Chapter 6

DETECTING ANOMALOUS PROGRAMMABLE LOGIC CONTROLLER EVENTS USING PROCESS MINING

Ken Yau, Kam-Pui Chow and Siu-Ming Yiu

Abstract Programmable logic controllers that monitor and control industrial processes are attractive targets for cyber attackers. Although techniques and tools have been developed for detecting anomalous programmable logic controller behavior, they rely heavily on knowledge of the complex programmable logic controller control programs that perform process monitoring and control. To address this limitation, this chapter describes an automated process mining methodology that relies on event logs comprising programmable logic controller inputs and outputs. The methodology discovers a process model of normal programmable logic controller behavior, which is used to detect anomalous behavior and support forensic investigations. Experiments involving a popular Siemens SIMATIC S7-1212C programmable logic controller and a simulated traffic light system demonstrate the utility and effectiveness of the methodology.

Keywords: Programmable logic controller, process mining, anomaly detection

1. Introduction

Programmable logic controllers, which are widely used to monitor and control industrial processes, are computer systems that are designed to operate reliably under harsh industrial conditions such as extreme temperatures, vigorous vibrations, humidity and/or dusty conditions [21]. Historically, programmable logic controllers were proprietary systems that operated in isolation with no external network connections. However, modern programmable logic controllers use common embedded system platforms and commercial off-the-shelf software [3]. Addition-

Published by Springer Nature Switzerland AG 2022
J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XV, IFIP AICT 636, pp. 119–133, 2022.
https://doi.org/10.1007/978-3-030-93511-5_6

ally, they are often networked using TCP/IP and wireless protocols and may connect to vendor networks, corporate networks and even the Internet [12]. The vital roles played by modern programmable logic controllers in critical infrastructure assets make them attractive targets for attackers. Their use of commodity hardware and software coupled with their network connectivity significantly increase their exposure to cyber threats. Securing programmable logic controllers from cyber attacks is a priority [13].

Programmable logic controllers exhibit anomalous behavior in attack scenarios as well as during malfunctions and error situations. Detecting anomalous behavior is an important first step in securing programmable logic controllers and mitigating the negative impacts on the industrial processes they operate. However, anomaly detection is a challenging problem because it demands detailed knowledge of the complex programmable logic controller control programs that monitor and control industrial processes. Additionally, due to their real-time operation of industrial processes, it is difficult to stop programmable logic controllers to investigate anomalies [19].

To address these limitations, this chapter presents an automated process mining methodology that relies on event logs comprising programmable logic controller inputs and outputs. The methodology discovers a process model of normal programmable logic controller behavior, which is used to detect anomalous behavior and support forensic investigations. Experiments involving a popular Siemens SIMATIC S7-1212C programmable logic controller and a simulated traffic light system demonstrate the utility and effectiveness of the automated process mining methodology.

2. Related Work

Process mining techniques enable analysts to extract insights about process operations from collections of event records or logs [1]. Process mining has traditionally been used to analyze business processes. However, process mining is increasingly being applied to anomaly detection.

Van der Aalst and de Medeiros [14] advocate the use of process mining to analyze audit trails for security violations. They demonstrate how process mining can support security efforts ranging from low-level intrusion detection to high-level fraud prevention. They also show how process mining can be used to identify anomalous behavior in an online shopping website.

Myers et al. [7] have investigated the application of process mining discovery algorithms on control device log data to detect cyber attacks

on industrial control systems. Their research shows that inductive miner process discovery (without noise alteration) is most effective at discovering process models for detecting cyber attacks on industrial control systems.

Laftchiev et al. [6] have proposed an anomaly detection approach for discrete manufacturing systems. They focus on detecting incorrect event execution sequences in discrete manufacturing systems using output data from programmable logic controllers. They employ process mining to develop models of normal behavior and identify anomalous behavior as event sequences that deviate from the modeled normal behavior.

The methodology presented in this chapter differs from related work in that it attempts to detect anomalous behavior by monitoring and analyzing programmable logic controller inputs and outputs, eliminating the need to capture and engage detailed knowledge about the system being controlled. Additionally, the programmable logic controller event logs used for process mining can support post-incident forensic investigations.

