Towards Securing SCADA Systems Against Process-Related Threats

Dina Hadžiosmanović, Damiano Bolzoni and Pieter Hartel
University of Twente, The Netherlands

***Abstract*—We propose a tool-assisted approach to address process-related threats on SCADA systems. Process-related threats have not been addressed before in a systematic manner.**

Our approach consists of two steps: threat analysis and threat mitigation.

For the threat analysis, we combine two methodologies (PHEA and HAZOP) to systematically identify process-related threats.

The threat mitigation is supported by our tool, MELISSA, that helps to detect incidents (attacks or user mistakes). MELISSA uses SCADA system logs and visualization techniques to highlight potential incidents.

A preliminary case study suggests that our approach is effective in detecting anomalous events that might alter the regular SCADA process work-flow.

***Index Terms*—SCADA, security, intrusion detection, visualization.**

I. Introduction

SCADA (Supervisory Control and Data Acquisition) systems are computer systems commonly used for monitoring equipment and controlling processes in industrial, transport, transmission and distribution facilities. Examples of such systems are: power plants and power grid systems, water and gas distribution systems, building monitoring (airports, railway stations), production systems for food, cars, ships and other products.

Many facilities that employ SCADA systems are critical infrastructures. Although failures in the security or safety of critical infrastructures highly impact people and can produce serious damages to industrial facilities, recent reports state that current critical infrastructures are not protected properly. For example, according to Rantala [1], around 2700 critical infrastructures in the U.S. detected 13 million cybercrime incidents, faced $288 million of monetary loss and experienced around 152 200 hours of system downtime in 2005. To understand why this is so, we must look at the history. In the past, SCADA systems were separated from public infrastructures, had dedicated network architecture and used proprietary communication protocols. As a result, the systems were isolated and not vulnerable to network attacks. Nowadays, an increasing number of interconnected services (e.g., billing systems, maintenance, resource management) make SCADA systems vulnerable to internal and external attacks. Although companies reluctantly disclose incidents, there are several published cases where safety and security of SCADA systems were seriously endangered [2], [3], [4], [5].

1) Problem: The standard approaches for the detection of malicious behaviour (e.g. monitoring network traffic [6], inspecting protocol specifications [7]) cannot address process-related threats in SCADA systems. These threats take place when an attacker gains user access rights and performs actions, which look legitimate, but which are intended to alter and/or disrupt the industrial process in SCADA. Process-related threats also include situations when a system user makes an operational mistake (e.g, a user inserts a highly oversized or an undersized value for device parameter and causes process to fail).

2) Contribution: We propose a tool-assisted approach to address process-related threats. The approach consists of:

- a combined methodology (PHEA and HAZOP [8]) to identify process-related threats caused by the activity of SCADA engineers;
- analysis of SCADA logs generated by a real-life facility to detect potential incidents;
- visualization techniques to present the analysis of SCADA logs.

II. Securing a SCADA system

We first introduce the main concepts that we use in the rest of the paper.

A threat is either (1) an intention and method targeted at the exploitation of a vulnerability or (2) a situation and method that may accidentally trigger a vulnerability. A vulnerability is a flaw or weakness in system procedures, design, implementation, or internal controls that could be exercised (accidentally triggered or intentionally exploited) and result in a security breach or a violation of the systems security policy [9]. We call a target any system asset that may be the object of an attack. A system asset is any tangible or intangible thing that has value to an organization [10].

An instance of a threat exploiting a vulnerability over a system target is an incident. An incident may leave traces in various data records. An anomaly represents an unusual/unexpected occurrence in a data record that originates from either an incident or a benign exception in system behaviour.

Securing a computer system from a specific threat implies taking measures to detect anomalies that originate from incidents. These measures usually consist of applying various techniques to analyse data sources (such as network data) and deploy tools to protect the system (such as intrusion detection).

We argue that, due to its specific character, threats, vulnerabilities and protection controls in SCADA systems differ from regular computer systems and should be addressed accordingly. Figure 1 gives an example of a threat, a vulnerability

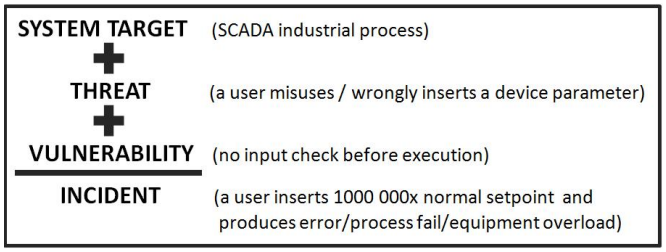

Fig. 1. Example of an incident in SCADA environment. An incident is generated by a combination of: a system target, a threat and a vulnerability.

and an incident in the SCADA context and how they relate. SCADA systems are, as any other computer systems, prone to threats. However, SCADA environments are also prone to a specific class of threats. These threats, we call them process-related threats, bypass traditional IT security controls as they materialise by issuing legitimate commands to disrupt or alter SCADA process, without violating communication patterns (e.g., network traffic).

To the best of our knowledge, research has not comprehensively addressed the detection of malicious behaviours in SCADA systems at a high semantic level, when performed commands are legitimate but the impact on the system is still negative.

We take a two-step approach (Figure 2) to address process-related threats in SCADA systems: (1) Threat and vulnerability analysis and (2) Mitigation controls. During the first step (Section IV) we assess a real working SCADA deployment and, using a combination of two methodologies (PHEA and HAZOP), systematically identify process-related threats. In Section IV-B1 we describe the performed methodologies in detail. In the second step (Sections V to VII) we design mitigation controls to highlight suspicious situations. We build a semi-automated tool (MELISSA: Mining Event Logs for Intrusions in SCADA Systems) to analyse logs. We use visualization techniques to present the content of logs and detect potentially malicious behaviour.

