Detecting Anomalous Programmable Logic Controller Events Using Machine Learning

Ken Yau, Kam-Pui Chow

Chapter 5

DETECTING ANOMALOUS PROGRAMMABLE LOGIC CONTROLLER EVENTS USING MACHINE LEARNING

Ken Yau and Kam-Pui Chow

Abstract Industrial control system failures can be hazardous to human lives and the environment. Programmable logic controllers are major components of industrial control systems that are used across the critical infrastructure. Attack and accident investigations involving programmable logic controllers rely on forensic techniques to establish the root causes and to develop mitigation strategies. However, programmable logic controller forensics is a challenging task, primarily because of the lack of system logging. This chapter proposes a novel methodology that logs the values of relevant memory addresses used by a programmable logic controller program along with their timestamps. Machine learning techniques are applied to the logged data to identify anomalous or abnormal programmable logic controller operations. An application of the methodology to a simulated traffic light control system demonstrates its effectiveness in performing forensic investigations of programmable logic controllers.

Keywords: Programming logic controllers, forensics, machine learning

1. Introduction

Industrial control systems, which are widely used in the critical infrastructure, contribute to safety and convenience in every aspect of modern society. These systems have served reliably for decades, but a changing technological environment is exposing them to risks that they were not designed to handle [4]. In particular, their reliance on networking technologies, including remote access and control over the Internet, significantly increase the likelihood of attacks.

A common approach when investigating attacks and anomalies involving an industrial control system is to concentrate on the central server of the digital control system or supervisory control and data acquisition (SCADA) system [4]. These servers typically use commodity operating systems, enabling the use of standard digital forensic tools. However, field devices in an industrial control system such as programmable logic controllers (PLCs) and remote terminal units (RTUs) typically rely on proprietary hardware and embedded operating systems, and, therefore, require specialized digital forensic tools and techniques. Unfortunately, these tools and techniques are very limited in their functionality or simply do not exist.

Programmable logic controllers, which interact with and manage sensors and actuators, are important components of industrial control systems. As a result, they are attractive targets for attackers. A notable example is the Stuxnet malware that targeted Siemens programming logic controllers that operated Iran's uranium hexafluoride centrifuges [3]. The malware reprogrammed programmable logic controller code to cause malfunctions and damage while providing fabricated data to the operators in order to mask the attacks.

Unlike traditional digital forensics, no standard guidelines, procedures and tools are available for performing programmable logic controller forensics. A key challenge is the lack of system logging for forensic investigations. This chapter proposes a forensic methodology that captures the values of relevant memory addresses used by a programmable logic controller program in a log file. Machine learning techniques are applied to the logged data to identify anomalous or abnormal programmable logic controller operations. The methodology is applied to the popular Siemens Simatic S7-1212C programmable logic controller. Experiments with a simulated traffic light control system demonstrate the effectiveness and utility of the methodology in forensic investigations of incidents involving programmable logic controllers.

2. Programmable Logic Controllers

A programmable logic controller is a special microprocessor-based device that uses programmable memory to store instructions and implement functions such as logic, sequencing, timing, counting and arithmetic in order to monitor and control equipment and processes [2]. Figure 1 shows a schematic diagram of a programmable logic controller.

The programming of controllers is an important task when designing and implementing control applications. Each programmable logic controller has to be loaded with a program that controls the status of

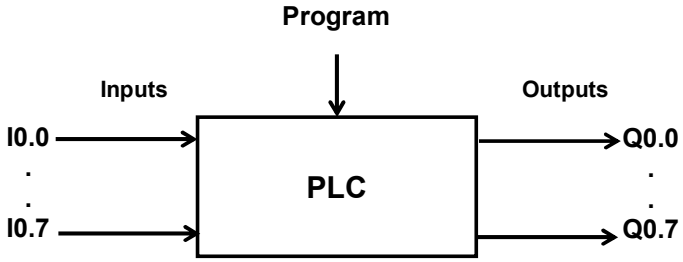

Figure 1. Programmable logic controller.

outputs based on the status of inputs. A programmable logic controller identifies each input or output according to its memory address. In the case of Siemens programmable logic controllers, the addresses of inputs and outputs are expressed in terms of their byte and bit numbers. For example, I0.1 is an input at bit 1 in byte 0 and Q0.7 is an output at bit 7 in byte 0.

