, PLCs (Programmable Logic Controllers) and RTUs (Remote Terminal Units). The PLCs and RTUs are responsible for controlling the industrial process, receiving signals from the field devices and sending notifications to upper layers. Layer 2 consists of SCADA servers (Connectivity and Aspect Servers, together with a Domain Controller for user authentication) responsible for processing data from Layer 1 and presenting process changes to Layer 3. The various clients in Layer 3 represent SCADA users.

System users in process control: There are two kinds of SCADA users: engineers and operators. An engineer is responsible for managing object libraries and user interfaces, setting working ranges for devices, defining process set-points, writing automation scripts, etc. An operator monitors the system status and reacts to events, such as alarms (e.g., by opening a pump, closing a valve), so that the process runs correctly (Figure 1). Although industrial processes in various domains differ in the details, the user interaction with a SCADA system is broadly similar. Our stakeholders acknowledged that an engineer is a more powerful system user than an operator (e.g., an engineer writes scripts that

define process automation while operators usually only run the script). Also, operators perform actions that are predefined by engineers (e.g., an engineer defines pump speed range, while an operator works within the range only). This means that operator actions are security and safety constrained depending on the way the engineer implemented safety controls. By contrast, there is no mechanism that will ensure that engineer actions are safe for the process (e.g., an engineer can, by mistake, assign a capacity 10 times bigger than in reality to a tank, and thus shuts tank level alarms off).

System logs: System logs capture information about user activity information. Depending on the size of the facility, a SCADA system records thousands of events per day. The SCADA logs in different domains are stored in a similar format consisting of (1) basic attributes such as timestamp, subject of action, type of action, user and (2) context-specific attributes such as message description, subtype of action, etc. We acknowledge that an attacker could manipulate logs by sending false data to the control application which generates log (e.g., by controlling a field device) or by deleting existing log entries. However, these actions would require accessing higher privileges (and perhaps exploiting a system-related threat). In general, the most recent part of the logs is shown on the operator screens. Severe events are highlighted as alarms. However, alarms typically include only critical system statuses (e.g., tank level is very low) and cannot extract a more general view on the system behaviour.

In the next section we describe potential threats that relate to the user activity in the system.

III. SECURITY THREATS

We classify possible threats in two groups: system- and process-related. System-related threats typically exploit software/configuration vulnerabilities (e.g., a buffer overflow or a flaw in a communication protocol [4]). Such attacks are low level and typically occur at Layer 1 and Layer 2 of the SCADA architecture (Figure 2).

On the other hand, process-related threats exploit weak process controls, and imply that an attacker obtains (e.g., through social engineering) or has user access rights and issues legitimate SCADA commands to cause defects in the industrial process. We analyse process-related threats during a focus group session with stakeholders. Based on two possible places where a SCADA user can interact with the process, we distinguish two types of process-related threats: (1) threats that leverage access controls on field devices and (2) threats that leverage vulnerabilities of the centralised SCADA control application. The first type of threats typically results in sending bad data to the SCADA state estimation which can then produce errors in system state analysis [13]. The second type of threat includes scenarios of performing legitimate user actions (from the control application) that can have negative impact on the

Table I
IDENTIFIED PROCESS-RELATED THREATS AND THREAT EXAMPLES IN THE SYSTEM CONTEXT

Type of threat	Threat	Example
Scripting error	An engineer loads a script that causes errors in the system automation	An engineer inserts a wrong ratio of chemical components
Misconfiguration /functional	An engineer modifies a device parameter	Change capacity of a tank to prevent the alarming system from going off
Misconfiguration /functional	An engineer modifies auditing policy	Turn off all auditing
Misconfiguration /control	An engineer modifies the range of allowed actions for a specific device	A pump cannot be any longer stopped
Misconfiguration /control	An engineer modifies the system topology	Some devices become invisible, and thus inaccessible

process production or devices. In this paper we focus on the second type of process-related threats. These attacks are high level (occur via the process control application) and typically relate to Layer 3 (Figure 2).