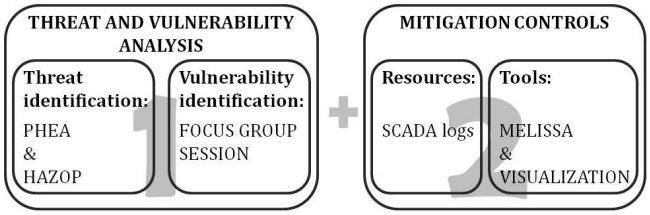

Fig. 2. A two-step approach to address process-related threats

III. System description

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 3). Large systems may have more than one control room. The network infrastructure binds the two domains together. SCADA users control the industrial process from the control room using computers that provide a real-time overview of the process field with 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 process field consists of power switches and transformers while a water-related process field consists of water pumps and valves. However, based on interviews with stakeholders who belong to various business sectors, we argue that the computer systems controlling these processes behave in a similar way.

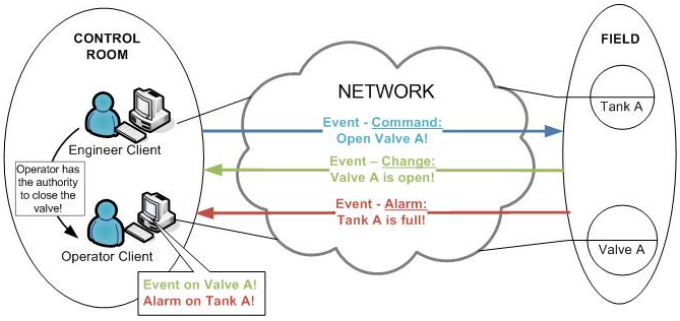

Fig. 3. SCADA system overview: control room and process field

A. System architecture

Despite the fact that there are several vendors, system architectures in various SCADA systems are similar and the terminology is interchangeable. Figure 4 shows a typical SCADA layered architecture [11]. Layer 1 consists of physical field devices, PLCs (Programmable Logic Controllers) and RTUs (Remote Terminal Units). An RTU evaluates signals from the process field (e.g., from a pump or a tank) and sends notifications to event collectors (Connectivity Servers (CS) in Layer 2). CS aggregate events from the field and forward them to SCADA users in the control room. The Domain Controller (DC) in Layer 2 holds local DNS and authentication data for user access. The Aspect Server (AS) is responsible for implementing the logic required to automate the process that SCADA controls. For example, an Aspect Directory (AD) in the AS holds information about working ranges of the field devices, the device topology, user access rights. Besides, the AS collects and stores data from the CS into audit and event logs. Various clients in Layer 3 represent SCADA users.

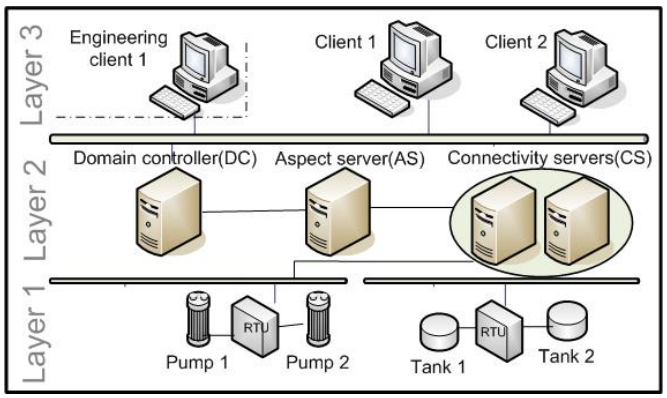

Fig. 4. SCADA system layered architecture

B. SCADA users

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 setpoints, etc. An operator monitors the system status and reacts to events, such as alarms, so that the process runs correctly (Figure 3). Typical operator actions, depending on the underlying industrial process, include commands such as: change switch status, increase temperature, open outlet, start pump.

Although industrial processes in various facilities differ in the details, user interaction with a SCADA system is broadly similar. To produce changes in the configuration of the plant side, an engineer needs to perform changes to the AS. After authentication to the DC, an engineer loads the AD, browses modules and performs modifications. Also, an engineer is responsible for making process scripts that automate the process. The content of the AD varies and it is customised for each stakeholder.

To perform a change in the industrial process, an operator manipulates process controls using a graphical interface. The interface shows topology of field devices, device statuses and real-time alarms. These notifications come from the CS. When performing an action on a field device (e.g., close a switch), the operator command is checked in the AS and then forwarded to the CS which communicates with RTUs in the field.

IV. Threat and vulnerability analysis

A. Types of threats in a SCADA environment

We classify possible threats in two groups: system- and process-related. System-related threats imply that an attacker hits computers, networks, sensors, PLCs or radio signals to cause failures in the SCADA system. These attacks are low level and typically occur on Layer 1 and Layer 2 of the SCADA layered architecture (Figure 4). On the other hand, process-related threats imply that an attacker gains user access rights (e.g., through social engineering) and issues legitimate SCADA commands to cause defects in the industrial process. These attacks are high level and typically occur on Layer 3 of the SCADA layered architecture (Figure 4). The system-related threats exploit software/configuration vulnerabilities of the SCADA system (such as buffer overflow or misconfigured radio network) whereas process-related threats exploit weak process controls in the SCADA system (such as no semantic check of engineer inputs before execution).

Some system-related attacks are not SCADA specific (e.g., malware attacks [2]). Yet, there is an increasing number of attack examples that relate only to SCADA technologies: e.g., an attacker manipulates data readings (e.g., for a tank) and sends misleading values to the system operator [2] or an attacker disrupts communication in the underlying network infrastructure by exploiting a protocol vulnerability [7]. In Section VIII we briefly indicate related work that addresses the detection of system-related attacks.