A programmable logic controller exhibits anomalous operations in the following situations: (i) hardware failure; (ii) incompatible firmware version; (iii) control program bugs created by an authorized programmer or attacker; (iv) stop and start attacks [1]; and (v) memory read and write attacks [1].

The first step in detecting these anomalous operations is to capture the values of the inputs and outputs used by the control program in a log file. Machine learning techniques are subsequently applied to the logged data in order to detect anomalous operations.

3. Forensic Challenges

Digital forensic guidelines, procedures and tools have been developed for traditional information technology infrastructures and environments. A digital forensic process includes identification, collection, analysis and reporting. However, performing digital forensic techniques on programmable logic controllers is subject to several challenges [11]:

- **Lack of Documentation:** Low-level documentation of proprietary hardware, firmware and applications is usually not available for programmable logic controllers.

- **Lack of Domain-Specific Knowledge and Experience:** Digital forensics of programmable logic controllers is hindered by the lack of expertise and experience on the part of investigators.

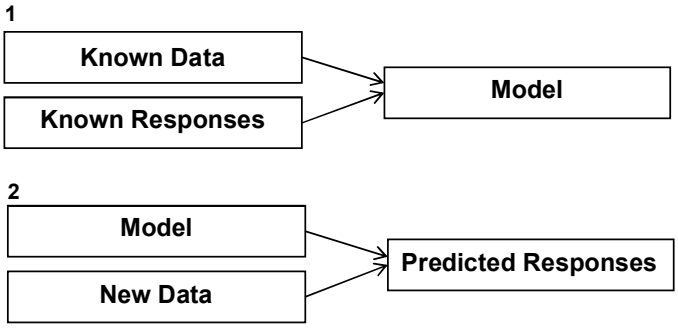

Figure 2. Supervised learning.

- **Lack of Logging Mechanisms:** Programmable logic controllers typically do not log data for forensic purposes.

- **Lack of Forensic Tools:** Limited, if any, forensic tools are available for conducting investigations of incidents involving programmable logic controllers.

- **Availability/Always On:** The availability of programmable logic controllers in an industrial control environment is a priority. It is extremely difficult to shut down a control system and physical process in order to conduct a forensic investigation.

4. Machine Learning

Machine learning is a data analysis method that automates model building. By leveraging algorithms that iteratively learn from data, machine learning enables computer systems to find hidden insights without being explicitly programmed [12]. Machine learning techniques have been applied in a number of areas, including pattern and image recognition, email spam filtering and network intrusion detection.

Supervised learning is the most common machine learning approach and several algorithms such as decision trees, support vector machines and artificial neural networks have been developed to implement supervised learning. In general, a supervised learning algorithm takes a known set of input data and known responses to the data, and creates a model that effectively predicts the responses to new input data [7].

Figure 2 shows a schematic diagram of the supervised learning approach. The experiments conducted in this research leveraged decision tree (DT) and support vector machine (SVM) learning algorithms to

analyze log file data in order to detect anomalous programmable logic controller operations.

5. Related Work

The Stuxnet attack [3] significantly increased research efforts related to industrial control system security, including intrusion detection and anomaly detection. However, very little research has specifically focused on applying machine learning techniques to detect intrusions and anomalous operations. An example is the work of Morris et al. [9], which trained a classifier on log data captured from a laboratory-scale gas pipeline and used it to detect 35 cyber attacks.

Another example is the research of Mantere et al. [6], which leveraged network traffic features to detect anomalies in specific industrial control systems. Mantere and colleagues used machine learning to decrease the amount of manual customization required to deploy network security monitors and intrusion detection systems in industrial control systems.

The research presented in this chapter differs from related work in that it concentrates on monitoring and capturing data directly from programmable logic controllers to support forensic investigations of intrusions and anomalous operations.

6. Experimental Setup and Methodology

This section describes the experimental setup and the methodology for identifying anomalous programmable logic controller operations.

6.1 Experimental Setup

The experiments used a Siemens S7-1212C programmable logic controller loaded with the TLIGHT traffic light control program. TLIGHT is a sample program provided with the Siemens SIMATIC S7-300 Programmable Controller Quick Start User Guide [16]. As shown in Figure 3, the TLIGHT program controls vehicles and pedestrian traffic at an intersection.