A. Identifying process-related threats

Together with the stakeholders we analyse threats that include user activity in SCADA application control software. We define user activities as actions that are (1) performed by a signed-on user or (2) performed on a known user workstation. We analyse the threats that leverage legitimate system commands performed by a legitimate user, or by an illegitimate user who has managed to obtain valid credentials. Our focus is on the threats that can be triggered by a single user action (i.e., we do not analyse sequences of actions). Based on interviews with the stakeholders, we distinguish two threat scenarios, namely (1) an attacker impersonates a system user or (2) a legitimate system user makes an operational mistake.

To identify process-related threats, and the leveraged vulnerabilities, we analyse a real-life SCADA system controlling a water treatment facility located in the Netherlands. Table I shows a sample of identified threats. Generally, we distinguish two types of serious threats: scripting errors and misconfiguration. Both types of threats typically originate from the activity of engineers. The threats exploiting a scripting error imply writing (and loading) faulty process automation scripts or leveraging scripts already developed by system engineers. A misconfiguration implies forcing the settings of unsafe configurations.

A possible way of mitigating these vulnerabilities is to deploy a tool to analyse data resources from SCADA and thus detect malicious behaviours.

IV. APPROACH

We propose an approach to detect process-related threats based on an automated way of processing SCADA logs. Our

goal is to identify the most interesting events from the logs, and thus allow operators to focus on a set of potentially suspicious events than can be inspected manually. To this end we built a tool called MELISSA (Mining Event Logs for Intrusion in SCADA Systems).

The stakeholders acknowledge that anomalous events cannot be extracted by performing simple queries on logs. In general, less than 1% of the total log entries generated by a SCADA systems directly account for user activities (i.e., the event performed by a legitimately signed-on user or on one of the known user workstation). The remaining entries represent automated system responses which can and do consist of indirectly related user actions (e.g., system report on tank level change, communication loss, activation of a workplace). In practice, only understanding the context of latter events can help to extract potentially suspicious user actions. Thus manual analysis is infeasible. Also, a large amount of entries is timer-triggered (e.g., some events always appear with the same number of daily occurrences). Therefore, we believe that a pattern-based analysis of the system behaviour is suitable for the SCADA context.

The basic idea of our approach is that a frequent behaviour, over an extended period of time, is likely to be normal [3][14]. For example, a repeated error about a missing device becomes less interesting as time passes (because operators are supposed to tackle the problem without delay. If it is ignored, it can be considered as normal behaviour). By contrast, a rare event, in a semi-automated and stable environment as SCADA, is likely to be anomalous. For example, an engineer operating from a machine that is usually inactive outside the working hours is considered suspicious. Our analysis consists of pattern mining on SCADA logs. We use an algorithm for mining frequent patterns to identify the most and the least frequent (expected to be anomalous) patterns of system behaviours. Also, we include the process knowledge to evaluate the severity of specific log entries on the process (described in the Implementation section).

To mine logs we translate SCADA log entries into patterns. Figure 3 depicts the relation between a log entry, an itemset, an item and a pattern. Each unique log entry, with all attributes, represents a single itemset (Figure 3.A). A unique value of an attribute in the log entry represents one item. A support count is the number of log entries that contain the given itemset. Formally, if the support count of an itemset I exceeds a predefined minimum support count threshold, then I is a pattern [8]. In Figure 3, log entries 2 and 3 are the same, thus the corresponding pattern has the support count 2 (Figure 3.B).

Input data for analysis: We leverage the knowledge of the stakeholders about their processes to choose the subset of attributes that is expected to provide the most relevant information gain for the analysis. We base our selection on the threats we describe in Section III-A: we need information about the time at which the event occurs and on which

Figure 3. Log translation: A) mapping log entries into itemsets and items, B) mapping itemsets into patterns

Table II
EXAMPLE OF A SIMPLIFIED SCADA LOG

W. shift	Aspect of action	Type of action	Object path	User account	SCADA node
2	-	Process Simple Event	-	-	CS01
2	Layout change	Configuration change	Root/./ layout	Engineer1	EN01
1	-	Operator action	Plant1/ ./ tanks	Operator2	OP03

system/node, about the user that performed the action and about the nature of the action. We have performed some preliminary tests to narrow down the list of potentially interesting attributes, and to eliminate redundant (e.g., user full name) and unstructured (e.g., object description) attributes.