There are several examples of serious process-related attacks on SCADA systems from the past. For example, during the Maroochy water breach [12], an intruder managed to access SCADA network (by abusing a radio connection) and then used valid engineer credentials to performe a series of malicious actions: sewage leakage, pump station lockups, shutdown of the alarm system [12]. Similarly, during an attack on the L.A. traffic light system [13], intruders used valid engineer credentials to reprogram traffic lights causing jams on major city intersections and disconnect signal boxes at targeted intersections to slow down reparations [13].

The malicious behavior in these cases can not be recognised by traditional intrusion detection systems (such as the IDS that analyse network traffic data) because the anomalies generated by these attacks are not reflected in communication patterns/data, but can be seen only at a higher semantic level.

B. Identification of process-related threats

We focus on addressing process-related threats. This means that we analyse threats that originate from legitimate system commands performed by a legitimate user, or an illegitimate user who managed to obtain valid credentials. We distinguish two threat scenarios, namely (1) an attacker impersonates a system user or (2) a legitimate system user makes an operational mistake. Both scenarios represent potential threats to the system security and safety. Our goal is to identify single malicious user commands that may lead the system to an unwanted state.

The analysis of user behavior in SCADA systems is not trivial but it is structured enough to be feasible. There are two distinct kinds of behavior: one performed by an engineer and one performed by an operator. An engineer works with a user account that is typically used by only one person. However, an operator user account is, in practice, often shared among several people. Also, to perform a change on the plant side, an operator typically needs to perform a sequence of commands whereas an engineer uses individual commands. 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).

For these reasons, we focus our work on the analysis of commands performed by system engineers.

Our stakeholders come from various critical infrastructures (power grid, gas and water treatment). All these systems use similar underlying control systems. The main differences are in the speed of occurrences: power-related systems are considered fast (the consequences of actions are immediate- e.g., turning off a power supply) while gas- and water-related systems are considered slow processes (the consequences of actions are delayed - e.g., it takes two hours to overload a tank even while pumping at maximum speed). We focus our analysis on the engineer behaviour which is similar in both cases. Therefore, we believe that our approach could be extended to a variety of relevant contexts.

As a proof of concept, we analyse engineer behaviour in a real SCADA controlling water treatment. The facility is located in the Netherlands.

1) Methodology: We combine two methodologies to identify process-related threats. Those are: PHEA (Predictive Human Error Analysis) and HAZOP (Hazard and Operability

Study) [8]. PHEA takes a user-oriented approach to analyse human errors by building a task analysis tree. We use a PHEA tree to represent possible engineering actions in a SCADA system. PHEA then analyses the tree using human error classification (e.g., action is taking too long, action is performed incomplete [8]) This part of the analysis is not suitable in our case. Our goal is to identify actions that produce malicious consequences on the process, whereas PHEA originally focuses on identifying possible causes of human errors. To identify possible misuses of engineer actions (on the PHEA tree) we use HAZOP methodology.

HAZOP is the best documented methodology for addressing process safety problems [8]. HAZOP process deviations are built by combining chosen process keywords and process guidewords. We perform the HAZOP analysis on a restricted domain (engineer tasks).

We implement the combined methodology in two steps. First, we build a task analysis tree (as in PHEA). Our tree consists of possible engineering tasks within a SCADA customized AD. Figure 5 represents a part of an anonymised AD of the stakeholder company.

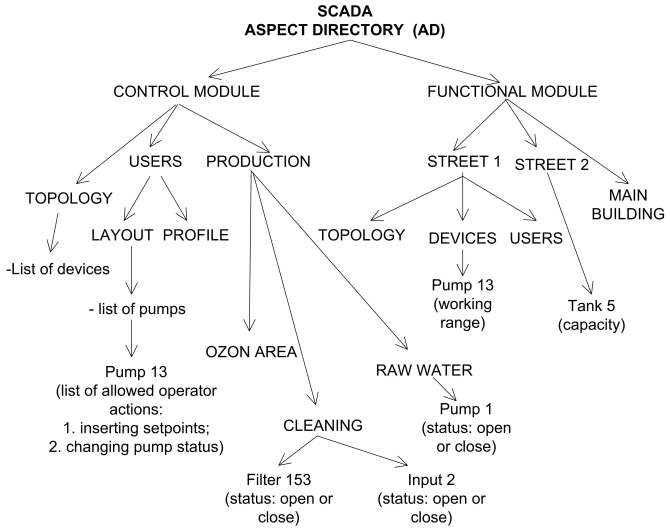

Fig. 5. Part of the AD in a working SCADA facility for water treatment

Second, we perform the HAZOP step. The leaves of the PHEA tree represent HAZOP keywords (such as pump 1, filter 153, tank 5). The SCADA engineering application allows an engineer to perform only three distinct operations: add, modify and delete. We use these operations as HAZOP guidewords. Following the concept of the HAZOP methodology, we build process deviations for all possible combinations of given keywords and guidewords. Table I shows examples how deviations are generated. The first column of the table consists of all chosen keywords (leaves of the PHEA task analysis tree from Figure 5). The second column consists of three chosen guidewords. Each keyword from the first column is combined with all three guidewords from the second column to build a deviation in the third column.