The Stuxnet attacks [4] stimulated research efforts on discovering vulnerabilities in programmable logic controllers and addressing potential threats. Wu and Nurse [16] have shown that attacks on programmable logic controllers can be detected by monitoring the values of memory addresses in control programs. They identify the memory addresses used by control program code, and monitor and log the memory values to capture normal programmable logic controller behavior. The normal behavior serves as a reference to identify anomalous programmable logic controller behavior.

Yau et al. [17, 18, 20] have focused on forensic approaches for programmable logic controllers. One approach detects and records anomalous programmable logic controller events based on changes to the control program logic. Another approach captures the values of relevant memory addresses of a running control program and applies machine learning techniques to the captured data to create a model for recognizing anomalous programmable logic controller behavior.

Wu and Nurse [16] and Yau et al. [16–18, 20] have shown that anomalous programmable logic controller behavior can be detected by checking whether or not the control program logic has been changed. However, the methods require the collection and analysis of numerous memory values used by control programs. In contrast, the methodology described in this chapter detects anomalous programmable logic controller behavior simply by monitoring programmable logic controller input and output values.

3. Process Mining

Process mining provides theoretical and practical foundations for discovering process models from various types of event data [10]. Many successful applications of process mining have been developed for business process management, but the adoption of process mining in other domains is limited.

Process mining seeks to discover, monitor and improve real processes by extracting knowledge from event logs. The techniques include process discovery, conformance checking and process enhancement [7]. This research employs process discovery to learn a Petri net model of a programmable logic controller as it controls a traffic light system. The Petri net model obtained by the process mining of event logs captures normal programmable logic controller behavior. Programmable logic controller behavior that does not match the expected behavior modeled by the Petri net is deemed to be anomalous.

Several types of process models can be constructed using process mining. The proposed methodology employs the Petri net formalism, which is widely used in computer science and systems engineering. Petri nets combine a mathematical theory with graphical representations of dynamic system behavior. The theory enables the precise modeling and analysis of system behavior; the graphical representation provides visualizations of modeled system states [15].

Definition. A Petri net is a five-tuple $PN = (P, T, I, O, M_0)$ where P is a finite set of places; T is a finite set of transitions where $P \cup T \neq \phi$ and $P \cap T = \phi$; $I : P \times T \rightarrow \mathbb{N}$ is an input function that defines directed arcs from places to transitions where $\mathbb{N}$ is a set of non-negative integers; $O : T \times P \rightarrow \mathbb{N}$ is an output function that defines directed arcs from transitions to places; and $M_0 : P \rightarrow \mathbb{N}$ is the initial Petri net marking.

A Petri net marking is an assignment of tokens to places. Tokens reside in places. The numbers and positions of tokens may change during Petri net execution. In fact, tokens are used to define Petri net execution.

The basic idea in Petri net modeling is to describe state changes in a system using transitions that symbolize actions. A Petri net contains places (circles) and transitions (boxes) that may be connected by directed arcs (Figure 1). Places symbolize states, conditions or resources that must be met or be available before an action can be carried out. Places may contain tokens that move to other places by executing (firing) actions. A token in a place means that the corresponding condition is fulfilled or that a resource is available. In the transition diagram in

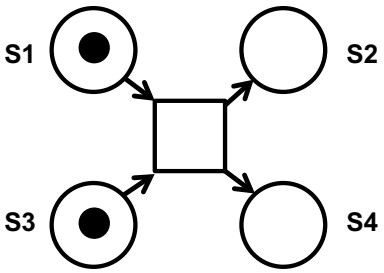

Figure 1. Petri net transition diagram.

Figure 1, the transition may fire when there are tokens in places S1 and S3. Firing removes the tokens in S1 and S3 and puts new tokens in S2 and S4.

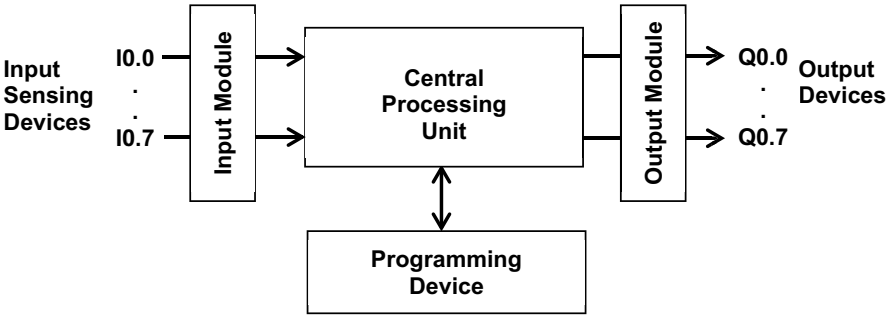

Figure 2. Programmable logic controller.