Our final set consists of 6 nominal attributes: *Working shift*, *Aspect of action*, *Type of action*, *Object path*, *User account* and *SCADA node*. A *SCADA node* represents a computer that sends event details to the log. In our case, there are 8 different nodes. The attribute *Type of action* takes one out of 12 nominal values. This attribute describes the general type of action, such as: operator action, layout configuration change, network message, etc. For types of action which are performed by users, the attribute *Aspect of action* is applicable. This attribute takes one out of 6 nominal values in the log and details the character of the user action, such as: change of workplace layout, change in workplace profile, etc. The attribute *Object path* provides information about the location of the object device (e.g., main site/street1/tanks). The attribute *User account* represents the username of the signed-on user. Table II represents a sample of the analysed log.

V. ARCHITECTURE

MELISSA consists of two interacting components: the Data Preparator (DP) and the Pattern Engine (PE). Figure 4 depicts MELISSA and its internal components.

Figure 4. MELISSA architecture

Data Preparator: We perform data aggregation (e.g., variance reduction) and transformation (e.g., value coding) on the dataset to get a suitable data format for pattern mining. We describe performed operation in the Implementation section.

Pattern Engine: The PE first runs the algorithm for mining frequent patterns over log and outputs an ordered list of patterns based on the frequency of the occurrence. We then use the severity weights, defined by the stakeholders, in combination with the frequency to perform a fine-grained re-ordering of the output list. This way we differentiate patterns that occur with the same frequency but may result with different consequences on the process.

To select the most suitable algorithm for mining frequent patterns, we identified a list of required features. The requirements are as follows: (1) maximal pattern mining, (2) scalability, (3) selection of interesting events based on the absolute support count.

Maximal pattern mining There are different types of pattern mining (e.g., closed, complete, maximum). An itemset can be frequent but not (necessarily) interesting and useful for stakeholders in a specific context. For us, the stakeholders agreed that no subset of attributes carries enough semantics to distinguish between anomalous and normal events. For example, it is not sufficient to describe an event with only two attributes (e.g., itemset attributes {type of action, user account}; itemset instance {Operator action, Operator 2}). Therefore, we set a requirement that the algorithm has to deliver output patterns which consists of as many attributes as possible. This type of mining is in the data mining terminology referred as mining maximal patterns [8]. In our context, a maximal itemset is one log entry consisting of all applicable log attributes.

Scalability For the cases when the same plant setup is running for years, we might want to run the tool continuously, and receive events as they occur. Thus, the tool needs to scale well when processing logs that may consist of years of plant work. The tool can then leverage the knowledge of past behaviours to update the top patterns and detect anomalies. Also, the speed of processing is important as operators must take immediate action in case of an alarm. There are two main types of mining algorithms: 1) algorithms that use candidate generation (e.g., in [1]) and 2) algorithms that do not use candidate generation [8].

We expect our log size to grow up to several million entries (e.g., around 2,500,000 entries correspond to the

stakeholder’s annual system logs). Based on benchmark results of the FIMI workshop [6], we choose to use an algorithm that does not use candidate generation to comply with the scalability and speed requirements.

Selection of interesting events based on absolute support count To distinguish between interesting and uninteresting itemsets, algorithms use the concept of “cut off” parameter. For example, some algorithms use absolute minimum support count (e.g., consider an itemset frequent if it appears at least 5 times in the dataset) while others use the relative minimum support (e.g., take itemsets which appear in 10% of total dataset entries). Some algorithms use top k ranking of patterns (e.g., extract top 5 ranked patterns). In our context, the output of the algorithm is then inspected by security experts. Thus, to improve the usability, we decide to use an algorithm which implements absolute support count and ranking. We determine the final “cut off” parameter with stakeholders (discussed in Section VI).

An algorithm that meets most of our requirements is presented by Grahne et al. [7].

