PLC-BASED CYBER-ATTACK DETECTION: A LAST LINE OF DEFENCE

DAVID ALLISON and PAUL SMITH
AIT Austrian Institute of Technology
Vienna, Austria
Email: david.allison@ait.ac.at

KIERAN MCLAUGHLIN
Queen's University Belfast
Belfast, UK

FAN ZHANG and JAMIE COBLE
University of Tennessee-Knoxville
Knoxville, USA

RODNEY BUSQUIM
University of Sao Paolo
Sao Paolo, Brazil

Abstract

Nuclear facilities are making increasing use of digital technology for Instrumentation and Control (I&C). This has several benefits, including increased efficiency and maintainability of systems. However, digitalization introduces the risk of cyber-attacks to systems that support critical facility functions. To realize such a cyber-attack, an attacker must compromise and manipulate the I&C systems that are associated with a targeted function, e.g., by targeting the I&C systems that are controlling pressure levels in the pressurizer vessel of a Pressurized Water Reactor (PWR). To sabotage a function, an adversary can compromise the systems that directly support the function. A widely-used digital technology in this context is the Programmable Logic Controller (PLC), which executes user programs that engages actuators, based on sensor readings. There are several ways that PLCs can be compromised and manipulated. A particularly challenging type of attack is a False Data Injection Attack (FDIA), which involves an adversary manipulating sensor data to cause a PLC to incorrectly control a process. In this paper, a system to detect cyber-attacks, which aim to sabotage a facility function, is presented. Leveraging capabilities of a widely-available state-of-the-art PLC, the system executes directly on the device that is controlling a process. The intention is to provide a *last line of defence* to sabotage attacks. A method to detect FDIAs, which is intended to be integrated into the proposed system, is described. The results from an initial evaluation are shown that suggest the proposed approach is feasible, both regarding the capabilities of the PLC and the ability to detect FDIAs.

1. INTRODUCTION

Nuclear facilities are increasingly using digital technologies to support important facility functions, such as reactor cooling. This has several benefits, including the potential for increased efficiency of operations and maintainability of systems. However, digitalization introduces a greater risk of cyber-attacks sabotaging facility functions, potentially resulting in nuclear security or safety consequences. To implement a cyber-attack that results in these forms of consequence, an adversary should have to compromise several systems, which are arranged using a defence-in-depth strategy. This requirement gives a defender the opportunity to detect an attack, prior to it reaching the systems that directly support a function that is being targeted. However, for several reasons, it may not be possible to detect an attack prior to it engaging the targeted systems.

A commonly used digital technology that supports process control in nuclear facilities is Programmable Logic Controllers (PLCs). These devices execute user programs that control actuators, which affect physical processes, based on input from digital and analogue sensors. To cause the sabotage of a facility function, an adversary can manipulate the operation of a PLC. This can be achieved in several ways, for example, by stopping or changing the user program executing on the PLC, or by manipulating the sensor data that is used to inform control behaviour. The latter is known as a False Data Injection Attack (FDIA).

In this paper, a system is presented for detecting attacks to PLCs that executes directly on the device. The aim is to provide a last line of defence, in case an attack is not detected prior to it engaging the PLC and the data

it uses for operation. The system is implemented on the Siemens SIMATIC S7-1518 PLC, which supports the execution of programs that are implemented using the C++ programming language. These C++ programs can interface with the portion of the device that executes user-defined control programs. In contrast to previous work, the proposed system has a reduced attack surface – as it executes directly on the PLC – and its failure should not affect the normal control behaviour of the device. Additionally, an approach to detecting FDIAs is presented that uses the residual (difference) between the predicted state of a system and the measured state to identify anomalous behaviour. The intention is to integrate this approach into the system for intrusion detection that executes on the PLC. An initial evaluation has been performed using a so-called hardware-in-the-loop testbed, wherein PLC hardware is integrated with a Pressurized Water Reactor (PWR) simulator. Results indicate that an FDIA targeting a PLC that is controlling the pressure level in pressurizer vessel of a PWR can be detected and that the execution of resource-intensive C++ programs on the PLC does not adversely affect the control behaviour of the device.

2. RELATED WORK

In this section, we provide an overview of closely-related work on approaches to detecting cyber-attacks using Programmable Logic Controllers (PLCs) and identifying attacks that target the manipulation of processes.

2.1 PLC-based intrusion detection systems

Jin *et al.* have created a "lightweight" cyber-attack detection solution, called *Snapshotter,* that uses a host agent to log PLC inputs and outputs [1]. The input and output (IO*)* values are then periodically forwarded to a server that runs a simulation of the PLC's control. The server's simulated values are compared to the real IO from the PLC and security events are generated when they do not match. The logging mechanism is *forward-secure,* ensuring data integrity and using fresh encryption keys for each log. One drawback of this external monitoring approach is that such a setup increases the number of hardware and software components on the control network and, therefore, increases the attack surface. For example, the authors have used the OpenPLC framework for their agent, which runs on a Raspberry Pi and implements a web server.