4. Overview

A programmable logic controller uses programmable memory to store instructions and implement functions such as logic, sequencing, timing, counting and arithmetic for its monitoring and control tasks [2]. Figure 2 shows a schematic diagram of a programmable logic controller.

A programmable logic controller executes a control program that changes the status of programmable logic controller outputs based on the status of its inputs. Each input and output is identified by its address. In the case of a Siemens SIMATIC S7-300 programming logic controller, the addresses of the inputs and outputs are expressed in terms of byte and bit numbers. For example, I0.1 is an input at bit 1 in byte 0 and Q0.2 is an output at bit 2 in byte 0.

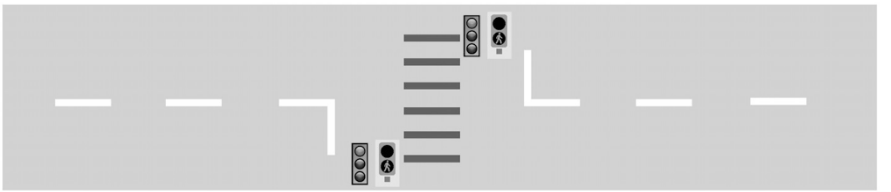

Figure 3. TLIGHT traffic light system.

The experiments employed a Siemens S7-1212C controller loaded with a TLIGHT traffic light simulation program to demonstrate the process-mining-based anomaly detection methodology. TLIGHT is a sample program that is provided with the Siemens SIMATIC S7-300 Programmable Controller Quick Start User Guide [11]. Figure 3 shows the TLIGHT traffic signal system controlling vehicle and pedestrian traffic at an intersection.

5. Proposed Methodology

This section describes the methodology for detecting anomalous programmable logic controller behavior using process mining. The methodology involves the following steps:

- A simulated traffic light system is set up using a Siemens programmable logic controller. The system is isolated from other networks.
- Programmable logic controller activities (inputs and outputs) are recorded every second to create an activity log that represents normal programmable logic controller behavior.
- Process mining is used to create a Petri net model from the activity log.
- An invalid state transition detector is created to identify anomalous behavior based on the Petri net model.
- The traffic light system is connected to a network and anomalous traffic light system behavior is induced.
- The accuracy of the invalid state transition detector is evaluated based on its ability to identify anomalous programmable logic controller behavior.

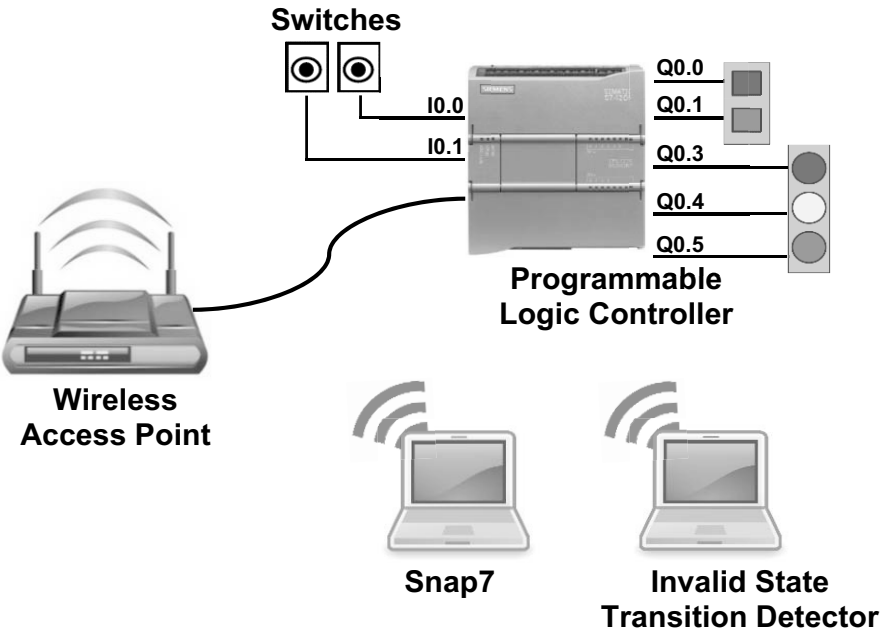

Figure 4. Experimental setup.

