PLCBlockMon: Data Logging and Extraction on PLCs for Cyber Intrusion Detection

http://dx.doi.org/10.14236/ewic/ICS2018.12

PLCBlockMon: Data Logging and Extraction on PLCs for Cyber Intrusion Detection

Mislav Findrik and Paul Smith
AIT Austrian Institute of Technology
Center for Digital Safety & Security
Vienna, Austria
firstname.surname@ait.ac.at

Kevin Quill and Kieran McLaughlin
Queen's University Belfast
Center for Secure Information Technologies
Belfast, UK
kieran.mclaughlin@qub.ac.uk

The threat landscape for industrial control systems is ever-expanding and these systems have proven to be attractive targets for cyber attackers. Programmable Logic Controllers are major components in ICSs and hence need to be well-protected and monitored. By examining the existing research in this field we found that there is a void in comprehensive analysis of data logging and extraction features on industrial devices. However, analysis of these features and evaluation of their applicability for cyber intrusion detection would significantly facilitate their adoption by intrusion detection tools. In order to close the gap, we analyzed the logging and extraction capabilities of the Siemens S7-1200 PLC and HMI panel. We implemented a PLC logic for data logging called PLCBlockMon. In this paper, we provide guidelines for its usage and demonstrate its applicability for cyber intrusion detection in selected scenarios.

Keywords: *Industrial Control Systems, Cyber Security, Logging and Extraction, PLC, Intrusion Detection*

1. INTRODUCTION

Industries are increasingly relying on machines and automated processes. For these machines to operate reliably, Industrial Control Systems (ICS) have been designed to operate with high availability, reliability and safety, which leaves the security of these systems typically lacking. The ICS industry has in the past relied on isolation ("air gaps") to make their ICS network secure. However, in the last decade, infrastructures of ICS networks have started to become tightly integrated with standard IT networks. This has the advantage of allowing for remote access and control of devices by engineers leveraging the TCP/IP protocol suite and Commercial Off-The-Shelf (COTS) hardware. Besides these benefits, this has also opened the doors for malicious users to deploy remote attack vectors on insecure ICSs.

There are multiple examples of cyber-attacks on ICSs, the most famous of them is the Stuxnet. This malware was designed to destroy gas centrifuges in Iran's uranium enrichment plant, by over-pressurizing them and over-speeding their rotors (Lagner (2011)). This was achieved by modifying the control logic on certain PLC controllers in the plant. Other examples of ICS attacks with physical consequences are two cyber-attacks on Ukranian electricity distribution networks. One of them occurred in December 2015, which caused a large power blackout (E-ISAC report (2011)), while the other one, named Crashoverride (Dragos Report (2017)), occurred almost exactly one year later, and lead to an hour long power outage.

In order to detect cyber-attacks in ICS networks many security companies now offer network traffic monitoring solutions (e.g., Silent Defence (2018), Dragos Platform (2018), SCADAguardian (2018), Darktrace Industrial (2018)). Also, many researchers are proposing different analysis methods on industrial protocols (Ponomarev (2016)) providing new ways for attack detection. However, monitoring of end-devices, Programmable Logic Controllers (PLCs), which are responsible for the control of industrial processes, for security purposes is largely an unexplored area. In fact, these devices are mainly monitored by control engineers during regular check-ups, in order to query the status of the PLC (e.g., cycle time, operational mode, etc.) (Copy (2015)). This is typically done using vendor-specific tools for PLC diagnostics.

In this paper, we explore different data logging and extraction methods on industrial devices that are available in our ICS testbed and explore their usage for cyber-intrusion detection purposes. Therefore, in the first section, we introduce the experimental ICS environment used in the experiments. In Section 3, an analysis of different data logging and extraction capabilities is given. The main contribution of this paper is a logic for data logging on a PLC, which is presented in Section 4, along with an analysis of its application scenarios for cyber-intrusion detection in Section 5.

2. EXPERIMENTAL ICS ENVIRONMENT

ICSs are large and complex environments that are commonly used in industries, such as manufacturing, electric power generation, chemical plants, oil refineries, and wastewater treatment facilities for process control. Implementing a full-scale ICS environment for solely research purposes can be an overwhelming and expensive task, hence for conducting scientific experiments it is common to construct ICS testbeds using only a few representative pieces of equipment that is necessary answer research questions (Survey (2014)). In order to analyze data logging and extraction features on industrial devices, we have constructed a small-scale testbed that is comprised of Siemens software and hardware components that are used in real ICS facilities. Siemens is one of the biggest ICS vendors alongside Honeywell, Rockwell/Allen-Bradley, and Schneider Electric, thus its equipment has been chosen for the testbed. Siemens devices included in the testbed are an S7-1200 controller, HMI panel KTP-700, and a Windows 7 desktop machine with TIA portal v14 software application, which is used as an engineering workstation for PLC/HMI programming and configuration. While the experiments conducted in this work are limited to the S7-1200, this device offers insights into the larger S7 product suite, as they all share similar features. The architecture of our testbed is designed to mirror local monitoring and control of a physical process, therefore representing cell zones of a general ICS architecture corresponding to Levels 0 to 2 in the Purdue model (Obregon (2014)), as shown in Fig. 1.