Implementation: We have implemented a prototype of MELISSA using Java. Data aggregation operations gather and summarize data for easier analysis. We transform the *Timestamp* attribute to represent usual working shifts in the company. This way we aggregate a timeseries attribute into a 3-value discrete format that is more suitable for mining workload patterns. In our case, “working shift 1” covers all events occurring between 00:00 and 08:59hrs. “Working shift 2” includes events occurring between 09:00 and 16:59hrs. “Working shift 3” includes events occurring between 17:00 and 23:59hrs.

Also, we aggregate the attribute *Object path*. This attribute is in the format of structured text and represents a hierarchical tree of locations in the plant (both functional and geographical). In practice, the values of this attribute represent the “address path” of a device where the event has taken place or, in case of a configuration change, the system path of the change. The last substring of the path represents the name of a device (e.g., plant1/street1/pumps/pump3). To aggregate values, we take the substring of the tree path with up to and excluding the leaves of the tree. This way we semantically group together devices which are on the same location (or configuration paths) and thus aggregate the original attribute from 170 down to 36 nominal values.

Finally, for all nominal attributes in the dataset we code distinct numerical values as our algorithm only accepts numerical values.

In the Pattern Engine we use an algorithm for mining maximal frequent patterns proposed in [7]. We then use severity weights for the fine-grained re-ordering of patterns. The severity weights are derived from the ranking that is compiled by the stakeholders. For this, we asked the stakeholders to select a set of attributes. The stakeholders selected three attributes: *Type of action*, *SCADA node* and

Aspect of action and compiled the ranking of the values for each attribute (e.g., for attribute *Type of action*: 1. Aspect-Directory, 2. Network message, 3. Operation, etc. Here, the order of the value implies how severe is that specific type of action for the process). We run several experiments to generate different PE outputs using the selected attributes, and then submitted the results to the stakeholders. The stakeholders selected the attribute *Type of action* as the one whose ranking performed the most useful results within the extracted patterns. Thus we perform the final fine-grained re-ordering based on the severity weights of this attribute.

VI. BENCHMARKS

To evaluate the effectiveness of our approach we collected a dataset of logs generated by the SCADA system of the stakeholders, which processes waste, surface and drinking water. The 101,025 log entries were collected during a 14 day period, and each log entry consists of at most 12 attributes. The logs were captured with the default audit set up of the SCADA system that collects events continuously through time.

We use the subset of 6 log attributes that consist of 69 unique items. Since we aim at identifying the least frequent patterns, our minimal support count is 1. This means that each unique event which occurred at least once represents a pattern.

Testing MELISSA: At the time of logging, there were no known security incidents in the log. Therefore, lacking the notion of ground truth, we can not perform the typical detector validation. Nevertheless, the goal of the testing is to see if we can detect unexpected (but security interesting) user actions from real data. To keep the data clean and realistic, we thus choose not to add synthetic entries to the log.

As a proof of concept, we run our analysis in offline mode. This means that a user runs a “day after” analysis. For example, each day the user receives up to 20 least frequent events from the day before (normally, in the stakeholder’s facility under analysis, a user gets approximately 7,000 unclassified events per day, so a reduction to 20 is significant). This approach can detect silent mimicry attacks as operators have a daily overview of events and can spot unusually infrequent user actions spread over a long period of time (e.g., unplanned configuration changes [18]). The disadvantage is that the system does not use summarized knowledge of past behaviour (i.e., the “knowledge window” is 1 day).

We decide to run the analysis offline because:

- we were provided with only two weeks of system logs;
- we cannot claim that these two weeks represent a complete set of behaviours that occur in the facility through a year;
- water-related systems are considered as slow processes (the consequences of actions are delayed - e.g., it takes