5.1 Traffic Light System

Figure 4 shows the experimental setup. A Siemens S7-1212C programmable logic controller running the TLIGHT traffic light program was employed. The simulated traffic light control system was created by establishing the input and output associations. Specifically, the programmable logic controller inputs I0.0 and I0.1 were connected to switches associated with the two green light request buttons for pedestrians. The programmable logic controller outputs were connected as follows: Q0.0 to the red light for pedestrians, Q0.1 to the green light for pedestrians, Q0.3 to the red light for vehicles, Q0.4 to the yellow light for vehicles and Q0.5 to the green light for vehicles.

The Ethernet port of the programmable logic controller was used to establish a network connection for communicating with two peripheral devices, Snap7 [8] and the invalid state transition detector. Snap7 is an open-source, 32/64 bit, multi-platform Ethernet communications suite for interfacing with Siemens S7 programmable logic controllers. In the experiments, Snap7 was used to induce anomalous traffic light behavior by altering certain memory values in the control program of the programmable logic controller. The invalid state transition detec-

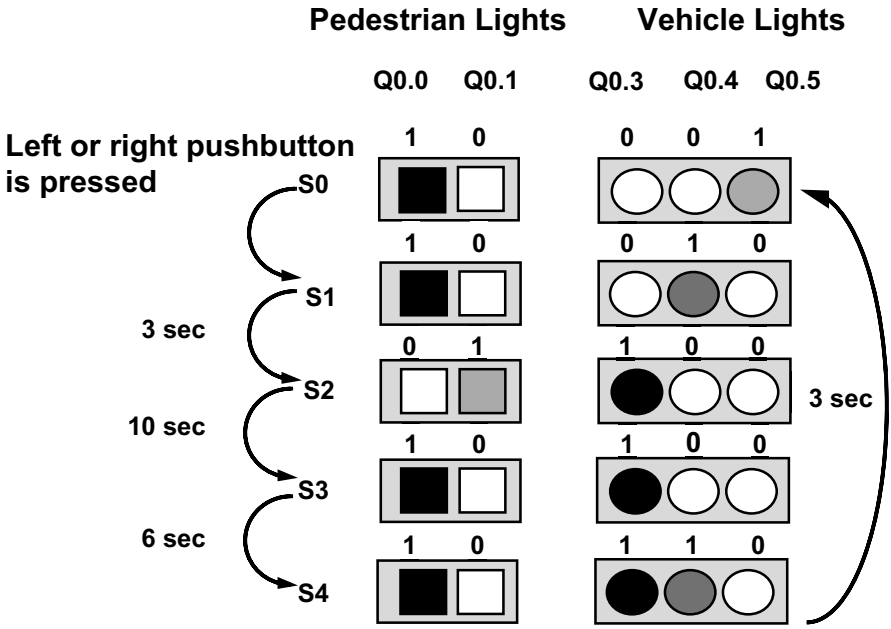

Figure 5. Traffic light state sequences.

tor was employed to detect anomalous behavior and capture evidence for forensic purposes. The detector was developed using `libnodave`, an open-source communications library for SIMATIC S7 programmable logic controllers [5]

The TLIGHT traffic light program controls vehicle and pedestrian traffic at an intersection. Figure 5 shows the five traffic light states (S0, S1, S2, S3 and S4) and their sequences (with timings).

5.2 Programmable Logic Controller Behavior

The `libnodave` open-source library was also used to develop a logging program. The values used by TLIGHT were stored at input/output addresses I0.0, I0.1, Q0.0, Q0.1, Q0.3, Q0.4 and Q0.5. The logging program monitored these programmable logic controller addresses over the network and recorded the values along with their timestamps every second.