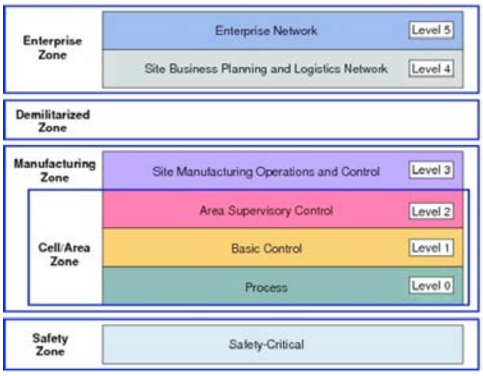


Figure 1: *The Purdue Reference Architecture for ICS.*

Figure 2 illustrates the architecture of our testbed. In the future, we plan to extend the testbed to include Level 3 systems that are responsible for plant operations, such as a data historian and desktop-based HMI, which are responsible for plant-wide data logging across all cell zones, and can also be utilized for cyber-intrusion detection.

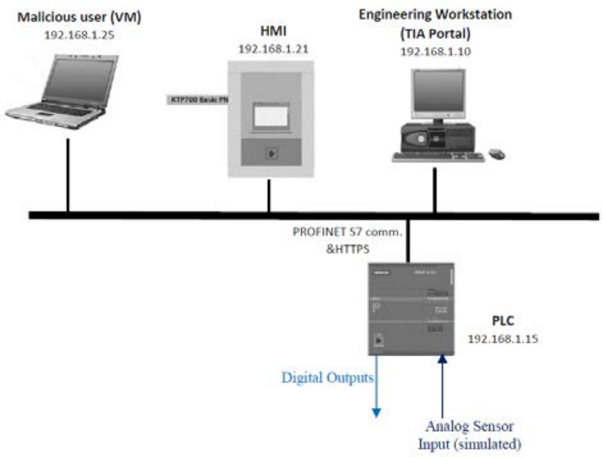


Figure 2: *Architecture of the experimental ICS testbed.*

All the aforementioned devices are connected to a network switch. In addition, a laptop running Kali Linux (Kali (2018)) in a virtual machine is used for the generation of experimental attack scenarios, which are described in later sections. The S7-1200 PLC used as a process simulator and the water level controller of a coolant tank. The water tank has three valves (fill, drain and coolant valve) with a fixed flow rate. The current water level is simulated on the PLC as an analog input, and its value is calculated based on the current status of the valves (ON or OFF) and knowledge of their flow rates. Figure 3 shows the visualization of the process on the HMI display.

The tank water is used as a coolant for another process, hence it is assumed that the coolant valve is turned ON or OFF at random times, in order to simulate the demand for coolant. To balance out the lack of water in the tank, the fill valve is turned on after the water level drops under a certain level (the orange indicator on the tank) and turns off after a certain water level is reached (the red indicator on the tank). The drain valve is not automatically controlled and it serves for occasional maintenance related tank emptying. Control actions can also be manually overridden by the operator using the HMI screen buttons.

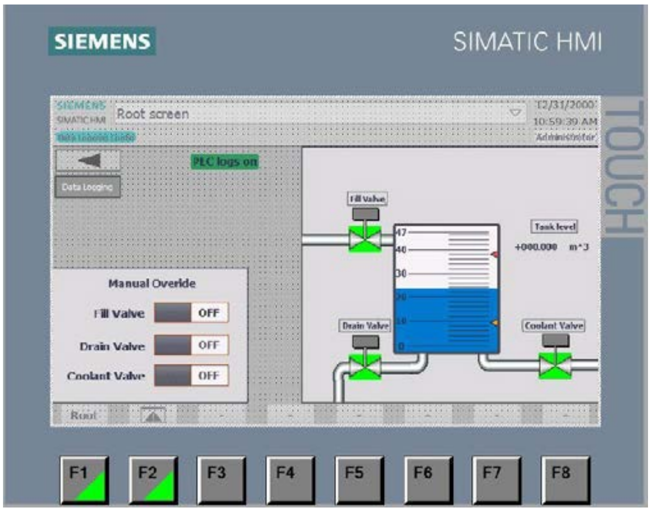


Figure 3: *HMI visualization of the tank water level process.*

3. DATA LOGGING AND EXTRACTION CAPABILITIES

This section will introduce the monitoring functions that are available on the testbed devices i.e., the S7-1200 controller and the HMI panel. Furthermore, the specific capabilities of the logging functions will be elaborated alongside their security features against manipulation and their capabilities for extraction of logged data.

3.1. PLC Data Logging and Extraction

3.1.1. PLC Logging Data Blocks

The PLC logging data blocks allow for logs of logic tags and data to be stored in the local memory. The data logging blocks included in the TIA portal provide the basic functions needed to build a logging system. This includes blocks to create, open, clear, delete and write to the log files. This leaves implementation of the logging to be decided by the programmer. The three main functions that need to be programmed to create a fully working logging system are data transfer to DataBlock (DB), handling of the file state, and triggering when to sample the data. The basic functionality of the blocks creates a single circular log. New log files can be created with the *NewFile* block to prevent overwriting data; however, this would be limited by the small internal memory size. Another consideration with the memory size is with the log sampling rate. Using a cyclic pulse to trigger this would create a lot of duplicate data and hence would overwrite data much faster. Only recording tag values when a change occurs is a much more efficient implementation. This block can accept most data types, and therefore can record most of the tags, DB values and memory locations natively. One known exception to the data types accepted is Boolean, but this can be overcome by storing the value in a different type. The logs are formatted using CSV log format, which makes parsing information from the logs very easy. As the creation of this log requires programming the functionality into the ladder logic, this logging method requires the most effort to implement out of all the features looked at. This change of logic would result in the need for system validation. For this reason, the use of this logging feature would be most applicable for facilities being designed from the ground up or that are in the process of replacing equipment. As these logs are programmed in the logic of the PLC, they are only as secure as the load memory of the PLC. Fortunately, more recently produced Siemens SIMATIC S7 controllers (e.g. the S7-300 and S7-400 PLC series delivered since April of 1998, Siemens Security Bulletin (2011)) offer integrated protection mechanisms for restricting access to memory areas. Memory access can be set to three different levels (write, write/read and complete protection) and it is password protected (S7 security (2000)). Hence, the memory of the logs programmed in the logic of the PLC should be adequately protected against potential malicious alterations by setting at least protection level against writing.