WEIGHTED VALUE	SUPPORT	SHIFT	ASPECT NAME	TYPE OF ACTION	OBJECT PATH	USER ACCOUNT	SCADA NODE	
0.25	1	1	Operator 2 Note	AspectDirectory	Root/Plant2/tanks	Operator 2	OP04	
0.5	1	1	none	Network Messages	none	none	EN03	X A1
0.5	1	1	none	Network Messages	none	none	OP03	
0.5	1	1	none	Network Messages	none	none	OP01	
0.5	1	2	none	Network Messages	none	none	OP01	
0.5	1	3	none	Operation	none	none	OP01	
0.5	1	1	none	Operation	none	none	OP01	
0.75	1	2	Control Module	Operator 2Action	Plant2/_/Ozonisation	Operator 2	OP03	
0.75	1	1	Event List	Operator 2Action	Building/Street1	Operator 2	OP04	
1.5	2	2	none	Network Messages	none	none	OP03	
1.5	2	2	none	Network Messages	none	none	CS01	X A2
1.5	2	3	none	Network Messages	none	none	OP01	
1.5	2	2	none	Network Messages	none	none	OP04	
1.5	2	2	none	Operation	none	none	CS01	X A3
1.75	1	1	Name	AuditEvent_Acknowledge	Admin/node/	Operator 2	OP04	
1.75	1	1	Name	AuditEvent_Acknowledge	Building A/_/pump	Operator 2	OP04	
1.75	1	1	Name	AuditEvent_Acknowledge	Admin/node/	Operator 2	OP04	
1.75	1	1	Name	AuditEvent_Acknowledge	Admin/node/	Operator 2	OP04	
1.75	1	1	Name	AuditEvent_Acknowledge	Root ABC/_/pumps	Operator 2	OP04	

Figure 5. MELISSA testing, output of the day 4

several hours to overload a tank even while pumping at maximum speed), thus we can afford to run the analysis with a delay.

MELISSA found 486 unique patterns from the 14 days long SCADA log. The number of unique patterns per day varies from 12 to 79. Also, the support count per day per pattern varies from 1 to 1,151. According to our stakeholders, an acceptable level of usability is that they receive up to 20 events per day for manual inspection, with the exception that all events with a support count of 1 should be reported. We use these requirements to set the threshold for extracting the most interesting events.

We first summarize the results after 14 days of inspection. After applying the threshold on the whole dataset, approximately 198 events (represented in 131 patterns) are labeled for inspection. During the daily inspections, the stakeholders label 20 patterns as suspicious. After having collected additional information about the context, the stakeholders finally label 3 patterns as anomalous. These 3 patterns occur in 5 events.

We now describe the context of the patterns which were labelled as anomalous. All anomalous patterns occur on day 4. Figure 5 represents a projection of the pattern analysis from this day. The table consists of 8 columns. The first column represents the final PE output value for the specific pattern (the value is a combination of pattern support and severity weight). The second column represents the pattern support count. The remaining columns represent the attributes used in the analysis. The wavy horizontal line represents the border between interesting and uninteresting patterns as decided by the stakeholders (maximum 20 events per day). On the righthand side of table, the stakeholders labeled each pattern as either normal or anomalous (A1...A3). For anomalous patterns, circles imply why the pattern is unusual.

Anomalous pattern A1 occurred only once (support count is 1). Node EN03 represents an engineering workstation. Shift 1 represents the night shift. For the stakeholders A1 is anomalous because engineers are expected to work only

Table III
MELISSA RESULTS: SIZE OF INPUT LOGS, SIZE OF TOOL OUTPUT,
NUMBER OF FIRST AND SECOND STAKEHOLDERS' INSPECTION

	Full log	For inspection	Suspicious	Anomalous
number of events	101,025	198	23	5
number of patterns	486	131	20	3

during day shifts. Inspecting the complete log we found that, except this event, all activities performed by engineers or on engineering workstations did occur during day shifts only. After a thorough internal inspection, the stakeholders found a software emulator with a faulty automation script that remotely attempted to connect to the EN03 engineering workstation. This entry represents a trace of a scripting error by an engineer (thus an operational mistake) which could have effect on system performance, as other actions could depend on it.

Anomalous patterns A2 and A3 occur twice. Node CS01 represents the primary Connectivity Server. *Network message* item typically reports problems in the network communications. *Operation* item reports negative system responses to a user action (such as input expression error messages). The stakeholders evaluate patterns A2 and A3 as anomalous because these patterns reflect network and operational errors on the main connection backbone node (CS01). After a thorough internal inspection, the stakeholders found out that all events from these two patterns occurred in the same minute of day 4. User Engineer 1 was logged in on CS01 during the time these errors happened. The stakeholders assume that Engineer 1 inserted a value which triggered an overflow in a device cache, which in turn generated an error report from the system. We verify that this is the only case, over two weeks of operations, that error messages were triggered on Connectivity Servers. The stakeholders classify these patterns as misconfiguration threats where user triggered cache overflows by inserting unexpected values.