In order to simulate the hardware configuration of a traffic light control system, the programmable logic controller inputs I0.0 and I0.1 were connected to switches and the outputs Q0.0, Q0.1, Q0.5, Q0.6 and Q0.7 were connected to traffic lights. Figure 4 shows the input/output connections of the Siemens S7 1212C programmable logic controller. The Ethernet port of the programmable logic controller was used to establish a network connection for communicating with a peripheral device such as a laptop for programming the system.

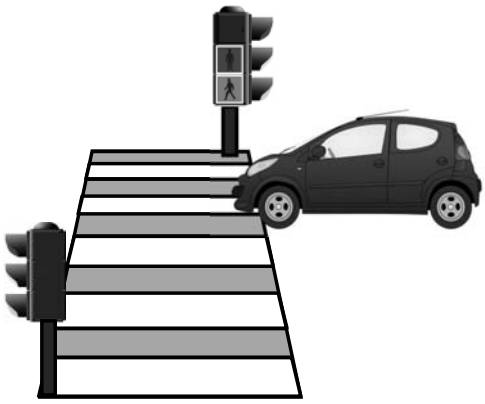

Figure 3. TLIGHT control system.

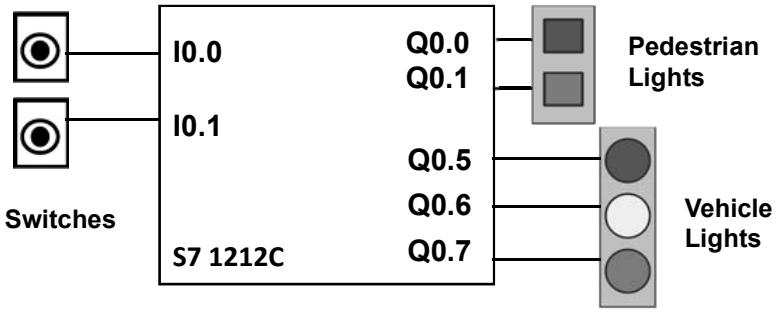

Figure 4. Siemens S7 1212C showing the input/output connections.

A program using the `libnodave` open source library [5] was used to log the values of relevant memory addresses used by the TLIGHT program. In particular, the program monitored the programmable logic controller memory addresses over the network and recorded the values along with their timestamps. To simplify the supervised learning process, all the non-binary values of memory addresses (e.g., timers) were converted to binary values.

6.2 Classifying Anomalous Operations

A machine learning technique typically splits the available data into two parts: (i) training set for learning the properties of the data; and (ii) testing set for evaluating the learned properties of the data. The accuracy of response prediction was evaluated using the testing set [14].

Table 1. TLIGHT control program instructions.

Instruction	Address	Description
Outputs	Q 0.0	Red for pedestrians
	Q 0.1	Green for pedestrians
	Q 0.5	Red for vehicles
	Q 0.6	Yellow for vehicles
	Q 0.7	Green for vehicles
Inputs	I 0.0	Switch on the right-hand side of the street
	I 0.1	Switch on the left-hand side of the street
Memory Bit	M 0.0	Memory bit for switching the signal after a green request from a pedestrian
Timers (On-Delay)	T 2	Red for pedestrians
	T 3	Green for pedestrians
	T 4	Red for vehicles
	T 5	Yellow for vehicles
	T 6	Green for vehicles

In order to implement supervised leaning, it was first necessary to understand the TLIGHT program logic. The TLIGHT program comprises instructions that involve inputs, outputs, memory bits and timers. Table 1 provides details about the TLIGHT instructions. Figure 5 shows the input and output signal states during the TLIGHT sample program sequence [16].

The supervised learning approach involved the following steps:

- **Step 1: Training Set Creation:** A training example corresponds to a pair of input objects (values of relevant addresses) and known responses (normal/anomalous operations of the traffic lights). Normal and anomalous operations of TLIGHT were determined according to the values at the relevant addresses (timers and outputs). The seven normal operations of TLIGHT presented in Figure 5 are based on the values of the timers and outputs at various time intervals. Table 2 shows the input objects and known responses for the seven normal operations of TLIGHT. These were transformed to the input data matrix format for use in supervised learning.