3.1.2. Diagnostic Buffer Logs

The *Diagnostic Buffer* is a log of PLC system level events and is stored locally in each CPU. Originally, the diagnostic buffer is designed to provide users with logs for rapid detection of errors and evaluation of events that caused PLC stoppage (i.e., PLC entering in STOP mode) (S7-1200 (2014)). The log has a size of 50 events, in the case of S7-1200, of a circular log format. A configuration setting can be enabled which allows for some system events of the same type to be collated together, increasing the time until old events get overwritten (S7-1200 (2014)). The features which are recorded in the Diagnostic Buffer are: (i) error events (e.g., CPU, IO, module, submodule errors, network connectivity problems), (ii) CPU mode changes (e.g., CPU RUN/STOP changes), and (iii) user-defined messages (e.g., water tank level too high notifications).

The Diagnostic Buffer can be accessed in several ways using existing Siemens features. These are the integrated web server, TIA portal and via the HMI panel, if configured to do so. To use the web server to view this information, the user rights need to be expanded from the minimum to include query diagnostics. This gives the user access to the Diagnostic Buffer, but also the resource consumption of the device, connection resource statistics and information about the module. The diagnostic information can also be requested in the ladder logic using the *GET_Diag* block (S7 diagostics (2014)).

This would then allow the programmer to use any of the existing communication blocks (e.g., SMTP over TLS) to transmit the diagnostic information. The TIA portal is intended as a programming and diagnostic tool and there can only be one connection active at any time to a given PLC. For this reason, this method is less practical than the other options, as it creates more challenges for the engineers working with the devices. Use of the desktop HMI to save the logs, appears to be the most practical method. The Diagnostic Buffer can be cleared by resetting the PLC to factory settings if no memory card is installed; however this is not possible without a password if the minimum access protection level is set on the controller.

3.1.3. Extraction of Log Information

The main method for extracting the PLC logs remotely is with the built-in web server. To do this, the user requires the minimum rights plus read files access. This allows the user to download the data logs and receive data from the device, but not to delete data. The web server has an automatic refresh setting and the minimum rate is every 5 seconds. This means that the size of the data logs should be large enough to last for the minimum refresh rate that is set in the web server. Other options for sending the data to a remote system include using the *T(U)SEND_C* function blocks which supports ISO-on-TCP, TCP or UDP (S7 User Communication (2000)). For example, the UDP function block has been used to implement log extraction in SYSLOG format (S7 Syslog (2018)). While these protocols are not secured, they might be used in conjunction with a network intrusion detection system identifying potential manipulations of logs in transmission. Another way to securely extract logs is by sending them over secure email connections (SMTP over TLS), which is available in the function block *TMAIL_C* (Siemens SMTP, (2004)). This option is a good alternative for cases when the web server is not enabled. Lastly, if the logs are stored on a memory card used by the PLC, they could also be manually extracted. This is highly impractical for real facilities and might only be useful for forensic purposes in a case where the PLC has crashed.

3.2. HMI Data Logging and Extraction

The HMI panel has the capability to locally store logs on a connected Siemens USB device or SD card. This feature is called *Historical Data*, and offers several different options for configuring the behaviour of the logs. The logs stored on the HMI consist of recorded values of HMI tags and the PLC tags which are read by the HMI. These logs have three options for when the tag values are recorded and this can be set individually for each tag being recorded. The logging options are: on change of the tag value, cyclically with a set period, or can be triggered by the change in a different tag. The HMI does not rely on running logic to read the PLC memory and hence can continue to log memory values while the PLC is in stop mode. A limitation of this feature is the HMI will not be able to log a pulse in a value on the PLC that lasts less than the update cycle of the HMI tags. This would need to be considered for the configuration of the HMI, as any sensitive control signal that can cause a change would need the update cycle period to be set to the minimum.

For accessing the logs on the SIMATIC Basic HMI panels, there is only one option, which is using a removable USB key. However, this is not practical for a large facility or for use in an automated system. The advanced HMI panels (i.e., the SIMATIC HMI Comfort panel series) do have the option to enable a web server for remote data extraction (Siemens HMI Web Access (2000)). The desktop version of the HMI (Siemens WinCC (2000)), typically used for implementation of Level 3 operators screens, has capabilities for logging PLC/HMI tags and alarms on disk memory, thus offers much greater options for saving and transmitting the logs to a detection system. Besides PLC alarms, the Basic HMI panel also has the ability to display analog (triggered by exceeding/falling below a previously defined limit value) and bit alarms that are defined by users on the HMI screen, and store their history in a cyclic buffer. The integrity of the logs on the HMI panel and similarly the desktop version is reliant on the configuration of the HMI being secure. A summary of the these features can be seen in Table 1.