Garcia *et al.* use a Siemens S7-1515 PLC to monitor PLC logic by using the PLC's own hypervisor which runs a Windows virtual machine (VM) [2]. The authors create a dynamic-link library (DLL) and load it to the hypervisor. The Windows DLL uses a shared area of memory between the hypervisor and the PLC logic. This shared area of memory acts as a temporary buffer for the PLC's IO. The DLL checks the IO against a set of pre-defined safe values, and only if deemed safe are they forwarded to their destination. This integrated Windows DLL solution is perhaps the most valuable work in the field of PLC-based security solutions. The DLL's use of shared memory between virtual machines means that the IO verification is happening at one of the lowest levels possible for a Windows software application. However, this IO verification is executed during the PLC's program cycle. This means that if the DLL encounters errors, or if the shared memory addresses change, then the maximum cycle time could be violated, and the PLC could enter the FAULT or ERROR mode, thus ceasing program execution. Furthermore, the authors explicitly state that their solution assumes that S7CommPlus has not been reverse engineered and that the attacker has no programming connection; this situation is unlikely to persist [12].

In contrast to these contributions, our approach to PLC-based attack detection uses capabilities that are natively supported by the Siemens S7-1518 PLC. This has several advantages, including reducing the attack surface that the detection capability introduces (when compared to the approach that has been proposed by Jin *et al.*), and using standard interfaces that, in the case of the failure of our approach, do not adversely affect the control operation of the PLC.

2.2 Detecting cyber-attacks to processes

Goh *et al.* have applied an unsupervised Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) to predict the expected values of parameters from a process [3]. The cumulative sum (CUSUM) method is applied to the residuals to detect the deviation of predicted values from the actual sensor data. A dataset that was collected from a large-scale Secure Water Treatment Testbed (SWaT), which is built by the Singapore University of Technology and Design, is utilized to test the effectiveness of the method [4]. The results show that 9 out of

10 cyber-attacks were detected. However, this approach has high computational cost, so it was not applied to all the process stages in SWaT.

Eggers applied Principal Component Analysis (PCA) and Independent Component Analysis (ICA) with a static and moving window to detect simulated cyber-attacks in Nuclear Power Plants (NPPs), based on modified real-time normal data that was acquired from an NPP [5]. These cyber-attacks simulate a simultaneous physical Small Break Loss Of Coolant Accident (SBLOCA) with an FDIA towards safety systems, which fools the safety system with normal data to prevent the reactor safely shutting down. Twenty-nine signals from different subsystems, including the reactor coolant system, reactor building, and makeup tank, were selected to evaluate the proposed attack detection model, and the results show that the models detect attacks successfully.

Zhang *et al.* have developed a cyber-attack detection system to provide broad attack coverage using supervised and unsupervised machine learning algorithms, based on the combination of cyber data and process data [6][7]. Process data was utilized to detect small deviations from normal operation using an unsupervised model. The data collected from a real-time testbed with a physical flow-loop facility and a control system [8] under several cyber-attacks, was utilized to evaluate the system. The results show that the system detects these cyber-attacks effectively.

This research show that process data can be valuable to detect cyber-attacks. However, all of this research utilizes data across subsystems, which has two drawbacks*: (i)* the systems that are used to collect the data, which is used for detection, have a relatively large attack surface that could result in the data being tampered with; and *(ii)* in the event of a resource starvation attack, such as a Denial of Service (DoS) attack, it may be that data cannot be collected to enable detection. Therefore, this paper proposes a localized model to maintain a small attack surface and improve data availability by acquiring data locally.

3. PRELIMINARIES

Preliminary information is provided, including an overview of the PLC that the proposed detection system is based on and an introduction to the operation of a pressurizer in a Pressurized Water Reactor (PWR).

3.1 The Siemens Simatic S7-1518 MFP programmable logic controller

The Simatic S7-1500 range of controllers are the most functionally sophisticated PLCs that Siemens AG produces. They include features that are targeted for Industrial Internet of Things (IIoT) environments and offer several physical interfaces and support new industrial automation protocols, such as OPC Unified Architecture (UA). The newest S7-1500 PLCs include a hypervisor that executes a VM, as well as the standard PLC firmware and hardware. In the case of the S7-1518 Multifunctional Platform (MFP) PLC, Siemens has developed a custom, lightweight Linux operating system distribution, which executes on the same x86_64 chip that also executes the PLC's control logic. The Linux VM is referred to as the *C++ Runtime* and the term *CPU Runtime* is used to refer to the area in which PLC logic is executed. Siemens suggest that the intended purpose of the *C++ Runtime* is to execute more complex computational algorithms and to send data to enterprise resource planning systems. Applications that execute in the *C++ Runtime* are implemented using the C and C++ programming languages [9]. This arrangement is depicted in Fig. 1.