  The training set was generated by running the traffic light control system and logging system to capture the values of relevant addresses (inputs, outputs and timers) used by TLIGHT. Anomalous operations were created by altering some values in address locations using Snap7, an open-source, 32/64 bit, multi-platform

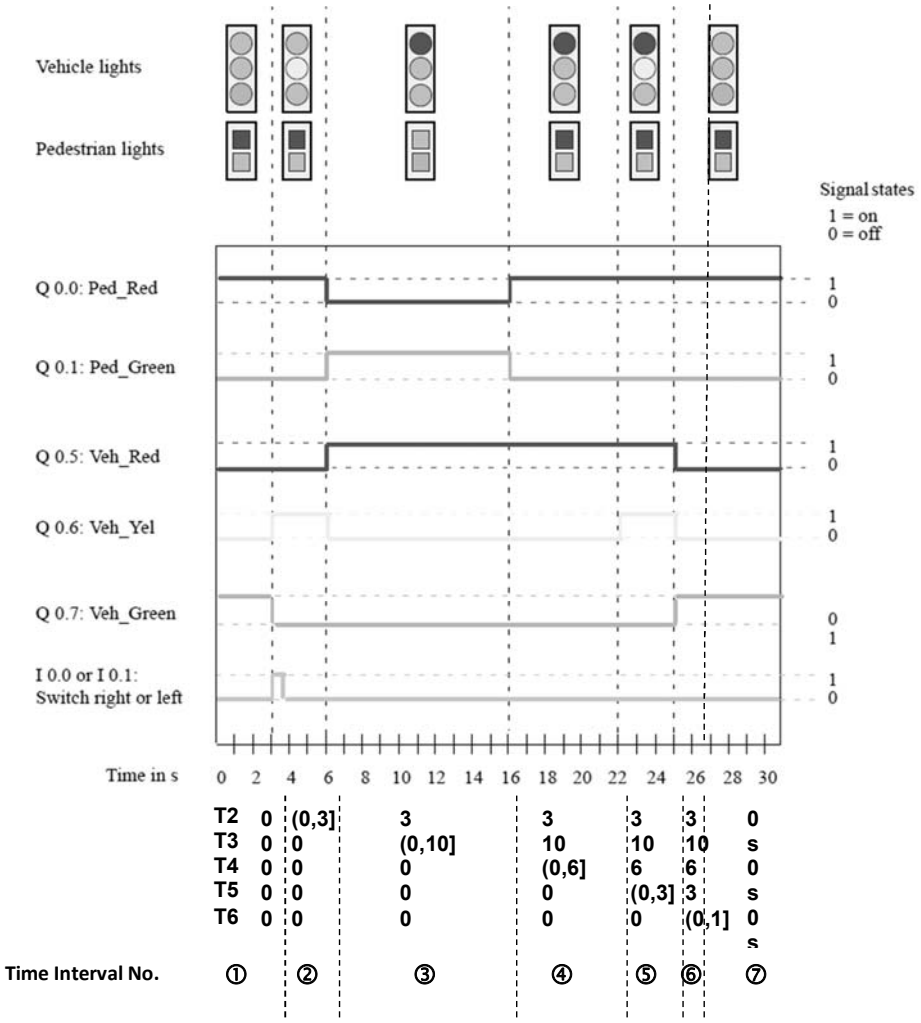

Figure 5. Input and output signal states during the TLIGHT program sequence.

Ethernet communications suite for interfacing with Siemens S7 programmable logic controllers [10]. Thus, the generated log file contained normal and anomalous traffic light operations.

- **Step 2: Supervised Learning Algorithm Selection:** Decision tree (DT) and support vector machine (SVM) supervised learning algorithms were employed in the experiments. A decision tree algorithm was selected to classify anomalous operations of TLIGHT for several reasons [8]. First, the target function has discrete output

Table 2. Seven normal TLIGHT operations.