4. PLCBLOCKMON: DATA LOGGING ON PLCS

In this section, we describe implementation of our data logging block, called PLCBlockMon. PLCBlockMon[1] is designed to log different PLC tags. The idea is to log only tags which are capturing the most important states of the process being controlled and respective PLC input/output values influencing the process states. Therefore, exactly which tags should be logged is dependent on the process being controlled, requiring it to be tailored to individual PLC logic. Furthermore, in this section, we will illustrate the process of configuring PLCBlockMon for logging the states of the water tank process.

The control ladder logic programmed in the PLC uses 3 digital inputs, 3 digital outputs, 1 simulated analog input and 3 HMI (digital) inputs. The three HMI inputs are the buttons which can be seen in Figure 3. The three digital inputs to the PLC indicate

[1] The source is publicly available at `https://github.com/CSR-AIT/PLCBlockMon`

	PLC Block Logs	**HMI Panel logs**	**PLC Diagnostic Logs**
Storage Location	Load memory of PLC	Siemens memory connected to the HMI	Load memory of PLC
Secure extraction methods	Memory card, Web Server or SMTP over TLS	USB Stick	TIA Portal, HMI or Web Server
Other extraction methods	TCP or UDP communication logic blocks	-	TCP or UDP communication logic blocks
Logging Scope	Logic and Memory Locations on PLC	HMI/PLC tags and alarms	Diagnostic events
Trigger method	Cyclic or on programmed trigger	Cyclic, On Demand or On Change	Event Triggered

Table 1: *A summary of logging features that are readily available on industrial testbed devices.*

positions of the valves, whereas digital outputs are used for setting their position. Lastly, the analog input to the PLC is a simulated value of the tank level. These are the ten most important tags selected to log, since they influence the PLC state thus the process.

After logging tags have been chosen, the control logic needs to be augmented with additional instructions for capturing tag changes in memory locations that are accessible by PLCBlockMon. In Figure 4, this is illustrated on a portion of the control logic responsible for switching the fill valve ON. The fill valve can be switched ON by an operator via the HMI (*HMI_Fill_EN*) input tag or automatically by the SR bistable logic, if water level is under a certain threshold.

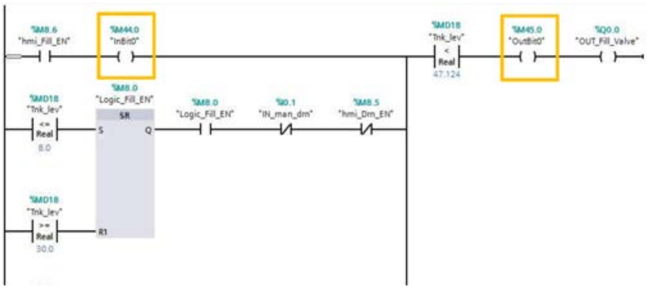


Figure 4: *PLC control logic augmented with tag saving instructions.*

Changes in the HMI input (i.e. *HMI_Fill_EN* tag) and the digital output for the fill valve control are saved on memory location for outputs (M44.0) and inputs (M45.0) - by instructions highlighted in yellow rectangles. After the changes have been captured, logs need to be saved on the load memory of the PLC. This is achieved by two main internal blocks of the PLCBlockMon - *Trigger* and *Data Log*, shown in Figure 5. PLCBlockMon needs to be placed at the end of the main control logic (i.e., organization block). By placing PLCBlockMon at the end it is ensures that the control logic will no longer change input/output tags in a given execution cycle.

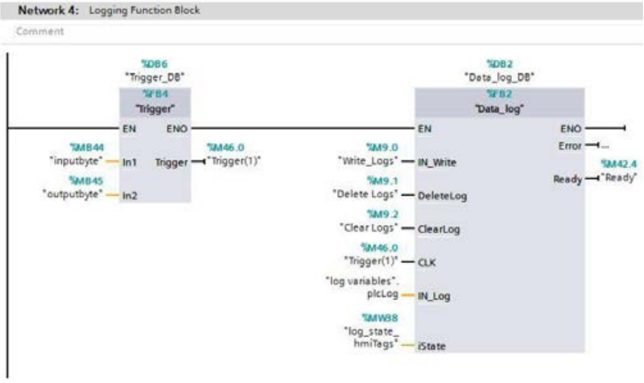
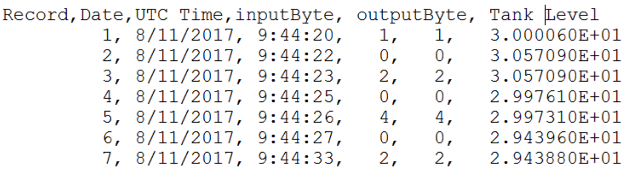


Figure 5: *Main functional blocks of PLCBlockMon.*

The trigger block first compares input and output bytes with their previous state saved internally. If there are any changes detected, the trigger for the subsequent data logging block is activated and saves the current changes in the CSV file residing in memory. For the sake of compression, digital inputs and outputs are logged into a byte array and represented in the file as integer values. Figure 6 provides an example of such a log file.



```
Record,Date,UTC Time,inputByte, outputByte, Tank Level
         1, 8/11/2017, 9:44:20,    1,    1,    3.000060E+01
         2, 8/11/2017, 9:44:22,    0,    0,    3.057090E+01
         3, 8/11/2017, 9:44:23,    2,    2,    3.057090E+01
         4, 8/11/2017, 9:44:25,    0,    0,    2.997610E+01
         5, 8/11/2017, 9:44:26,    4,    4,    2.997310E+01
         6, 8/11/2017, 9:44:27,    0,    0,    2.943960E+01
         7, 8/11/2017, 9:44:33,    2,    2,    2.943880E+01
```