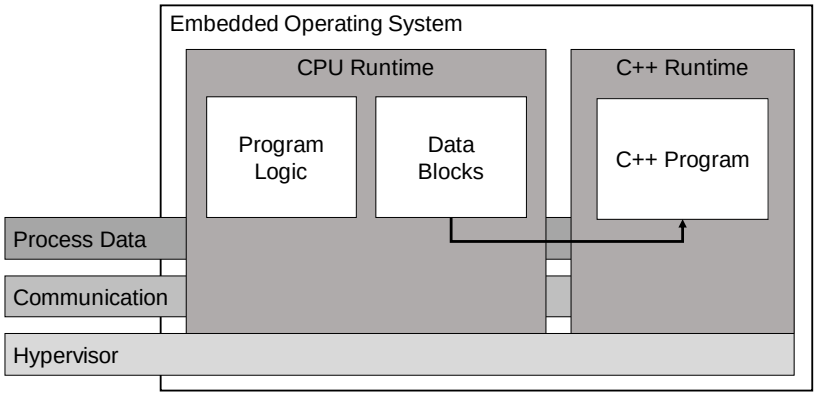

Fig. 1. An overview of the implementation of the hypervisor on a Siemens S7-1518 MFP PLC [9]

The S7-1500 controllers – including the S7-1518 MFP – can host an OPC UA server, which can be used to expose the PLC's state to clients. The variables in the PLC's data blocks, function blocks, and organisation blocks (OBs) can be set to "accessible from OPC-UA server" via the *TIA Portal* – Siemens' proprietary programming interface. This setting allows OPC UA clients to connect to, write to, and read from the server. It is possible to secure OPC UA connections using encryption and require clients to authenticate with a password.

The Siemens S7-1518 MFP PLC is not certified for use in nuclear facilities. However, in the future, devices that are like the S7-1518 may be introduced into facilities to support the control of non-safety processes, while enabling advanced on-device computation and data sharing that is supported by features such as the *C++ Runtime*.

3.2 The Asherah hypothetical NPP and pressurizer control

The PLC-based detection system has been evaluated using the hypothetical Asherah Nuclear Power Plant (NPP), which has been developed as part of the IAEA Coordinated Research Project (CRP) J02008. At the core of the Asherah NPP is a MATLAB/Simulink model of a Pressurized Water Reactor (PWR) that can be interfaced with external (to the model) systems, such as controllers. In this way, PLCs can control modelled processes. This feature enables cyber-attacks, which aim to sabotage a controlled process, to be executed against representative hardware devices and their consequences manifest in the model. For our experiments, we interface the Asherah model with two PLCs: a Siemens S7-1518 MFP and an older Siemens S7-400 PLC, which we use to control the pressurizer pressure level.

The purpose of a pressurizer in a PWR is to maintain the coolant (water) pressure in the reactor coolant system at a level such that it does not boil, i.e., such that it remains in a liquid state. The pressurizer is a large vessel that is filled with water to approximately a third of its height. To affect the pressure, there are two types of heaters in the bottom of the vessel that can heat the water to increase the pressure (so-called proportional and backup heaters); meanwhile, there are two spray valves in the top of the vessel that can spray cold water to reduce the pressure. In the Asherah model, the nominal pressure of the pressurizer vessel should be 15.1 Megapascals (MPa). The PLC implements control logic that uses the actuators to control the pressure level at the target setpoint. A summary of the control behaviour that is implemented by the PLCs is shown in Fig. 2.

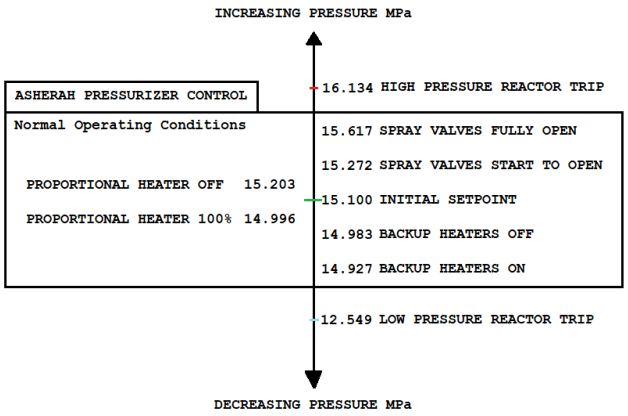

Fig. 2. A summary of the control behaviour for managing the pressure in the pressurizer vessel in the Asherah model

4. GOOSEWOLF: PLC-BASED INTRUSION DETECTION SYSTEM

The intrusion detection system that has been developed for the Siemens S7-1518 MFP PLC, called *Goosewolf*, is described in this section, along with an anomaly detection algorithm that can be executed on the PLC to identify when an attacker is attempting to sabotage the operation of a process, such as a pressurizer.