The programmable logic controller inputs and outputs were transformed to activities that manifested normal programmable logic controller behavior. The input activities were I-00 (I0.0=0 and I0.1=0), I-01, I-10 and I-11. The output activities corresponded to the traffic

Case number		Event ID	Activity	Input	Output	Date/Time
20	S0	103	Q-10001	00	10001	7/6/2019 7:24:10 PM
21		104	I-10	10	10010	7/6/2019 7:24:11 PM
21		105	Q-10010	10	10010	7/6/2019 7:24:11 PM
21	S1	106	Q-10010	00	10010	7/6/2019 7:24:12 PM
21		107	Q-10010	00	10010	7/6/2019 7:24:13 PM
21		108	Q-01100	00	01100	7/6/2019 7:24:14 PM
21		109	Q-01100	00	01100	7/6/2019 7:24:15 PM
21		110	Q-01100	00	01100	7/6/2019 7:24:16 PM
21		111	Q-01100	00	01100	7/6/2019 7:24:17 PM
21	S2	112	Q-01100	00	01100	7/6/2019 7:24:18 PM
21		113	Q-01100	00	01100	7/6/2019 7:24:19 PM
21		114	Q-01100	00	01100	7/6/2019 7:24:20 PM
21		115	Q-01100	00	01100	7/6/2019 7:24:21 PM
21		116	Q-01100	00	01100	7/6/2019 7:24:22 PM
21		117	Q-01100	00	01100	7/6/2019 7:24:23 PM
21		118	Q-10100	00	10100	7/6/2019 7:24:24 PM
21		119	Q-10100	00	10100	7/6/2019 7:24:25 PM
21	S3	120	Q-10100	00	10100	7/6/2019 7:24:26 PM
21		121	Q-10100	00	10100	7/6/2019 7:24:27 PM
21		122	Q-10100	00	10100	7/6/2019 7:24:28 PM
21		123	Q-10100	00	10100	7/6/2019 7:24:29 PM
21		124	Q-10110	00	10110	7/6/2019 7:24:30 PM
21	S4	125	Q-10110	00	10110	7/6/2019 7:24:31 PM
21		126	Q-10110	00	10110	7/6/2019 7:24:32 PM
22		127	I-00	00	10001	7/6/2019 7:24:33 PM

Figure 6. Programmable logic controller activity log.

light states: Q-10001 (S0), Q-10010 (S1), Q-01100 (S2), Q-10100 (S3) and Q-10110 (S4). Figure 6 shows a portion of the programmable logic controller activity log for normal traffic light operations.

Process mining requires data pertaining to a process, cases, events and attributes. A process comprises cases. A case comprises events (activities) where each event is related to precisely one case. Events within a case are ordered. An event may have attributes such as resource and timestamp.

The experimental data comprised one process, 7,832 cases and 34,956 events (activities). A case corresponded to a complete execution of the TLIGHT traffic light system.

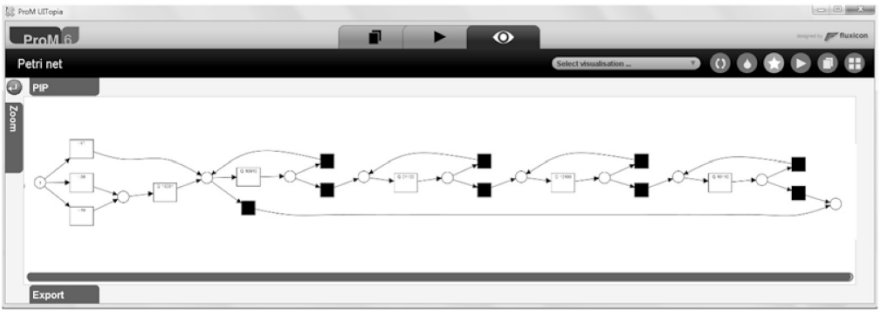

Figure 7. Petri net model.

5.3 Petri Net Model

The ProM 6.6 process mining framework [9] was used to discover a process model. This extensible framework supports a variety of process mining techniques in the form of plug-ins. The function mine Petri net with the inductive miner was applied to the dataset containing traffic light activities over a four-hour period. Figure 7 shows the discovered Petri net model.

To improve the Petri net model, the programmable logic controller activity log was aggregated to obtain the duration of each state and this data was added to the process model. Next, the Petri net model was transformed to a finite state machine. A finite state machine is very similar to a Petri net. In the case of the traffic light system, the transformation rendered the state transitions more clear and understandable. Figure 8 shows the final finite state model that represents normal TLIGHT operations (i.e., programmable logic controller behavior).

5.4 Invalid State Transition Detector

The invalid state transition detector, which was also developed using `libnodave`, identifies anomalous programmable logic controller behavior. Specifically, it used the finite state machine shown in Figure 8 to detect two types of anomalous behavior:

- Invalid states, i.e., states other than S0, S1, S2, S3 and S4.
- Invalid time intervals between two consecutive states.