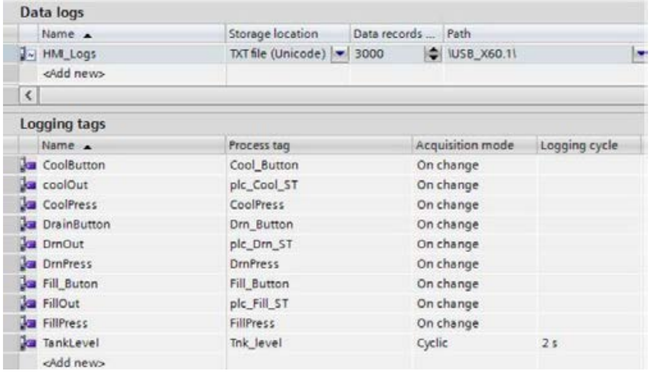
Figure 6: *Log file generated by PLCBlockMon.*

5. CYBER INTRUSION DETECTION

Our intrusion detection technique is based on the comparison of log files generated by PLCBlockMon on the PLC and log files collected on the HMI panel. This section first explains the logging method used on the HMI panel during the experiments, after which the experimental validation results are presented and explained.

5.1. HMI-based data logging implementation

Logging on the HMI panel was made available by a connected Siemens USB device. This allowed different tags to be logged using the HMI Historical Data build-in function. The tags to be logged and the type of acquisition mode to be selected were configured via the TIA portal. This can be seen in Figure 7.



Data logs

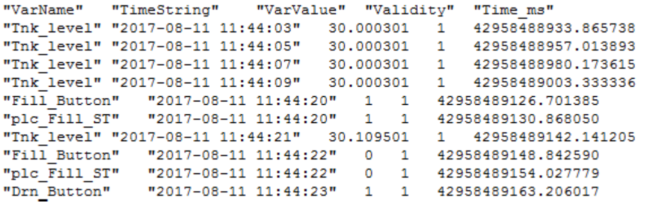
Name ▲	Storage location	Data records ...	Path
HMI_Logs	TXT file (Unicode)	3000	\USB_X60.1\
<Add new>			

Logging tags

Name ▲	Process tag	Acquisition mode	Logging cycle
CoolButton	Cool_Button	On change	
coolOut	plc_Cool_ST	On change	
CoolPress	CoolPress	On change	
DrainButton	Drn_Button	On change	
DrnOut	plc_Drn_ST	On change	
DrnPress	DrnPress	On change	
Fill_Buton	Fill_Button	On change	
FillOut	plc_Fill_ST	On change	
FillPress	FillPress	On change	
TankLevel	Tnk_level	Cyclic	2 s
<Add new>			

Figure 7: *Development Phase process steps for developing secure IoT applications.*

The tags logged on the HMI are: (i) *Button* tags linked to a tag on the PLC and hence if the tag is changed on one device the tag on the other device is also changed, either by the HMI sending a request to change the tag on the PLC or if the tag was changed on the PLC then the HMI will change its value when it reads the PLC tags - also logged by PLCBlockMon; (ii) *Press* tags internal to the HMI indicating when the switch is manually changed by an operator; (iii) *Out* tags directly linked to the digital control output tags on the PLC also logged by PLCBlockMon; (iv) *TankLevel* tag linked to the PLC tank level variable. It is necessary to log the same tags via PLCBlockMon and on the HMI in order to be able to compare their values.



```
"VarName"   "TimeString"    "VarValue"  "Validity"  "Time_ms"
"Tnk_level" "2017-08-11 11:44:03"   30.000301   1   42958488933.865738
"Tnk_level" "2017-08-11 11:44:05"   30.000301   1   42958488957.013893
"Tnk_level" "2017-08-11 11:44:07"   30.000301   1   42958488980.173615
"Tnk_level" "2017-08-11 11:44:09"   30.000301   1   42958489003.333336
"Fill_Button"   "2017-08-11 11:44:20"   1   1   42958489126.701385
"plc_Fill_ST"   "2017-08-11 11:44:20"   1   1   42958489130.868050
"Tnk_level" "2017-08-11 11:44:21"   30.109501   1   42958489142.141205
"Fill_Button"   "2017-08-11 11:44:22"   0   1   42958489148.842590
"plc_Fill_ST"   "2017-08-11 11:44:22"   0   1   42958489154.027779
"Drn_Button"    "2017-08-11 11:44:23"   1   1   42958489163.206017
```

Figure 8: *Log file generated by the HMI.*

For tag values which are recorded cyclically, if the network connection is interrupted, then the HMI will automatically log the last known value but set the validity flag to 0 to indicate that this is not a up-to-date value (as shown in Figure 8).

5.2. Experimental Validation

This section will discuss the results of the experiments carried out. Validation of the resource usage and network resilience of the logging features will be examined first, and then three attack scenarios will be used to illustrate applications of the logging features on the example of the water tank process.