Time Interval	Input Objects										Known Response
	Yes=1; No=0					On=1; Off=0					Yes=1; No=0
	T2=3?	T3=10?	T4=6?	T5=3?	T6=1?	Q0.0	Q0.1	Q0.5	Q0.6	Q0.7	Normal?
1	0	0	0	0	0	1	0	0	0	1	1
2	0	0	0	0	0	1	0	0	1	0	1
3	1	0	0	0	0	0	1	1	0	0	1
4	1	1	0	0	0	1	0	1	0	0	1
5	1	1	1	0	0	1	0	1	1	0	1
6	1	1	1	1	0	1	0	0	0	1	1
7	0	0	0	0	1	1	0	0	0	1	1

values (TLIGHT operations). Additionally, a decision tree algorithm is fairly robust at handling training data errors, including mislabeled attribute values. Indeed, somewhat noisy data (e.g., due to errors in assigning response values) do not pose much of a problem for a decision tree algorithm. A decision tree algorithm can also handle data with missing attribute values (e.g., missing values at memory addresses during data capture).

In addition to a decision tree algorithm, a support vector machine supervised learning algorithm was used. This was done to determine if any obvious differences in accuracy and performance occur when a different machine learning algorithm is used. Furthermore, the support vector machine algorithm performs well even with small training datasets

- **Step 3: Supervised Learning Algorithm Application:** The captured data was assigned response values corresponding to normal or anomalous operations and was subsequently transformed to a matrix format for input to the supervised learning algorithms (Table 2). Decision tree and support vector machine classifiers provided by scikit-learn [15] were used for model training. Table 3 lists the settings of the classifiers used in the experiments. The classifiers were implemented using default values of the input parameters. No k-fold cross validation was applied to the training samples. Finally, the accuracy of response prediction was evaluated based on the testing data.

Table 3. DT and SVM classifier settings in scikit-learn.

	DT	**SVM**
Class	*tree.DecisionTreeClassifier*	*svm.SVC*
Training Samples	Transaction records in data log files with assigned known responses	
• **Input Objects**	Memory addresses used in TLIGHT	
• **Known Responses**	Operational status of TLIGHT (normal/anomalous)	
Parameter Settings	Default settings	

Table 4. Classification accuracy.

	Dataset 1		**Dataset 2**	
Training Records	560		1,600	
Testing Records	2,240		6,400	
Learning Algorithm	**DT**	**SVM**	**DT**	**SVM**
Accuracy	99.91%	99.91%	99.94%	99.60%

7. Experimental Results and Discussion

In order to evaluate the accuracy of the learned models, two datasets were prepared for the decision tree and support vector machine learning algorithms. The first set (Dataset 1) contained 2,800 records, 560 for training and 2,240 for testing. The second set (Dataset 2) contained 8,000 records, 1,600 for training and 6,400 for testing. Table 4 shows the classification accuracy. The accuracy rates with Dataset 1 for the decision tree and support vector machine learning algorithms were 99.91% while the accuracy rates with Dataset 2 for the decision tree and support vector machine learning algorithms were 99.94% and 99.60%, respectively.

In the experiments, transaction records in the log file corresponding to anomalous operations were identified by machine learning. Because the values at the relevant addresses used by TLIGHT were recorded along with timestamps in the log file, it was possible to trace which values had been altered and when they were altered, and subsequently identify the anomalous TLIGHT operations. However, a log file alone may be insufficient in a forensic investigation because it does not contain information about what (e.g., IP address) induced an anomalous operation and how it was induced. For this reason, a forensic investigator should use a network packet analyzer such as Wireshark to capture