Instances of anomalous behavior raised alerts and the related evidence was recorded for a forensic investigation.

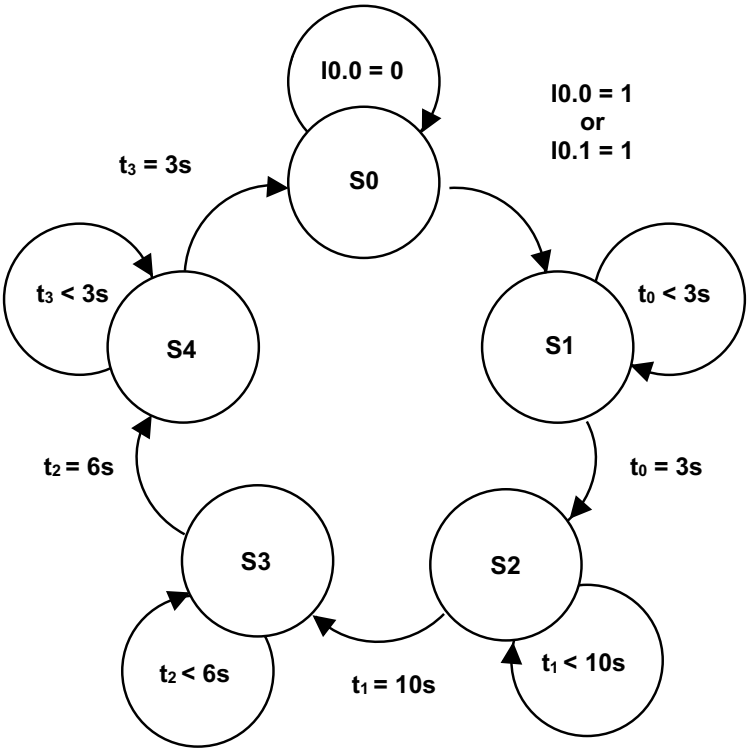

Figure 8. Finite state machine.

5.5 Anomalous Traffic Light Operations

Anomalous traffic light operations were induced using the Snap7 Ethernet communications suite. Specifically, Snap7 induced anomalous traffic light operations by flipping some memory values in the programmable logic controller control program between 0 and 1 intermittently.

5.6 Anomalous Behavior Detection

The invalid state transition detector monitored the programmable logic controller inputs and outputs as the S7 suite intermittently induced anom- alous traffic light operations. The detector created a log file of timestamped anomalous behavior transaction records containing programmable logic controller input values, previous state, current state and time intervals of state transitions.

Figure 9 shows a traffic light activity log with invalid states that start at 4:48:21 PM on 7/7/2019. The state 10000 corresponds to a situation

Date/Time	Input	Prevous State	Current State	Remarks
7/7/2019 4:48:21 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:22 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:23 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:24 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:25 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:26 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:27 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:28 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:29 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:30 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:31 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:32 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:34 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:35 PM	00	10000	10000	State 10000 is not valid
7/7/2019 4:48:36 PM	00	10000	10000	State 10000 is not valid

Figure 9. Traffic light log with invalid states.

where only one pedestrian red light is on and all the other traffic lights are off. Clearly, this is anomalous.

Date/Time	Input	Prevous State	Current State	Remarks
7/14/2019 9:44:08 AM	10	01100	01100	S2
7/14/2019 9:44:09 AM	10	01100	01100	S2
7/14/2019 9:44:10 AM	10	01100	01100	S2
7/14/2019 9:44:11 AM	10	01100	01100	S2
7/14/2019 9:44:12 AM	10	01100	01100	S2
7/14/2019 9:44:13 AM	10	01100	10100	Error :invalid time for transition S2 -> S3
7/14/2019 9:44:14 AM	10	10100	10100	S3
7/14/2019 9:44:15 AM	10	10100	10100	S3
7/14/2019 9:44:16 AM	10	10100	10100	S3
7/14/2019 9:44:17 AM	10	10100	10100	S3
7/14/2019 9:44:18 AM	10	10100	10100	S3
7/14/2019 9:44:19 AM	10	10100	10110	Error :invalid time for transition S3 -> S4
7/14/2019 9:44:20 AM	10	10110	10110	S4
7/14/2019 9:44:21 AM	10	10110	10110	S4

Figure 10. Traffic light log with invalid time interval between consecutive states.