TABLE I
SYSTEM DEVIATIONS CAUSED BY ENGINEER ACTIVITY

Keyword	Guideword	Deviation / Potential threat
list of devices (in the topology)	add	a user **adds** a device to the list
	modify	a user **modifies** the name of a device
	delete	a user **deletes** a device from the list
tank (capacity)	add	no
	modify	a user **modifies** the capacity of tank (e.g., the capacity of tank 5 is increased by double)
	delete	no
pump (allowed operator actions)	add	a user **adds** action type to the allowed actions (e.g., a user adds "inserting setpoint" and/or "changing pump status" to the list of allowed operator actions on pump 13)
	modify	no
	delete	a user **deletes** action type from allowed actions (e.g., a user deletes "inserting setpoint" and/or "changing pump status" from operator actions on pump 13)
input (status)	add	no
	modify	a user **modifies** the status of input (e.g., the status of input 2 changed from open to close)
	delete	no

2) Identified threats: The stakeholder facility has around 200 unique components that can be influenced by engineer activity. By grouping similar devices (such as all tanks in one group, all pumps in the second) we narrow down the number of components we have to analyse for deviations. After a preliminary inspection, we conclude that some combinations of keywords and guidewords are not meaningful, such as: filter status cannot be added or deleted, tank capacity cannot be deleted, etc. However, a number of deviations is interesting. We compile around 20 deviations from 7 groups of components.

We narrow down the list of deviations during a focus group session with stakeholders. For example, we discard deviations that perform interface layout changes but do not imply functionality changes. On the remaining threats we distinguish two types of threat behaviour: scripting errors and misconfiguration. Threats due to scripting errors imply behaviour of writing or exploiting wrongly written process automation scripts by system engineers. Misconfiguring implies setting semantically dangerous configurations in the AD. Table II shows a summarized list of identified threats.

On all identified threats we perform a detailed analysis, together with stakeholders, to define the cause, effects and mitigation recommendations. Table III shows results of the detailed analysis on example of three identified threats.

C. Identification of vulnerabilities

Next, we take advantage of the stakeholder experience to identify system vulnerabilities. After a focus group session, the list of vulnerabilities found includes:

- no process safety checks are in place during the manual system mode;

- an input is not validated before execution of an engineer command;
- weak password policy;
- no detail operator auditing;
- only two user access levels (an engineer and an operator);
- limited separation between production and administration (no principle of least privilege: e.g, at the same time, an engineer has access rights to both process configuration and user-account administration).

D. Evaluation of the methodology

The stakeholder company performed an internal threat analysis in 2006. The analysis was based on the guidelines of the ISO/IEC 15408 IT standard [14]. The result of the analysis was a number of threats which exploit software/configuration system vulnerabilities because this kind of analysis identifies only system-related threats.

To identify process-related threats in SCADA systems, we take a higher level approach. This requires the cooperation with the process engineers to understand safety problems and what is undesirable system/user behaviour. The process-related analysis can not identify low level vulnerabilities (e.g., the vulnerability of authentication protocols and injection of malicious code). By contrast, the system-related analysis can not identify high level vulnerabilities (e.g, no process safety checks in manual mode, no detail operator auditing, only two user access levels).

Therefore, we argue that a process-related analysis is complementary to typical IT security analysis.

E. Mitigation of process-related threats

We analyse possible ways for mitigating identified process-related threats. This is the second step of our approach for addressing process-related threats in SCADA systems (Figure 2). By analysing the list of identified system vulnerabilities (Section IV-C) and mitigation recommendations (Table III) we conclude that identified threats can be mitigated in two ways: (1) by upgrading the proprietary SCADA software to support additional functionalities (such as an additional input check in manual mode or by introducing safety checks for engineer actions before execution) and (2) employing an independent tool to analyse data resources from SCADA and detect malicious behaviour. Because the former option is infeasible, as it would require vendor to develop additional software, the latter option looks easier to apply. However, no tool is available to detect process-related threats. Thus, we develop a tool. In the next section we describe our approach in detail.

TABLE II
IDENTIFIED PROCESS-RELATED THREATS

Type	Threat	Example
scripting error	an engineer inputs a value that causes errors in the system automation	e.g., an engineer inserts a wrong ratio of chemical components
scripting error	an engineer inputs a value that influences system performance	e.g., a high input value of a chemical produces slow system response due to repeated calculation errors
misconfiguration / functional	an engineer modifies a device parameter	e.g., change capacity of a tank to spoof the alarming system and cause hardware damage
misconfiguration / functional	an engineer modifies auditing policy	e.g., turn off all auditing
misconfiguration / control	an engineer modifies the range of allowed actions for a specific device	e.g., one cannot stop a pump anywhere
misconfiguration / control	an engineer modifies the system topology	e.g., some devices become invisible, and thus inaccessible
misconfiguration / control	an engineer modifies user access rights	

TABLE III
UNDERSTANDING MITIGATION STRATEGIES

Guideword	**MODIFY** - An engineer modifies the quantity value of chemicals (e.g., input 2).
Cause	Inside/outside malicious attack Human error
Effects	Errors in calculations Equipment damage Influence the product quality
Recommendations	Additional input checks Track user behaviour
Guideword	**DELETE** - An engineer deletes a device from the device topology (e.g., a pump 13)
Cause	Inside/outside malicious attack Human error
Effects	Device becomes inaccessible Equipment damages due to overload Inconsistent alarms
Recommendations	Increase the number of access levels Track user behaviour Safety checks on engineer actions before execution
Guideword	**MODIFY** - An engineer modifies a tank capacity (e.g., tank 1).
Cause	Inside/outside malicious attack Human error
Effects	Tank damage due to overload No alarm when real maximum reached Damage of interdependent equipment
Recommendations	Include safety checks during manual working mode Additional configuration checks Track user behaviour

V. MITIGATION CONTROLS

A. Resources

SCADA system logs represent a rich information resource providing a complete view of the industrial process as a whole. Logs can be so detailed to the point of capturing commands and network messages exchanged by devices in response to user actions. However, the size and high dimensionality of the logs make manual inspection infeasible. For instance, a SCADA system controlling a water treatment process, depending on the size of the facility, records between 5 000 and 25 000 events per day.