4.1 The Goosewolf intrusion detection system

Goosewolf is the (arbitrary) name of a C++ program that has been developed for the *C++ Runtime* of a Siemens S7-1518 MFP PLC to detect when an adversary has manipulated the process control of the PLC logic.

The main features of Goosewolf are monitoring the PLC's execution, logging security events, and sending security alerts to external computers. The Goosewolf program connects as a client to the S7-1518 PLC's OPC UA server (see Section 3.1). This connection between the *C++ Runtime* and the *CPU Runtime* is not exposed to the network, as it is handled internally by the hypervisor. Once a connection to the OPC UA server is established, the tags in the server can be accessed using the *UA_NODE_STRING* method. These tags are the same variables that are found in the PLC's data block. The program checks the variables from the OPC UA server against "golden values" – those that represent idealized normal operation – every second. A summary of the main features that are checked and logged by the Goosewolf program are summarized in Table 1. By monitoring these features, Goosewolf aims to identify two main issues: *(i)* when there are abnormalities in the controlled process (the pressurizer); and *(ii)* there are changes in the PLC's state, which could indicate an attack.

TABLE 1. A SUMMARY OF THE FEATURES THAT ARE MONITORED BY GOOSEWOLF

	Feature
Process	Control actions that could result in the physical process being in a hazardous state
	Pressure values dynamics – stationary values could indicate problems
	Setpoints that are associated with the control logic (see Fig. 2)
PLC	Cycle time (the time to execute the program's organization and function blocks)
	Program checksum
	Program file size
	PLC state (determining whether the PLC is in RUN mode)

The operation of Goosewolf can be summarized, as follows. If the PLC is operating normally there are three logs generated every time the Goosewolf program is executed: *init.log*, *conn.log*, and *cycles.log*. *Init.log* logs the initial time of the program's execution. *Conn.log* logs the time of the first connection to the OPC UA server. Thereafter, each time that data is processed, cycle time information is logged in *cycles.log*. This includes the most recent cycle time for OB1 (the main user program), the longest full program cycle, the shortest full program cycle, and the most recent full program cycle. Like *init.log* and *conn.log*, each entry has a timestamp. In addition to the logs that are created under normal conditions, there are logs for abnormal operating conditions, hazardous control actions, and program change detection (*aoc.log*, *hca.log*, and *proginf.log*, respectively). This operation is summarized in Fig. 3.

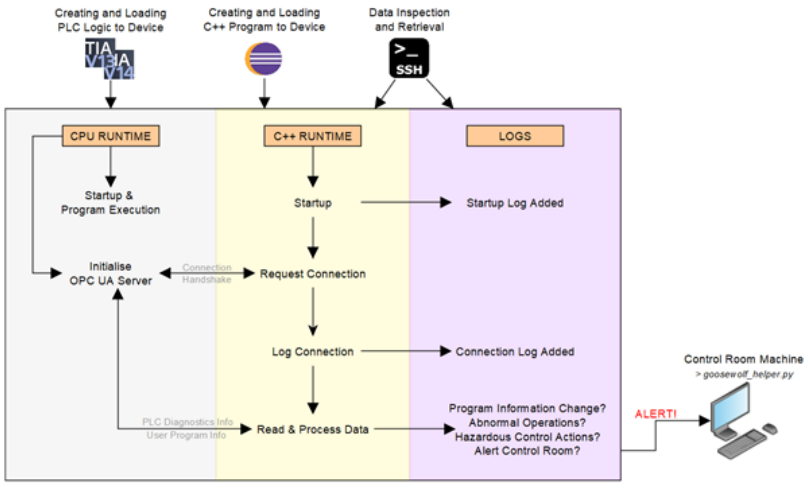

Fig. 3. An overview of the communications and functions of the CPU and C++ Runtimes of the Siemens S7-1518 MFP PLC running Goosewolf

Abnormal operating conditions are detected by comparing the constant values in the PLC program to the nuclear engineering documentation for the pressurizer – the aforementioned "golden values". If these constants – which dictate when and how rapidly the heaters and spray valves turn on and off – are different to the documentation, then a log entry is created in *aoc.log*. If the backup heaters, proportional heater, or spray valves

are turned fully on, a log entry is created in *aoc.log*. This is because the pressurizer is operating at its upper and lower limits which, if not controlled, could lead to problems in the pressurizer vessel. Furthermore, if the pressure value is unchanged for five seconds, Goosewolf assumes that the PLC is not receiving the correct, up-to-date, pressure value and creates a log in *aoc.log*. This assumption is based on the expectation that relatively small deviations in pressure are expected over time; an absence of these deviations could indicate a problem with the PLC, for example.