Usability: Table III summarizes the output of the performed log analysis through different phases. To inspect system behaviour in a currently running SCADA system, the users would have to look at individual events (a few thousand per day). By transferring the level of analysis to patterns, instead of individual events, we help stakeholders in aggregating log information. To discard a large number of uninteresting patterns, we perform frequency pattern analysis. With the suggested “cut off” threshold, our stakeholders receive for inspection 131 unique patterns in 14 days. The number of patterns per day varies, but on average it less than 10. Finally, after context analysis of suspicious patterns (i.e., an additional round of analyses), we estimate that the user had to inspect in average 11 patterns per day.

System performance: Testing has been performed on a machine with an Intel Core 2 CPU at 2.4GHz and 2Gb of

Table IV
SYSTEM PERFORMANCE RESULTS AND ESTIMATION OF PROCESSING
ANNUAL LOGS

Dataset information	SCADA log	“Accidents”	Estimated annual SCADA logs
number of instances	101,025	1,000,000	2,500,000
total number of items	69	500	70
avg size of itemsets	6	45	6
Data Preparator (s)	22.7	does not apply	1,080
Pattern mining (s)	0.97	100	200
Total MELISSA processing time (s)	23.6	does not apply	1,280

memory. Table IV shows runtime results of testing. The table consists of three columns. The first column shows the results of our testing on SCADA system logs. The second column shows benchmark results of the pattern mining algorithm by Grahne et al. [7] on the “Accidents” dataset [11]. We use these results to estimate the runtime of the expected size of system logs over a year time (shown in the third column). The complexity of the preprocessing is $O(n)$. Scalability of the used mining frequent patterns algorithm (in PE) is discussed in [7]. To estimate MELISSA’s performances on an annual SCADA log, we consider benchmarks of the pattern mining algorithm of [7] on the “Accidents” dataset. We argue that this dataset is more complex than the dataset we use, due to the higher number of attributes. Thus, we take the results from [7] as our worst case.

To summarize, we estimate that our tool would preprocess and mine patterns in size of approximately one year of work in the stakeholder facility in around 22 minutes.

VII. RELATED WORK

To detect anomalous behaviour in SCADA systems, authors use approaches based on inspecting network traffic [2] validating protocol specifications [4] and analysing data readings [13]. Process-related attacks typically cannot be detected by observing network traffic or protocol specifications in the system. We argue that to detect such attacks one needs to analyse data passing through the system [2][5] and include a semantic understanding of user actions. Bigham et al. [5] use periodical snapshots of power load readings in a power grid system to detect if a specific load snapshot significantly varies from expected proportions. This approach is efficient because it reflects the situation in the process in a case of an attack. However, data readings (such as power loads) give a low-level view on the process and do not provide user traceability data.

Several researches explore pattern mining of various logs for security purposes (e.g., alarm logs in [10][14], system calls in [12], event logs in [9]). These authors use pattern mining on burst of alarms to build episode rules. However, pattern mining can sometimes produce irrelevant and redundant patterns, as shown in [10]. We use pattern mining algorithms to extract the most and the least frequent event patterns from SCADA log.

To the best of our knowledge, only Balducelli et al. [2] analyse SCADA logs to detect unusual behaviour. There, the authors use case-base reasoning to find sequences of events that do not match sequences of normal behaviour (from the database of known cases). The authors analyse sequences of log events that originate from a simulated testbed environment. In contrast, we analyse individual logs from a real SCADA facility.

VIII. CONCLUSION AND FUTURE WORK

We analyse process-related threats that occur in the computer systems used in critical infrastructures. Such threats take place when an attacker manages to gain valid user credentials and performs actions to alter/disrupt a targeted industrial process, or when a legitimate user makes an operational mistake and causes a process failure.

Currently no control (e.g., monitoring tools) is available to mitigate process-related threats. To detect process-related threats, logs could be analysed. These logs hold critical information for incident identification, such as user activities and process status. However, system logs are rarely processed due to 1) the large number of entries generated daily by systems and 2) a general lack of the security skills and resources (time).