As the same underlying control system is used by different stakeholders (power grid, gas and water treatment companies), the logs in all these environments have a similar format. By structure, a SCADA event typically involves between 5 to 15 attributes of different data (e.g., time-stamp, type of operation, source component, object component, user, etc).

Data attributes can be nominal, interval, unstructured text, etc. Nominal attributes have a finite set of data values that appear in the log (e.g., 4 types of system messages, 5 different nodes in the system). The interval data type is used for numerical data (such as continuous values of tank level) and time data (such as time-series).

SCADA users do not normally inspect logs. The reason for this is that system engineers lack both time and specific tools/skills for performing a thorough analysis. Thus, a good deal of data never turns into information. As the logs keep traces about performed commands (e.g., an engineer performs a change in configuration), we believe that they hold critical information for the extraction of process-related incidents caused by a user activity.

B. Approach

Our approach to detect malicious behaviours (attacks or user mistakes) in SCADA systems is based on the semi-automated processing of system logs. We aim at detecting engineer activities that may potentially disrupt industrial process, cause harm to equipment or endanger people safety. We use visualization to highlight potential incidents. Visualization is the process of generating a picture based on log records. It uses tools for visualizing log content and a human brain as a pattern recognition tool. This concept has proved to be successful in supporting detection of outlier records and analysis of trends in various logs [15].

Our detection approach consists of two phases. First we use our tool, MELISSA, to prepare the data, reduce the size of the log and visualize log content. In the second phase, we manually inspect the visualized log to identify interesting events and detect potential malicious behaviour.

VI. Architecture

MELISSA consists of two interacting components: the Data Preparator (DP) and the Visualisation Engine (VE). Because of the size and a typically high dimensionality of SCADA logs, MELISSA performs two kinds of data reductions: variance and dimension reduction.

The Data Preparator (DP) loads logs exported from the SCADA system and performs data cleaning and variance reduction operations. The modified log is passed to the Visualisation Engine (VE). The VE performs dimension reduction and visualization of the log outputting a number of two-dimensional diagrams.

Figure 6 depicts MELISSA and its internal components.

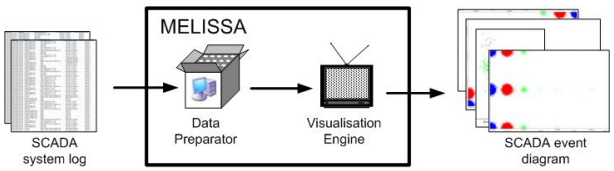

Fig. 6. MELISSA architecture

The DP prepares and cleans the data format of the input log before forwarding it to the VE. The variance reduction is performed through data aggregation operations. For example, we perform a time-based aggregation of an interval time stamp attribute. This way we transform the initial time stamp into several periods that correspond to the company's working shifts. This aggregation helps the final human inspection of the log because it groups the log events in a small number of time periods and provides a better view on performed activities. We explain the details of these transformations in the implementation section.

The DP also removes redundant events in the log (such as events from backup connectivity servers) and events that represent periodical signalling from SCADA components (these events are not caused by a user activity, thus are out of our scope of interest).

The Visualisation Engine (VE) accepts data from DP in a matching format and performs a dimension reduction. To reduce the number of dimensions, the VE processes data for each attribute and discards attributes that give no value to data quality (such as repeated and empty-value attributes).

Finally, our VE outputs the visualised log as n^2 two-dimensional diagrams where n is the total number of log attributes.

The final number of diagrams is decreased by a half (since the second half of diagrams are repeated diagrams with inverted axis).

The remaining diagrams are then inspected to detect malicious activities.

A. Implementation

We have implemented a prototype of MELISSA to address threats and vulnerabilities identified in Section IV. The prototype is written in Java. The DP automatically performs changes on the input log and formats the output file. For example, the DP aggregates values of the time-stamp attribute. The values of this attribute are transformed to one of three values: 1, 2 and 3. These values represent usual working shifts in the company. 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.

We use the Weka platform [16] as our Visualisation Engine. Weka provides an environment to load, analyse and visualize large datasets. It includes a large collection of machine learning algorithms and a number of data filters for data preprocessing. We use an attribute reduction filter (the unsupervised attribute filter - *ReduceUseless*) to reduce the number of dimensions in the log.

VII. Benchmarks

As a proof of concept we use a log generated by a real SCADA system for water treatment. Our initial dataset consists of 68 000 log entries, which corresponds to 14 days of normal process work. Each log entry has 13 attributes where 5 attributes are nominal, 4 are interval and 4 are strings. The stakeholders acknowledge that no known security incidents are captured in the log. Also, we manually process the log to confirm that the data is free of attacks and user mistakes. We

use logs captured with the default audit set up which collects events continuously through time.

Before analysis, we filtered out one alarm on hardware connection errors that repeatedly occurred in the log. This alarm was caused by a missing device due to maintenance.

We apply the attribute reduction filter on our log. This means that the applied filter reduces the number of data attributes from 13 down to 9. Three log attributes (*Message description, AD/Device Path* and *Source device*) have more than 200 distinct values each appearing in the log. Since visual human inspection can efficiently track only up to 8 distinct values [15], including these attributes into visualization process would not add useful information. Also, without a deep understanding of the process (which we do not currently own), it is difficult to derive an effective data aggregation. Nevertheless, we maintain these attributes for inspection during the later context analysis to confirm if suspicious events are actual incidents.