5.2.1. DoS Attack Detection

To examine the accuracy of the logs under strenuous network conditions, the logs were recorded under a DOS attack that is generated using hping3 tool (Hping (2014)) from the malicious user laptop shown in Figure 2. The HMI historical data logs rely on the information transmitted to it by the PLC and hence are susceptible to losing data when the network connection is interrupted. As can be seen in Figure 9, where the data was set to record when the value changes, the HMI does not change the value in the logs or record a default value when the network connection is lost. The DOS attack was started at 100 seconds and finished at 200. This is the reason for the delay in the HMI FillOut being set to zero, as it was waiting for the connection. For tag values which are recorded cyclically if the network connection is interrupted, then the HMI-based log will record the last known value but set the validity flag (shown in Figure 8) to 0 to indicate that this is not a valid value. In addition to log information provided by PLCBlockMon and the HMI, this attack could be correlated with information from a network intrusion detection system, if one is deployed on the same industrial control network.

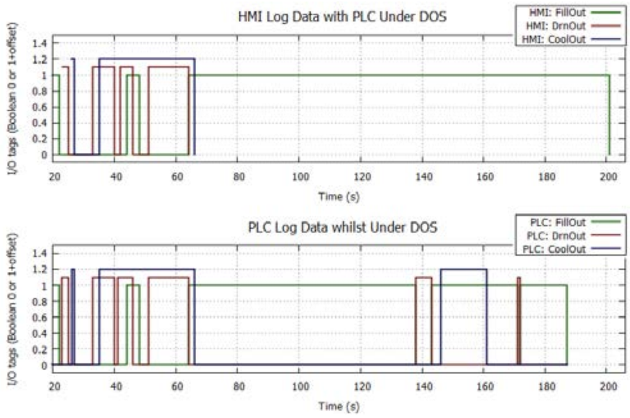


Figure 9: *PLC and HMI tag values during DoS experiment.*

5.2.2. Protocol Attacks on the PLC

This scenario is based on a vulnerability being identified by an attacker or similarly, where a known protocol vulnerability can be used, due to older firmware on operational devices. Security issues with the protocol can allow an attacker to replay, manipulate packets, or even change memory areas, and by doing so change the operation of the PLC. For example, exploitation of security vulnerabilities on the S7 protocol have previously been demonstrated in following papers (Bresford (2011), Cheng et al (2017)). The stages of such attacks could occur as follows: (i) to produce a specific modification to a process the attacker would need to have knowledge of what the PLC is doing and observe the packets to build up a profile of the memory locations being set over the network, (ii) having built up a behavioral profile and gained knowledge of the PLC configuration, and attacker could capture the specific packet which is intended to replay, modify it to avoid rejection and forward it to the PLC. Or in case the memory area is not write protected, the attacker could simply issue memory force commands. Where the vulnerability in the protocol allows for packet manipulation, an additional step in this scenario could be to modify the return packets to the HMI to hide the changes caused by the replay attack. In this case, if done correctly, then both logs would look normal compared to each other, however the PLC logs would show process control inconsistencies e.g., fill valve opened when it should be closed (i.e., when the water level is above a certain threshold). However, if the attacker is forcing PLC tags which are normally changed only via the HMI button, their actions could be detected by the log comparison, as shown in Figure 10.

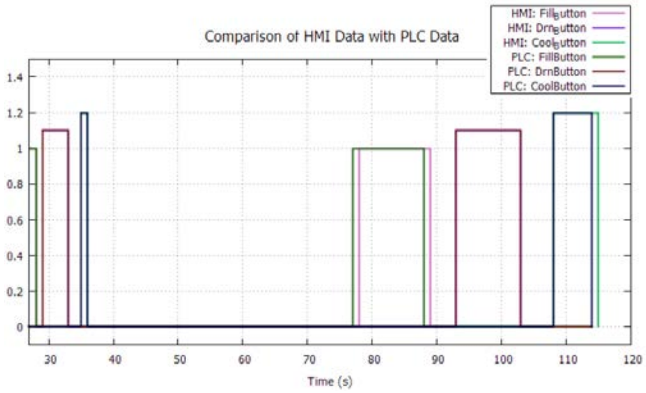


Figure 10: *Normal button changes (left) and changes caused by forcing PLC memory (right).*

In reference to the operating conditions replicated in the testbed, the ability of an attacker to modify the memory tag was simulated using the force function in the TIA portal. A comparison of the HMI Historical Data and the PLC logs can now be used to identify if the memory change was initiated from the HMI directly. The difference in timestamps can indicate the direction of data flow, this can be seen in Figure 10, where the delay on the right-hand pulses are larger than the pulses on the left-hand where no delay is visible. This delay is the result of the HMI not changing the log until the next request for data, which is a cyclic process of a pre-set period, and hence has a longer delay than immediate update for legitimate changes from the HMI. The information shown in the figure could be provided to a sequence-aware IDS (Caselli et al (2015)), which can identify intrusions based on the execution order of the events.

Besides looking at the sequence of the events, a third way in which the logs could be used for the detection of attacks manipulating HMI control packets, is if the HMI is configured to log button presses as well as lot changes in the tag controlled by that button - as explained in Section 5.1. If this is implemented and the tag on the PLC is changed using the force function or with replay attack, then the press will not be logged in the HMI Historical Data. The difference between a normal and malicious button change would be captured in HMI Historical Data logs and detected by analyzing the execution order of the events.