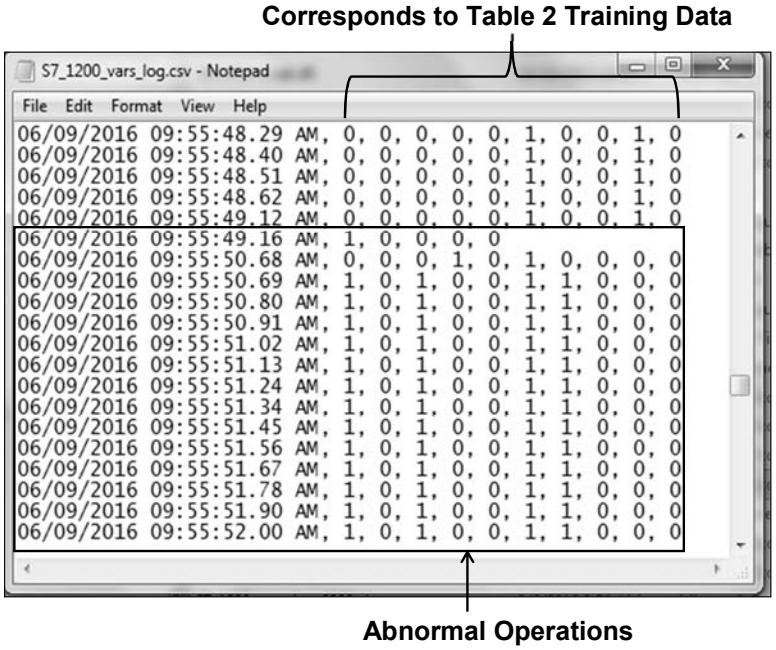

Figure 6. Log file with anomalous operations.

packets while employing the logging system to record the relevant addresses used by TLIGHT. Wireshark supports the PROFINET industrial data communications standard in order to record and analyze Ethernet message frames. It can be used to dissect the ISO-on-TCP packets in Siemens S7 programmable logic controller communications after adding the Wireshark S7 dissector plugin. Note that S7 is a function-oriented or command-oriented protocol in that each transmission contains a command or a reply.

After collecting and analyzing the log file and network packet data, a forensic investigator can discover where the compromise originated, how it was carried out and, possibly, who was responsible for the incident. For example, the machine learning algorithms (decision tree and support vector machine) identified that anomalous operations started on 09 June 2016 at 09:55.49.16 AM (Figure 6) and on 29 August 2016 at 09:27:38.96 PM (Figure 7). This step saves a forensic investigator considerable time in identifying anomalous transactions. Based on the timestamps in the log file, an investigator can focus on checking the actions performed on the system (e.g., firmware or user control program updates by authorized operators and alterations performed by unauthorized entities).

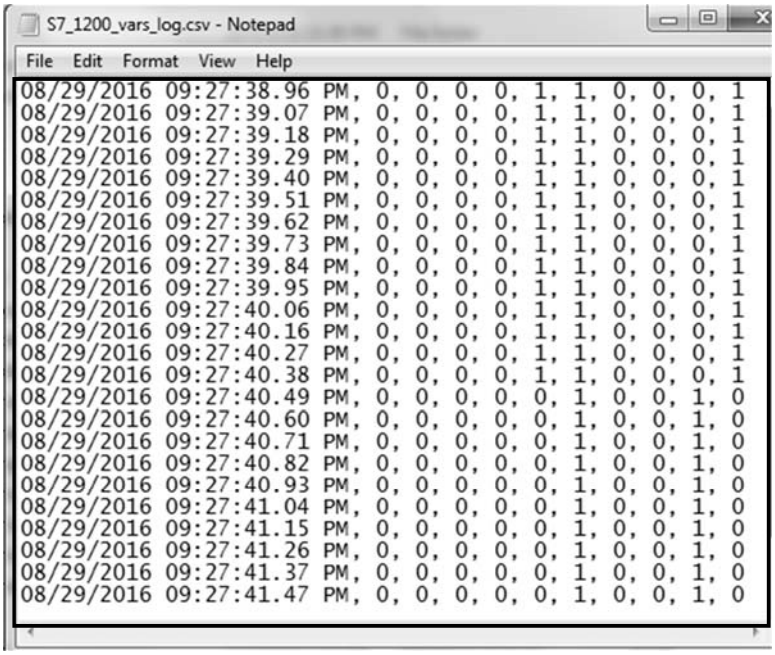

Figure 7. Anomalous operations.

In the experiment, the anomalous operations that started on 09 June 2016 at 09:55.49.16 AM were the result of a programmable logic controller self-test and the anomalous operations that started on 29 August 2016 at 09:27:38.96 PM were due to a simulated attack (Snap7) [10]. These examples demonstrate that machine learning can help a forensic investigator filter unnecessary log data and narrow the scope of the forensic investigation.