Our final experimental data set consists of 6 attributes (*Date, Working shift, SCADA node, Type of event, Aspect of event* and *User account*). A *SCADA node* represents a computer which sends event details to the log. In our case, there are three kinds of nodes: engineering, operator and system node. Engineering and operator nodes represent static stations where users (engineers and operators) login and perform actions. System nodes are typically CS of the SCADA system. This means that all the occurrences in the process field that appear in the log come from system nodes. Except to the engineering node, an engineer can also connect to a system node (e.g., connect to the CS and make changes on PLC). We find two active user accounts in our log: Engineer Bob and Operator X. The attribute *Aspect of event* takes one of 6 nominal values in the log. This attribute explains the character of event, such as: change of workplace layout, change in workplace profile, change in control module, etc. The attribute *Type of event* takes one of 8 nominal values. In the subsection VII-B we describe types of events that we found interesting for our analysis.

Based on 6 attributes, the VE outputs 15 unique two-dimensional diagrams. Since we focus our analysis on engineer behaviour, some remaining diagrams are not informative. For example, we do not get any useful information about engineer activity by analysing diagrams that visualise:

- *Date* versus *Working shifts*,
- *Date* versus *Type of event*,
- *Working shifts* versus *Type of event*, etc.

To highlight targeted incidents, we focus on diagrams that give an insight into:

- activities of various SCADA nodes (with focus to engineer and system nodes),
- time and location behaviour patterns of engineer user accounts,
- types of events that performed by each engineer user.

In the next sections we demonstrate three suspicious situations we found during the analysis of the initial SCADA log.

A. Diagram: SCADA node versus Working shifts

We consider a diagram with dimensions: *SCADA node* and *Working shifts*. In this case, we investigate how the system workload depends on the time of day. Figure 7 shows the activity of 8 SCADA nodes (aligned horizontally) versus 3 working shifts (aligned vertically). Each point in the graph represents one event from the log.

As we expected, system nodes (CS01 and CS02) are active during the whole day. Also, the activity of operator nodes (OP01, OP02, OP03 and OP04) shows that operators in a SCADA system work all day through and their actions appear in all working shifts. We expect activity from engineering nodes only during the usual working day time (from 08:00 to 17:59hrs). Activity of the engineering node E02 fits this pattern.

However, we detect one unexpected event from the engineering node E01 that occurred in the late night hours. A detailed view shows that the event represents an error message in an attempt of loading the AD. We investigate the context of the event during our focus session with the stakeholders. We conclude that this event represents the only activity of the node E01 in the log. This "night engineer activity" event did not produce any known problems. However, stakeholders decide to perform a detail analysis internally.

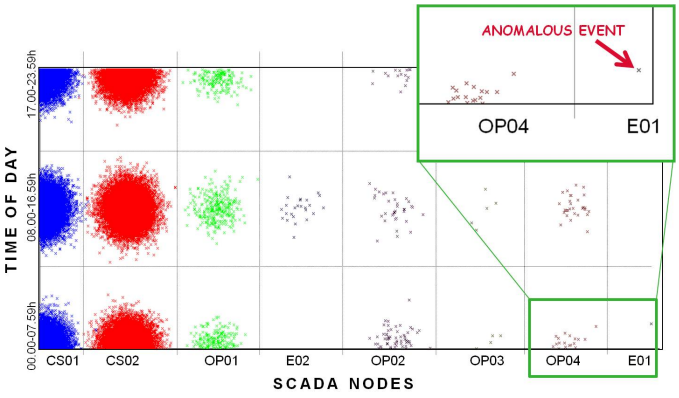

Fig. 7. Activity of SCADA nodes during the time of day

B. Diagram: Type of event versus SCADA node

Figure 8 shows data distribution by two log attributes: *Type of event* (aligned horizontally) and *SCADA nodes* (aligned vertically). As a third dimension, colour, we use attribute *User account*. Events performed by the user account Engineer Bob are coloured red and marked with dashed line borders in the diagram. Events performed by Operator X are coloured blue and marked with vertical full line border. Grey events (unmarked areas) represent all other occurrences coming from the system (such as system responses, device signals, error messages, etc).

We can see that Engineer Bob performs actions on two SCADA nodes (EN02 and CS02) producing two types of events (*SystemUser* and *AspectDirectory*). We now focus on the analysis of these two nodes. A *SystemUser* event occurs when a user logs in to the SCADA system. *AspectDirectoy* event occurs when a user performs changes on the AD.

1) Activity on the node E02: The top part of Figure 8 highlights the activity on node E02. The user-based actions

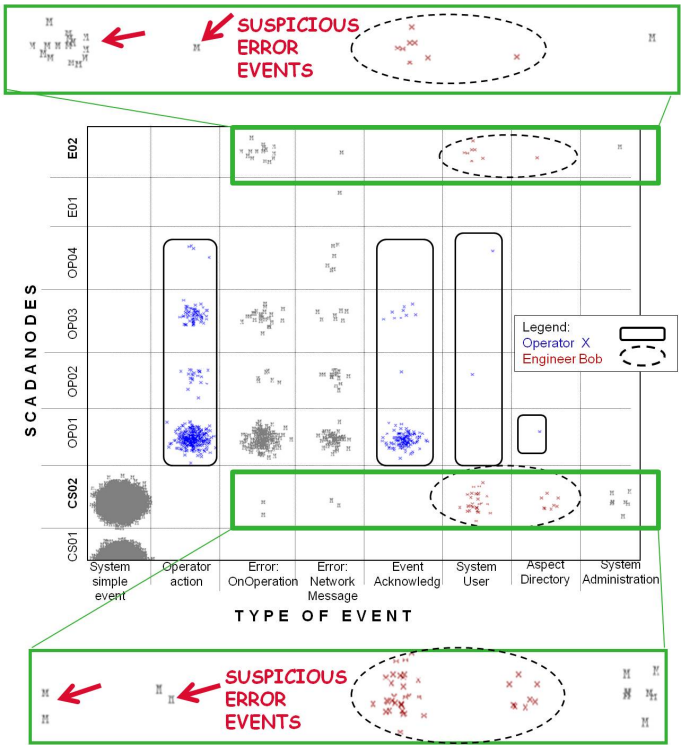

Fig. 8. Diagram showing user activity and suspicious errors (on nodes E02 and CS02)