Hazardous control actions are defined as actions that further facilitate under- or over-pressure in the pressurizer vessel, i.e., control actions that do not conform to the control narrative that is depicted in Fig. 2. The pressure value is compared to the proportional heater, backup heater, and spray valve values to make sure that control actions are appropriate. For example, if the pressure is at the setpoint of 15.1 MPa, the main heater should be at 50% output. However, if the heater value returns 0%, this would mean that the control is further facilitating a low-pressure state. Likewise, if the backup heater is on when the pressure is at the setpoint of 15.1 MPa, this could facilitate a dangerous increase in pressure. Therefore, these are hazardous control actions. The same check is done using the pressure value and the values from the spray valves.

PLC program information that is checked includes the program size, cycle times, and a program checksum. Program information is logged in *proginf.log*. Cycle times are logged (along with a timestamp) in *cycles.log*. The cycle times include OB1's most recent cycle time, the longest full program cycle time, the shortest full program cycle time, and the most recent full program cycle time. Cycle logging is for post-incident analysis, as there is no processing or interpretation of the cycle times in the C++ program. The Siemens checksum function is a checksum of all "non-safety" blocks in the program. The pressurizer logic contains no safety blocks; therefore, the checksum encompasses the full user program. If there is a difference in the hard-coded checksum and that from the PLC, a log entry is created once. For every comparison thereafter, the checksum is compared to the hard-coded value as well as the last logged checksum. Only if it is unique to both checksums is another log entry created. This is to avoid having multiple log entries for the same checksum and to reduce log size. The program size (in kilobytes) is checked against the hard-coded program size. If there is a change in size, a log entry is created. This is implemented in the same way as the checksum, in which there is only one log entry for each time the program size changes.

In addition to the log files that are generated, selected abnormalities that have been identified by Goosewolf result in an alert being generated and transmitted via the network to a pre-configured destination. Alerts are generated when a change in the PLC state has been detected and when abnormal or hazardous control actions are identified. The alerts can then, for example, be ingested by a Security Information and Event Management (SIEM) tool for processing by an operator.

4.2 Detecting false data injection attacks

To sabotage the operation of a process, an adversary can conduct an FDIA on the PLC. An FDIA involves the adversary manipulating sensor measurements, which the PLC uses for control, to cause incorrect control actions to be applied. In the example considered in this paper, an adversary could manipulate sensor data about the pressurizer state (pressure) to cause the PLC to incorrectly engage the spray valves or heat banks, to cause under- or over-pressure.

To detect these forms of attack, a localized Auto-Associative Kernel Regression (AAKR) model has been developed. (This model has proven to be effective in detecting FDIAs in previous work [10].) AAKR is a nonparametric, memory-based algorithm that predicts expected measurements by calculating a weighted average of a memory matrix. This is a matrix that has been created using a fault-free training data set that represents the normal behaviour of a system. Let $\boldsymbol{X}$ denote the memory matrix and $X_{i,j}$ is the ith observation of the jth variable from the process under consideration (e.g., the pressurizer pressure); n_m is the number of the memory vector and p is the number of variables:

$$\boldsymbol{X} = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,p} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n_m,1} & X_{n_m,2} & \cdots & X_{n_m,p} \end{bmatrix}$$

The new input data point, which needs to be predicted is called the query vector $\boldsymbol{x}$:

$$\boldsymbol{x} = [x_1 \quad x_2 \quad \dots \quad x_p]$$

The similarity of the query vector and the memory matrix is measured by the distance between them – here the Euclidean distance d_i is utilized:

$$d_i = \sqrt{\left(X_{i,1} - x_1\right)^2 + \left(X_{i,2} - x_2\right)^2 + \cdots + \left(X_{i,p} - x_p\right)^2}$$

Subsequently, the weights of each memory vector are calculated; h is the Gaussian Kernel that can be optimized by the user:

$$w_i = e^{-{d_i}^2 / h^2}$$

The predicted value of the query vector is obtained using the following equation:

$$\hat{x} = \frac{\sum_{i=1}^{n_m}(w_i\, X_i)}{\sum_{i=1}^{n_m} w_i}$$

To detect an attack using the predicted value $\hat{x}$, the residual between the prediction and the measured values (i.e., that communicated to the PLC) is calculated. A threshold for the residual can be defined that indicates the measurements are anomalous – i.e., are understood to be manipulated. In the study presented herein, $p = 5$ and the variables that are used for detection are the *(i)* pressurizer pressure, *(ii)* spray valves state (%) – there are two spray valves in the model, *(iii)* proportional heater state (%), and *(iv)* backup heater state (a binary value). These values are available at the PLC and can be collected from the *CPU Runtime* using the OPC UA server on the S7-1518. The number of entries in the memory matrix that is used in this study and our evaluation is $n_m = 3458$. This state needs to be stored in the *C++ Runtime* of the PLC to enable the similarity (distance) between the query vector (measurement set) and the memory matrix which is calculated off-line in advance. This results in a file size of approximately 4Kb that can be readily stored on the PLC, which has 256Mb of storage available.