5.2.3. Malicious Logic Upload Detection

This scenario is designed to show the limitations of PLCBlockMon and Historical Data logs, while illustrating how the Diagnostic Buffer and the resource statistics available from the PLC can be used to look for unusual changes. The malicious logic upload attack considers the following steps: (i) an attacker sets up a phishing attack or installs a key logger in the engineer’s laptop, (ii) the engineer connects to PLC, entering password to carry out maintenance or view diagnostic information with TIA portal, (iii) the attacker captures the password and then connects to PLC with a different the TIA portal and downloads the existing logic, (iv) the attacker can also view the real-time operation of the system to build a profile, (v) once the attacker has the logic of the PLC, they can modify it to carry out their objective while simulating and displaying normal operating conditions, (vi) the attacker uploads the new logic. To achieve the steps described without modifying the HMI configuration requires an attacker to modify the mapping of the digital output pins on the PLC. To do this, PLC must do a full restart which clears the memory of the tags and databases. The log files are not removed at this point, however PLCBlockMon logic will fail to open the old file and hence it will override with a new file with the same name. Once the logic is compromised, the accuracy of any programmed logs is unclear, as an attacker can program the PLC logs to look as they want. This can be seen in Figure 11, where the logs under normal operating conditions can be seen on the left of the graph where the changes in tank level correspond to the change in the I/O tags.

The same behaviour linked to the normal PLC operations can be seen on the right side of the upper

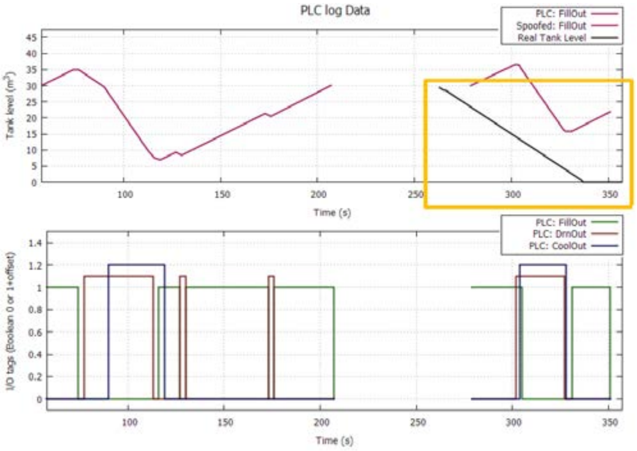


***Figure 11:** Plot of PLCBlockMon log data before and after logic is modified.*

graph. However, in this case, the purple values are not representing the true state of the PLC anymore (the true state is represented with a black line in the upper graph). The data on the right of the graph was recorded after the logic was modified and represents what was shown on the HMI and in the new fake PLC logs. The gap of information in the centre of the graph represents when the logic was being updated and the PLC was in STOP mode. In this restart process, where the PLC configuration is changed, the log files are cleared of the old data and this is the reason the gap is not interpolated by the plotting tool. In an operational environment, an intentional restart would require a system shutdown or some other measure to prevent system instability while the PLC is offline. For this reason, and the fact that the logic of PLC in this environment is very unlikely to change, makes a restart very suspicious. Thus clearing of the log file would be an unusual event to occur in a running system and might be an indication of a malicious logic upload attack. Moreover, the extra logic required to simulate the real process behavior increases the memory usage of the PLC - this change could also serve as an indication of the attack. Additional indications could also be found in the diagnostic buffers of the HMI and PLC. While the PLC is in STOP mode, the HMI diagnostic buffer would log network connection problems and the PLC restart would be logged on the PLC diagnostic buffer. A monitoring system with access to the diagnostic buffers could use these indications as a trigger for a logic validation system similar to that proposed by (Govil et al (2017)).

5.2.4. Resource Usage Validation

For industrial control applications, the real-time performance is of the utmost importance, hence the logging logic should not have considerable impact on the real-time performance of the PLC running control logic. Tests of the PLCBlockMon logging function block which was added to the normal tank control logic showed that the cycle time was the same compared against the PLC with no logging. However, more load memory was used when PLCBlockMon was running. For the log size of 100 entries, 6KB log file was created on the PLC allowing 2 byte values (maximum of 14 Boolean values) and one real value to be stored (enough to store 9 boolean digital inputs and tank level value).

6. CONCLUSION

The aim of this paper is to evaluate the potential for using the diagnostic and logging features that are built into widely-deployed contemporary industrial devices, specifically PLCs, for cyber intrusion detection. We have found little related research work (apart from Caselli et al (2015)) that looks into the use of ICS-specific functions and data logs for security purposes which motivated us to conduct this study. In particular, we have comprehensively analyzed the following options: the HMI Historical Data, the PLC Data Logging Blocks and the PLC Diagnostic Buffer, due to their features for relevant ICS data logging along with various possibilities for their extraction and usage for intrusion detection. Moreover, we have implemented a PLC logic allowing data logging on the Siemens S7-1200 PLC, called PLCBlockMon, explained its implementation and demonstrated the usage for intrusion detection in conjunction with the HMI Historical Data based on log comparison method. In addition, we have shown how PLC data logs can be successfully faked by uploading a well-constructed malicious logic, but not without leaving traces that might indicate such activity. It is also worth mentioning that comparison of logs from different devices requires time synchronization in the system, since timestamps are generated on the individual devices. Further investigation would be needed to evaluate the resilience of the logging technique to timing attacks such as NTP amplification attacks (NTP Attacks (2016)). Another potential area for further research would be to evaluate the application of PLCBlockMon for ICS forensic purposes (Van Vlie et al (2016)), its viability for deployment in practical circumstances, expand current work to other Siemens industrial devices and to consider other ICS equipment manufactures. We hope the findings from this research will serve as a base for future cyber-security studies on ICS intrusion detection, by providing collated information sources on the logging and extraction features available on wide suite of S7 devices.