The methodology presented in this chapter can be extended to other brands of programmable logic controllers and other control programs. However, it is not possible to create a single logging system for all programmable logic controller applications because different applications require different control programs. Therefore, each programmable logic controller application should have its own logging system. In order to create a logging system, it is necessary to understand the design of the control program and identify the programmable logic controller memory addresses that must be monitored and analyzed.

In order to simplify machine learning, the experiments did not consider the time sequences of normal programmable logic controller operations. Therefore, the accuracy of the results may vary. Note also that

supervised learning is by no means the only way to identify anomalous programmable logic controller operations. In fact, the work described in this chapter serves as an initial approach to determine whether or not supervised learning is feasible for programmable logic controller forensics. Indeed, the experimental results demonstrate that supervised learning can help predict anomalous operations with uncertain inputs and responses, even in the case of complicated user control programs.

8. Conclusions

A log containing the values at the relevant memory addresses used by a programmable logic controller program along with their timestamps can be very valuable in a forensic investigation of an industrial control system incident. In particular, machine learning techniques can applied to the logged data to identify anomalous programmable logic controller operations. The application of the methodology to a simulated traffic light control system demonstrates its effectiveness in a forensic investigation involving a programmable logic controller. Since different programmable logic controller applications require different control programs, each application should have its own logging system. In order to create the logging system, it is necessary to understand the design of the control program and identify the programmable logic controller memory addresses that must be monitored and analyzed. However, a log file alone may be insufficient in a forensic investigation because it may not contain information about what induced an anomalous operation and how it was induced. Therefore, it is recommended to augment the log file with data from a network packet analyzer such as Wireshark.

This research is an initial step in developing forensic capabilities for programmable logic controllers. Future research will attempt to apply and refine machine learning techniques to various industrial control system applications to support forensic investigations of intrusions and anomalous behavior in these vital systems that permeate the critical infrastructure.

References

[1] D. Beresford, Exploiting Siemens Simatic S7 PLCs, presented at *Black Hat USA*, 2011.

[2] W. Bolton, *Programmable Logic Controllers*, Newnes, Burlington, Massachusetts, 2009.

[3] N. Falliere, L. O'Murchu and E. Chien, W32.Stuxnet Dossier, Symantec, Mountain View, California, 2011.

[4] L. Folkerth, Forensic Analysis of Industrial Control Systems, InfoSec Reading Room, SANS Institute, Bethesda, Maryland, 2015.

[5] T. Hergenhahn, `libnodave` (`sourceforge.net/projects/libnodave`), 2014.

[6] M. Mantere, M. Sailio and S. Noponen, Network traffic features for anomaly detection in a specific industrial control system network, *Future Internet*, vol. 5(4), pp. 460–473, 2013.

[7] MathWorks, Supervised Learning Workflow and Algorithms, Natick, Massachusetts (`www.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html?requestedDomain=www.mathworks.com`), 2017.

[8] T. Mitchell, *Machine Learning*, WCB/McGraw-Hill, Boston, Massachusetts, 1997.

[9] T. Morris, Z. Thornton and I. Turnipseed, Industrial control system simulation and data logging for intrusion detection system research, *Proceedings of the Seventh Annual Southeastern Cyber Security Summit*, 2015.

[10] D. Nardella, Step 7 Open Source Ethernet Communication Suite, Bari, Italy (`snap7.sourceforge.net`), 2016.

[11] H. Patzlaff, D 7.1 Preliminary Report on Forensic Analysis for Industrial Systems, CRISALIS Consortium, Symantec, Sophia Antipolis, France, 2013.

[12] SAS Institute, Machine Learning: What it is and Why it Matters, Milan, Italy (`www.sas.com/it_it/insights/analytics/machine-learning.html`), 2016.

[13] S. Sayad, *An Introduction to Data Mining*, University of Toronto, Toronto, Canada, 2011.

[14] scikit-learn Project, An Introduction to Machine Learning with scikit-learn (`scikit-learn.org/stable/tutorial/basic/tutorial.html`), 2016.

[15] scikit-learn Project, Supervised Learning (`scikit-learn.org/stable/supervised_learning.html#`), 2016.

[16] Siemens, SIMATIC S7-300 Programmable Controller Quick Start, Primer, Preface, C79000-G7076-C500-01, Nuremberg, Germany, 1996.