5. EVALUATION

5.1 PLC real-time performance concerns

A concern when implementing an IDS on a PLC is the potential affect that it may have on the control functionality provided by the device. Specifically, implementing anomaly detection algorithms, such as AAKR described in Section 4.2, has the potential to affect the performance of the real-time *CPU Runtime* on the PLC. This could be problematic when the PLC is controlling processes that have strict real-time requirements. To evaluate the potential for any such impact, a computational stress test on the Siemens S7-1518 MFP PLC was conducted [11]. To achieve this, a C++ program was implemented that repeatedly calculates MD5 hashes of large bodies of literary work from William Shakespeare – a relatively computationally expensive task. This program was executed in the *C++ Runtime* of the S7-1518 PLC while *CPU Runtime* cycle times were collected using the OPC UA interface on the S7-1518. In these experiments, the *CPU Runtime* of the PLC is executing the control algorithm that is used to manage the Asherah pressurizer (see Fig. 2). The results of this exercise are presented in Fig. 4, which shows the moving average value of the *CPU Runtime* cycle times (in nanoseconds) over time.

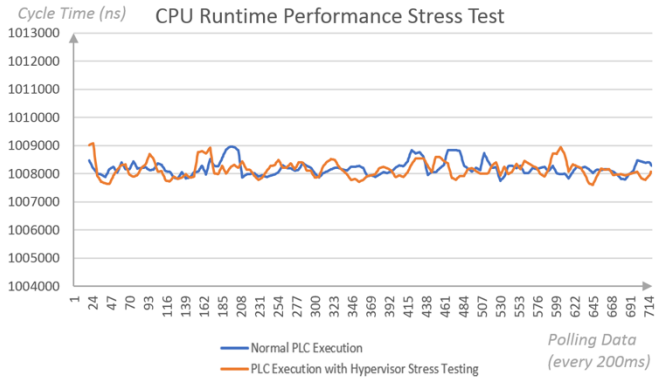

Fig. 4. CPU Runtime cycle times under normal conditions and while stress testing the C++ Runtime

It can be seen from the results that are presented in Fig. 4 there is no discernible impact on the *CPU Runtime* cycle times, which could be caused by the hashing program being executed. This result suggests there should be no impact on the real-time performance of the *CPU Runtime* on the S7-1518, which could be caused by executing an IDS in the *C++ Runtime*. This result indicates that the implementation of the hypervisor functionality in the Siemens S7-1518 MFP can correctly manage the resource utilization of the *C++ Runtime*, such that it does not affect the *CPU Runtime* of the device.

5.2 Process anomaly detection using AAKR

To evaluate whether the AAKR model that is presented in Section 4.2 can detect FDIAs that aim to subvert process control, we have conducted experiments with a Siemens S7-400 PLC and the Asherah MATLAB/Simulink model. The AAKR model was implemented using MATLAB.

As described in Section 3.2, the PLC is used to control the behaviour of the pressurizer in the model. We used the S7-400 for this experiment, as it communicates with an older version of the Siemens S7Comm control protocol that does not support, amongst other shortcomings, authentication of endpoints and integrity checking of messages. (The S7-1518 does support these features with a newer version of the S7Comm protocol, although progress has been made to subvert it [12].) This deficiency enables us to perform a so-called *Man-In-The-Middle (MITM) attack* on the communications between the Asherah model and the PLC. This attack enables an adversary to inspect, manipulate and drop the communication between two endpoints from a third device. In our experiments, the MITM attack is used to manipulate the pressurizer pressure values that are communicated to the PLC from the Asherah model. This arrangement is depicted in Fig. 5. In a real-world deployment, this communication could be between a Distributed Control System (DCS) and a PLC, or between a Remote Terminal Unit (RTU) that is collecting sensor data (e.g., pressurizer pressure) and a PLC or DCS, for example.

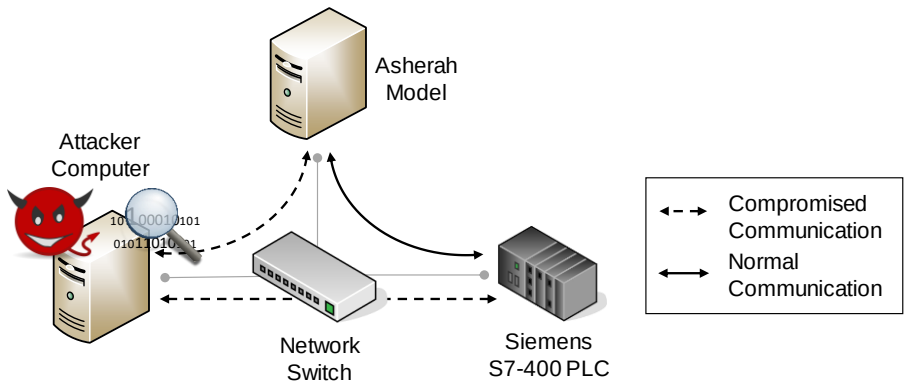

Fig. 5. An overview of the MITM attack between the Asherah model and the Siemens S7-400 PLC