We propose an analysis tool that extracts non-frequent patterns, which are expected to be the result of an anomalous events such a undesirable user actions. We benchmarked the tool with real logs from a water treatment facility. Although no real security incident occurred in the log we took into account, at least five events were labelled by the stakeholders as anomalous. A limitation of our approach is that the attacker could evade the detection by repeating the same command a number of times. We plan to address this limitation by increasing the “knowledge window” for learning common relations between the frequencies of performed actions for a longer period of time. Also, as future work, we plan to expand our tool to address anomalous sequences of actions, rather than single events/operations.

REFERENCES

- [1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB'94, Proc. 20th International Conference on VLDB, September 12-15, 1994, Santiago de Chile, Chile*, pages 487–499. Morgan Kaufmann, 1994.
- [2] C. Balducelli, L. Lavallo, and G. Vicoli. Novelty detection and management to safeguard information-intensive critical infrastructures. *Int. J. Emergency Management*, 4(1):88–103, 2007.
- [3] K. Begnum and M. Burgess. Principle components and importance ranking of distributed anomalies. *Machine Learning*, 58:217–230, February 2005.
- [4] C. Bellettini and J. Rrushi. Vulnerability analysis of SCADA protocol binaries through detection of memory access taintedness. In LTC John Hill, editor, *Proc. 8th IEEE SMC Information Assurance Workshop*, pages 341–348. IEEE Press, 2007.
- [5] J. Bigham, D. Gamez, and N. Lu. Safeguarding SCADA systems with anomaly detection. In *MMMACNS '03: Proc. 2nd International Workshop on Mathematical Methods, Models and Architectures for Computer Network Security*, LNCS 2776, pages 171–182. Springer Verlag, 2003.
- [6] B. Goethals and M. Zaki, editors. *FIMI '03, Frequent Itemset Mining Implementations, Florida, USA*, volume 90 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.
- [7] G. Grahne and J. Zhu. Fast algorithms for frequent itemset mining using FP-Trees. *IEEE Transactions on Knowledge and Data Engineering*, 17:1347–1362, 2005.
- [8] J. Han and M. Kamber. *Data mining concepts and techniques*. Morgan Kaufmann Publishers, 2 pap edition, 2006.
- [9] J. L. Hellerstein, S. Ma, and C.-S. Perng. Discovering actionable patterns in event data. *IBM Syst. J.*, 41:475–493, July 2002.
- [10] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proc. 8th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '02*, pages 366–375, New York, NY, USA, 2002. ACM.
- [11] T. Brijs K. Geurts, G. Wets and K. Vanhoof. Profiling high frequency accident locations using association rules. In *Proc. 82nd Annual Transportation Research Board, Washington DC (USA)*, pages 123–130. Transportation Research Board, 2003.
- [12] W. Lee and S. Stolfo. Data mining approaches for intrusion detection. In *Proc. 7th conference on USENIX Security Symposium - Volume 7*, pages 6–6, Berkeley, CA, USA, 1998. USENIX Association.
- [13] Y. Liu, P. Ning, and M. Reiter. False data injection attacks against state estimation in electric power grids. In *Proc. 16th ACM conference on Computer and communications security, CCS '09*, pages 21–32, New York, NY, USA, 2009. ACM.
- [14] S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz. A data mining analysis of RTID alarms. *Comput. Netw.*, 34:571–577, October 2000.
- [15] V. Paxson. Bro: a system for detecting network intruders in real-time. *Comput. Netw.*, 31:2435–2463, December 1999.
- [16] R. Rantala. Cybercrime against businesses. Technical report, U.S. Dept. of Justice, Office of Justice Programs, Bureau of Justice Statistics, Washington, D.C., 2004.
- [17] A. Rege-Patwardhan. Cybercrimes against critical infrastructures: a study of online criminal organization and techniques. *Criminal Justice Studies*, 22(3):261–271, Sep 2009.
- [18] J. Slay and M. Miller. Lessons learned from the Maroochy water breach. In *Critical Infrastructure Protection*, pages 73–82, 2007.