TABLE IV
EVENTS OCCURRING ON THE NODE E02 ON DAY A

Day	Time	Action	Location
Day A	08:35	Engineer Bob log in	E02
Day A	08:36	Engineer Bob changes profile values in the Aspect Directory	E02
Day A	16:09	Error: NetworkMessage on E02 node - Overflow	E02
Day A	16:09	Error: OnOperation - Expression error at input X on data change	E02
Day A	16:11	Error: OnOperation - Expression error at input X on data change	E02
Day A	16:31	Error: OnOperation - Expression error at input X on data change	E02
Day A	16:36	Engineer Bob log out	E02

on E02 (marked with dashed line) show that Engineer Bob performs several login actions and one change in the AD.

When observing nonuser-based events (unmarked areas), we see an isolated, potentially suspicious, event in the *SystemAdministation* type of event. In a diagram, this type of event repeats actions performed in the *AspectDirectory* type of event. Thus this event is not unusual.

However, we identify several error events on the same node (*Error: OnOperation* and *Error: NetworkMessage*). The context of these error messages in the log shows that all the errors on E02 occurred during one day. That day, only Engineer Bob was logged on E02. We show a summary of actions on node E02 during day A (Table IV). Expression errors (from Table IV) typically mean that a user inserted an invalid command or value.

During our focus session with the stakeholders we developed three possible scenarios causing these errors:

TABLE V
EVENTS OCCURRING ON THE NODE CS02 ON DAY B

Day	Time	Action	Location
Day B	08:35	Engineer Bob log in	CS02
Day B	08:36	Engineer Bob changes profile values	CS02
Day B	08:36	Engineer Bob log out	CS02
Day B	09:11	Engineer Bob log in	CS02
Day B	11:09	Error: Network Message on CS02 node - Overflow	CS02
Day B	11:13	Error: OnOperation - Expression error on device Y: Faceplate* open error	CS02
Day B	11:23	Error: OnOperation - Expression error on device Y: Faceplate open error	CS02
Day B	12:09	Error: Network Message on CS02 node - Overflow	CS02
Day B	16:15	Engineer Bob log out	CS02

* A device faceplate is the window that opens when a user asks for details about a specific device.

- Engineer Bob made a mistake while inserting a value;
- another system user worked on E02 node under Bob's credentials and inserts an invalid value;
- an intruder got access to the E02 node and tried to perform actions.

No unusual activities nor security incidents were reported on the specific day. However, stakeholders agree that this behavior could be considered suspicious and promice to further investigate.

2) Activity on the node C02: C02 is the second node where Engineer Bob performed actions. The bottom part of Figure 8 highlights activities on system node CS02. Similar to node E02, Engineer Bob logs in to the CS02 several times on different days of the log. Dense clouds of events (e.g., in *Process simple event* type) indicate that the node CS02 is rather active. However, there are only 4 error events occurring on this node in the whole log. All errors occur during the time that Engineer Bob is logged in. Table V shows a summary of the day that errors happened.

During our focus session with the stakeholders we derived four possible scenarios for causing errors from Table V:

- Engineer Bob made a mistake while inserting a value;
- another system user worked on C02 node under Bob's credentials and inserts an invalid value that caused errors;
- an intruder got access to the C02 node and tried to perform actions;
- a specific faceplate on device Y is previously misconfigured or triggers a script error on every opening.

No unusual activities nor security incidents were reported on the specific day. However, stakeholders note that there are delays in opening several device faceplates in the system. The reason for this could be a bug in process automation code.

The two suspicious situations we detected (errors on E02 and C02) represent potential threats of scripting errors by system engineers. We did not find any anomalies that would imply configuration (functional or control) threats. The reason for this can be that logs were collected randomly and do not include extensive configuration changes done by engineers.

Finally, we took another sample of logs to evaluate our approach. This log originates from the same company as our initial sample an covers approximately the same time period of

system work (two weeks). The sample is again taken randomly, after around one year since the initial log. In the meantime, the company employed a new version of SCADA software. With minor changes (e.g., time stamp parser from the DP) we managed to run our tool and get visualised image of the log content. Although the engineer activity in the log is more frequent, we did not spot any anomalies as in the examples from the initial log sample (Section VII-B).

This confirms that the three detected anomalous situations from the initial log ("night engineer activity" and errors on E02 and C02) do represent unusual engineer behaviour.

VIII. Related work

Traditional methodologies for addressing safety problems in process control systems (e.g., FMEA, FTA, HAZOP [8]) do not consider security threats. By introducing a special set of guidewords, Winther et al. [17] show how HAZOP can be extended to identify security threats. Srivantakul et al. [18] combine HAZOP study with UML use case diagrams to identify potential misuse scenarios in computer systems. We take a similar approach to combine PHEA study with HAZOP and analyse user (engineer) behaviour in a SCADA environment.

Several researches address threats in SCADA systems. For the identification of threats, authors typically use questionnaires and interviews (such as in [19], [20]). To detect anomalous behaviour, authors use approaches based on inspecting network traffic [21], [6], validating protocol specifications [7] and analysing data readings [5]. 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 [5], [21] 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 only a partial view on the process state, the one concerning device occurrences. To the best of our knowledge, only Balducelli et al. [21] analyse SCADA logs to detect unusual behaviour. Authors use case-base reasoning to find sequences of events that do not match sequences of normal behaviour (from the database of known cases). Balducelli et al [21] use sequences of log events that originate from a simulated testbed environment. In contrast, we analyse individual events in logs from a real SCADA facility using visualization techniques.