To define $\boldsymbol{X}$ – the memory matrix for the AAKR model – nominal data from the pressurizer was collected from the Asherah model. As mentioned in Section 4.2, this data includes the pressurizer pressure, the state of the two spray valves, and the proportional and backup heater state. To generate test (attack) data for the AAKR model,

we collected the same type of data from the PLC, which includes a period of nominal behaviour and then introduces the FDIA after 2060 observations. The FDIA changes the pressurizer pressure value to indicate under-pressure in the vessel. This behaviour is depicted in Fig. 6, which shows the pressurizer pressure (Pa) (Fig. 6a) and the setpoints for the proportional heater and spray valves (%) (Fig. 6b). As the pressurizer pressure is manipulated to indicate under-pressure, the PLC attempts to compensate by setting the proportional heater to 100% and engages the backup heaters. The short peaks that can be seen during the attack relate to untampered data reaching the PLC, due to shortcomings of the MITM attack implementation (i.e., temporarily, the MITM attack fails, allowing untampered data to be communicated directly between the Asherah model and the S7-400 PLC). As this occurs, there are two interesting observations: *(i)* the actual pressure in the pressurizer is increasing, because of the attack, as indicated by the increasing peak pressure levels that are observed during the attack period; and *(ii)* the PLC briefly disengages the proportional heaters and fully-engages the two spray valves to mitigate the over-pressure situation in the pressurizer.

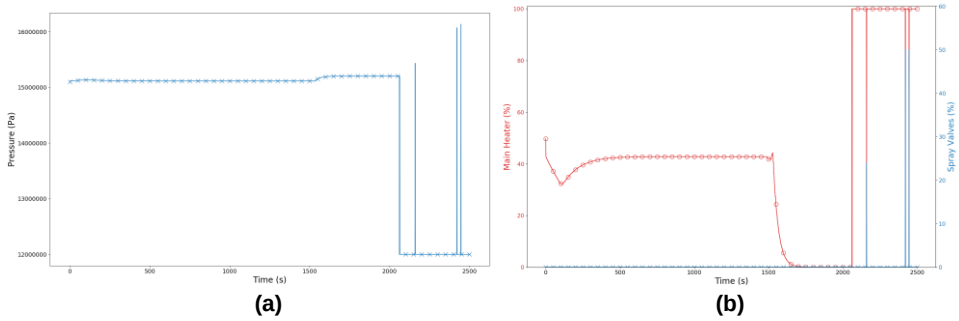

Fig. 6. The (a) pressurizer pressure level (Pa) and (b) main heater (%) and spray valve settings (%) during the FDIA attack on the S7-400 PLC. The FDIA starts at 2060 seconds.

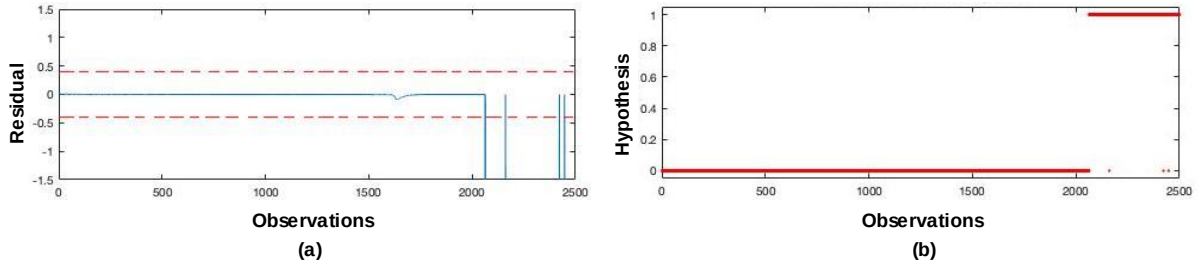

Fig. 7. False data injection scenario detection results using the AAKR model

Fig. 7 shows the detection results during the pressurizer FDIA using the AAKR model. The solid blue line in Fig. 7a show the residuals (i.e., differences) between the predicted values from the AAKR model and the values that are collected (measured) from the attack computer. (For the proposed PLC-based IDS, the measured values would be collected from the *CPU Runtime* by the Goosewolf program.) Meanwhile, the dashed red line in the figure depict a threshold – residuals that exceed these thresholds are deemed to be anomalous, which could indicate an attack. In the MATLAB implementation of the AAKR model, when the residuals exceed the thresholds, a hypothesis of '1' – an alarm – is generated, as shown in Fig. 7b. In the experiments, the AAKR model starts generating alarms at approximately observation 2060, which corresponds to the start of the FDIA. These initial results indicate that the AAKR model can correctly detect the FDIA to the PLC.