ACKNOWLEDGMENT

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

Siemens, Sending Emails over Secure Email Connections with S7-1500 and S7-1200, `https://cache.industry.siemens.com/dl/files/803/46817803/att_926218/v2/46817803_EMail_with_CP1543-1_en.pdf`

Siemens, SIMATIC HMI Comfort Panels, `https://cache.industry.siemens.com/dl/files/153/109476153/att_848919/v1/109476153_Remote_Panels_Webserver_DOKU_en.pdf`

Siemens, Security with SIMATIC controllers, `https://cache.industry.siemens.com/dl/files/010/90885010/att_876214/v1/77431846_Security_SIMATIC_DOKU_V20_en.pdf`

Siemens, Configuring Messages and Alarms in WinCC (TIA Portal), `https://support.industry.siemens.com/cs/document/62121503/configuration-of-messages-and-alarms\penalty-\@M-in-wincc-(tia-portal)?dti=0&lc=en-WW`

Langner R., “Stuxnet: Dissecting a Cyberwarfare Weapon,” in IEEE Security & Privacy, vol. 9, no. 3, pp. 49-51, May-June 2011.

E-ISAC, Analysis of the Cyber Attack on the Ukrainian Power Grid Defense Use Case, March 18, 2016

Dragos, Crashoverride: Analysis of the Threat to Electrical Grid Operations, `https://dragos.com/blog/crashoverride/CrashOverride-01.pdf`

SecMatters, SilentDefence Datasheet, `https://www.secmatters.com/hubfs/Security_Matters-March2017/PDF/SilentDefense-Datasheet.pdf`

Dragos, Dragos Platform Datasheet, `https://dragos.com/media/Dragos_Platform_Data_Sheet.pdf`

Darktrace Industrial Immune System, `https://www.darktrace.com/resources/ds-iis.pdf`

Nozomi Networks SCADAGuardian Datasheet, `https://www.nozominetworks.com/downloads/US/Nozomi-Networks-SG-Data-Sheet.pdf`

Copy B. , Zimny M. , Milcent H., “Standards-based open-source PLC diagnostics monitoring,” in Proceedings of ICALEPCS2015, Melbourne, Australia

Ponomarev S. and Atkison T., “Industrial Control System Network Intrusion Detection by Telemetry Analysis,” in IEEE Transactions on Dependable and Secure Computing, vol. 13, no. 2, pp. 252-260, March-April 1 2016.

Obregon L., “Secure Architecture for Industrial Control Systems,” `https://www.sans.org/reading-room/whitepapers/ICS/secure-architecture-industrial-control-systems-3632`

Siemens, SIMATIC S7-1200 System Manual, Siemens, Nrnberg, 2014.

Kali Linux, `https://www.kali.org/`

Kali Tools, hping3 Package Description, `https://tools.kali.org/information-gathering/hping3`

Siemens, Diagnostics in User Program with S7-1500, 2014.

Holm H., Karresand M., Vidstrm A., Westring E. (2015) A Survey of Industrial Control System Testbeds. In: Buchegger S., Dam M. (eds) Secure IT Systems. Lecture Notes in Computer Science, vol 9417. Springer, Cham

Siemens, Open User Communication with TSEND_C and TRCV_C, `https://cache.industry.siemens.com/dl/files/808/67196808/att_108115/v2/net_s7-1200_isoontcp_en.pdf`

Siemens, Sending SYSLOG messages with a SIMATIC S7 CPU, `https://support.industry.siemens.com/cs/document/51929235/-sending-syslog-messages-with-a-simatic-s7-cpu?dti=0&lc=en-WW`

Siemens, Siemens Security Bulletin Response to ICS Alert (ICSA-11-223-01A), `https://www.industry.siemens.com/topics/global/en/industrial-security/news-alerts/Documents/Summary_on_ICS_Alert_ICSA-11-223-01A.pdf`

Beresford D., Exploiting Siemens Simatic S7 PLCs, in Black Hat USA+2011, Las Vegas, NV, USA, 34 Aug. 2011. [Online]. `https://media.blackhat.com/bh-us-11/Beresford/BH_US11_Beresford_S7_PLCs_WP.pdf`

Cheng L. et al, The spear to break the security wall of S7CommPlus, in Black Hat EU 2017. [Online]. `https://www.blackhat.com/docs/eu-17/materials/eu-17-Lei-The-Spear-To-Break%20-The-Security-Wall-Of-S7CommPlus-wp.pdf`

Caselli M. et al, Sequence-aware Intrusion Detection in Industrial Control Systems. In Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS '15). ACM, New York, NY, USA, 13-24. DOI=http://dx.doi.org/10.1145/2732198.2732200

Govil, Naman, Anand Agrawal, and Nils Ole Tippenhauer. "On Ladder Logic Bombs in Industrial Control Systems." arXiv preprint arXiv:1702.05241 (2017). `https://arxiv.org/pdf/1702.05241.pdf`

US-CERT Alert (TA14-013A), NTP Amplification Attacks Using CVE-2013-5211, `https://www.us-cert.gov/ncas/alerts/TA14-013A`

Van Vlie et al. "Forensics in Industrial Control System: A Case Study." (2016), `https://arxiv.org/ftp/arxiv/papers/1611/1611.01754.pdf`