IX. Conclusion and future work

We present a preliminary study where we address a class of threats that occur in SCADA environment, but which was not addressed before in a systematic manner. We call these threats "process-related threats". 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 (e.g., an engineer) makes an operational mistake and causes a process failure.

We propose a tool-assisted approach to address process-related threats. The approach consists of two steps: threat analysis and threat mitigation.

During the threat analysis, we first combine two different methodologies (PHEA and HAZOP) to systematically identify process-related threats. We then organize a group focus session with stakeholders to validate our results. By taking advantage of their experience, we identify vulnerabilities related to process-related threats.

Most vulnerabilities are due to a lack of system controls. Stakeholders acknowledge that currently no control (e.g., monitoring tools to detect misbehaviours) is available to mitigate process-related threats, and the only solution would be to apply additional system controls (which can be only done by vendors).

To mitigate process-related threats, we propose a tool that helps to identify incidents occurring due to process-related threats. The tool analyses system logs which hold critical information for incident identification, such as user activities and process occurrences. In real life, logs are rarely processed (let alone looked at) by stakeholders due to 1) the large number of entries generated daily by systems and 2) a general lack of security skills and resources (time).

Our tool can automatically process a large number of log entries and generate visual diagrams that can then be used by operators to spot potential incidents. We test the tool with real life logs from a water treatment facility located in the Netherlands. Preliminary results suggest that the tool helps stakeholders in identifying possible problems. Although no real incident occurred during the period of time we analyse, at least three events were labelled as "suspicious". Our suspicions have been acknowledged by stakeholders and two of them were explained as the result of scripting errors.

As future work, we plan to expand our tool to address anomalous command sequences, rather than single events/operations. This would allow us to detect additional threats.

References

[1] R. R. Rantala, "Cybercrime against businesses," tech. rep., U.S. Dept. of Justice, Office of Justice Programs, Bureau of Justice Statistics, Washington, D.C., 2004.

[2] C. G. Chittester and Y. Y. Haimes, "Risks of terrorism to information technology and to critical interdependent infrastructures," *Journal of Homeland Security and Emergency Management: Vol. 1 : Iss. 4, Article 402*, pp. 341–348, 2004.

[3] A. Rege-Patwardhan, "Cybercrimes against critical infrastructures: a study of online criminal organization and techniques," *Criminal Justice Studies*, vol. 22, pp. 261–271, Sep 2009.

[4] J. Stamp, J. Dillinger, W. Young, and J. Depoy, "Common vulnerabilities in critical infrastructure control systems," tech. rep., Sandia National Laboratories, 2003.

[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, pp. 171–182, Springer Verlag, 2003.

[6] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," in *IJCNN'09: Proc. International Joint Conference on Neural Networks*, pp. 102–109, IEEE Press, 2009.

[7] C. Bellettini and J. L. Rrushi, "Vulnerability analysis of SCADA protocol binaries through detection of memory access taintedness," in *Proc. 8th IEEE SMC Information Assurance Workshop* (L. J. Hill, ed.), pp. 341–348, IEEE Press, 2007.

[8] F.P.Lees, *Less' Loss Prevention in the Process Industries*. Butterworth-Heinemann, 3 ed., 2005.

[9] A. G. Gary Stoneburner and A. Feringa, *Risk Management Guide for Information Technology Systems, NIST Special Publication 800-30*. National Institute of Standards and Technology, 2002.

[10] "ISO/IEC 27001:2005 information technology – security techniques – information security management systems – requirements," 2005.

[11] *ABB University: T315, Engineering an 800xA System, Course manual*.

[12] J. Slay and M. Miller, "Lessons learned from the Maroochy water breach," in *Critical Infrastructure Protection*, pp. 73–82, 2007.

[13] S. Bernstein and A. Blankstei, *Two deny hacking into L.A.'s traffic light system*, accessed October 2010. http://seclists.org/isn/2007/Jan/50.

[14] "ISO/IEC 15408:2005 information technology – security techniques – evaluation criteria for – it security," 2005.

[15] R. Marty, *Applied Security Visualization*. Addison-Wesley Professional, 1 pap/cdr ed., August 2008.

[16] T. University of Waikato, *Weka 3: Data Mining Software in Java*, accessed October 2010. http://www.cs.waikato.ac.nz/ml/weka/.

[17] R. Winther, O.-A. Johnsen, and B. A. Gran, "Security assessments of safety critical systems using hazops," in *SAFECOMP '01: Proceedings of the 20th International Conference on Computer Safety, Reliability and Security*, LNCS 2187, (London, UK), pp. 14–24, Springer-Verlag, 2001.

[18] T. Srivatanakul, J. A. Clark, and F. Polack, "Effective security requirements analysis: Hazop and use cases," in *In Information Security: 7th International Conference*, LNCS 3225, pp. 416–427, Springer, 2004.

[19] E. Luiijf, M. Ali, and A. Zielstra, "Assessing and improving SCADA security in the Dutch drinking water sector," in *Critical Information Infrastructure Security: Third International Workshop, CRITIS 2008, Rome, Italy*, LNCS 5508, (Berlin, Heidelberg), pp. 190–199, Springer-Verlag, 2009.

[20] M. G. Jaatun, E. Albrechtsen, M. B. Line, S. O. Johnsen, I. Wærø, O. H. Longva, and I. A. Tøndel, "A study of information security practice in a critical infrastructure application," in *ATC '08: Proceedings of the 5th international conference on Autonomic and Trusted Computing*, LNCS 5060, (Berlin, Heidelberg), pp. 527–539, Springer-Verlag, 2008.

[21] C. Balducelli, L. Lavalle, and G. Vicoli, "Novelty detection and management to safeguard information-intensive critical infrastructures," *Int. J. Emergency Management*, vol. 4, no. 1, pp. 88 – 103, 2007.