6. CONCLUSION AND FUTURE WORK

Nuclear facilities are increasingly making use of digital technologies. This has many benefits, but it also introduces the potential for new forms of cyber-attack. A pernicious form of attack aims to sabotage the operation of facility systems and functions that, for example, ensure continued safe operation of facility processes. To realize these forms of attack, an adversary can manipulate the behaviour of I&C systems, including devices such as Programmable Logic Controllers (PLCs). This is non-trivial and should require an adversary to compromise

several other systems before reaching a target that enables process manipulation. This need to compromise other systems, which are arranged using a defence-in-depth approach, gives a defender the opportunity to detect adversarial behaviour. However, for several reasons, an attack may not be detected prior to targeting I&C systems.

To address this issue, this paper has proposed a system for intrusion detection that can execute on state-of-the-art PLCs. The aim of the system is to provide a last line of defence against an attack. The system, called Goosewolf, can detect the manipulation of user-defined PLC functionality and tampering of data, e.g., as part of a False Data Injection Attack (FDIA). Goosewolf itself is potentially susceptible to attack. Nevertheless, it has several potential benefits and makes it more challenging for an adversary to attack PLCs in a manner that remains undetected. Furthermore, we propose that Goosewolf does not introduce a significant new attack surface, if configured securely. Initial results, which leverage a hardware-in-the-loop testbed that integrates real PLC hardware with a simulated nuclear facility, suggest the system can operate without affecting the control behaviour of the PLC, and an approach to detecting FDIAs is effective and could execute directly on the PLC.

Future work includes a more complete analysis of the AAKR model, which is used to detect FDIAs, to examine its detection performance in the presence of "stealthier" FDIAs. Furthermore, the AAKR model will be implemented in C++ and integrated into the Goosewolf program. Finally, deployment guidelines for Goosewolf will be developed, such that it can be securely integrated into a facility's defensive computer security architecture.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the IAEA as part of the CRP J02008 on Enhancing Computer Security Incident Analysis at Nuclear Facilities.

REFERENCES

[1] JIN, C., VALIZADEH, S., VAN DIJK, M., "Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems", IEEE Industrial Cyber-Physical Systems (ICPS), St. Petersburg (2018).

[2] GARCIA, L., ZONOUZ, S., WEI, D., PFLEGER DE AGUIAR, L., "Detecting PLC control corruption via on-device runtime verification", Resilience Week (RWS), Chicago (2016).

[3] GOH, J., ADEPU, S., TAN, M., LEE, Z.S., "Anomaly detection in cyber physical systems using recurrent neural networks" IEEE 18th International Symposium on High Assurance Systems Engineering (HASE), IEEE (2017).

[4] GOH, J., ADEPU, S., JUNEJO, K. N., and MATHUR, A., "A dataset to support research in the design of secure water treatment systems", International Conference on Critical Information Infrastructures Security, pages 88–99. Springer, (2016).

[5] EGGERS, S.L., "Adapting anomaly detection techniques for online intrusion detection in nuclear facilities", PhD thesis, University of Florida (2018).

[6] ZHANG, F., KODITUWAKKU, H., HINES, J.W., COBLE, J., Multi-layer data-driven cyber-attack detection system for industrial control systems based on network, system and process data, IEEE Transactions on Industrial Informatics **5** 7 (2019) 4362–4369.

[7] ZHANG F., HINES, J.W., COBLE, J., "A robust cybersecurity solution platform architecture digital instrumentation and control systems in nuclear power facilities", Nuclear Technology (2019).

[8] ZHANG, F., HINES, J.W., COBLE, J., "Industrial control system testbed for cybersecurity research with industrial process data", International Congress on Advances in Nuclear Power Plants (ICAPP 2018), Charlotte, NC, USA (2018).

[9] SIEMENS AG, SIMATIC S7-1500/ET 200 CPUs Highlights FW2.5 with TIA Portal V15 (2018), https://www.totallyintegratedautomation.com/wpcontent/uploads/2018/04/Siemens-Advanced-Controllers-Webinar_Apr-25-2018.pdf.

[10] ZHANG, F., PAYNE T., HINES J.W., COBLE, J., "Enhancing the resilience of key Equipment to false data injection attacks in NPPs", ANS Winter Meeting 2019, Washington, DC, USA, November (2019).

[11] ALLISON, D.M., "Utilising the multi-functional platform of the Siemens S7-1518 PN/DP MFP programmable logic controller for security applications", M.Sc. Thesis, Queen's University Belfast, Belfast (2019).

[12] LEI, C., DONGHONG, L., LIANG, M., "The spear to break the security wall of S7CommPlus", Blackhat USA, Las Vegas USA (2017).