Figure 10 shows a traffic light activity log with an invalid time interval between consecutive states. The anomalous behavior was induced by Snap7. In the log file, an invalid time interval exists between state S2 and S3 at 9:44:13 AM on 7/14/2019. The programmable logic controller inputs, outputs and traffic light system state transitions recorded with timestamps in the log file constitute evidence of anomalies. How-

ever, the log file alone may not be adequate in a forensic investigation because it does not have information about how the anomalies were induced, including if they were caused by attacks or malfunctions. On the other hand, an invalid state transition detection log raises alerts and provides information that would initiate a forensic investigation. In a forensic investigation, it would also be necessary to obtain and analyze supplementary log files such as network logs and the system logs of the device used to program the programmable logic controller. The timestamps recorded by the invalid state transition detector would enable investigators to filter relevant data from the supplementary log files and narrow the scope of the forensic investigation.

The decision to focus on a Siemens SIMATIC S7 programmable logic controller was motivated by its widespread use and the fact that it was targeted by the Stuxnet malware [4]. However, the proposed methodology can be applied to other programmable logic controllers and other control applications because the solution approach is not tied to specific hardware, software and operating systems. Rather, it only considers programmable logic controller inputs and outputs.

6. Conclusions

The automated process mining methodology presented in this chapter produces a Petri net model from event logs comprising programmable logic controller inputs and outputs. The Petri net model, which expresses normal programmable logic controller behavior, serves as a reference to detect anomalous behavior. The event logs with timestamped input and output values are also useful in forensic investigations.

Experiments involving a popular Siemens SIMATIC S7-1212C programmable logic controller and a simulated traffic light system yielded high accuracy. All the anomalies were detected because the simulated traffic light system is simple and involves limited programmable logic controller inputs/outputs, states and state transitions. The accuracy would likely be lower for a physical system with large numbers of inputs, outputs, states and state transitions. Nevertheless, the methodology holds promise because it can detect anomalous behavior without relying on detailed system knowledge and a complicated control program. Additionally, the methodology differs from other approaches because it can detect anomalous programmable logic controller events simply by monitoring programmable logic controller inputs and outputs instead of monitoring numerous memory addresses.

This research is an initial step in developing protection and forensic capabilities for programmable logic controllers based on process mining

techniques. Future work will attempt to apply process mining techniques to various industrial control system applications to support anomalous behavior detection and forensic investigations in industrial control systems and civilian applications.

References

[1] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. Maria Maggi, A. Marrella, M. Mecella and A. Soo, Automated Discovery of Process Models from Event Logs: Review and Benchmark, *IEEE Transactions on Knowledge and Data Engineering*, vol. 31(4), pp. 686–705, 2019.

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

[3] European Union Agency for Network and Information Security, Critical Infrastructures and Services, Heraklion, Greece (`enisa.europa.eu/topics/critical-information-infrastructures-and-services`), 2017.

[4] N. Falliere, L. O'Murchu and E. Chien, W32.Stuxnet Dossier, Version 1.4, Symantec, Mountain View, California, 2011.

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

[6] E. Laftchiev, X. Sun, H. Dau and D. Nikovski, Anomaly detection in discrete manufacturing systems using event relationship tables, *Proceedings of the International Workshop on Principles of Diagnosis*, 2018.

[7] D. Myers, K. Radke, S. Suriadi and E. Foo, Process discovery for industrial control system cyber attack detection, in *ICT Systems Security and Privacy Protection*, S. De Capitani di Vimercati and F. Martinelli (Eds.), Springer, Cham, Switzerland, pp. 61–75, 2017.

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

[9] RapidProM Team, ProM Tools, Eindhoven University of Technology, Eindhoven, The Netherlands (promtools.org/doku.php), 2019.

[10] V. Rubin, A. Mitsyuk, I. Lomazova and W. van der Aalst, Process mining can be applied to software too! *Proceedings of the Eighth ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, article no. 57, 2014.

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

[12] T. Spyridopoulos, T. Tryfonas and J. May, Incident analysis and digital forensics in SCADA and industrial control systems, *Proceedings of the Eighth IET International System Safety Conference Incorporating the Cyber Security Conference*, 2013.

[13] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams and A. Hahn, Guide to Industrial Control Systems (ICS) Security, NIST Special Publication 800-82, Revision 2, National Institute of Standards and Technology, Gaithersburg, Maryland, 2015.

[14] W. van der Aalst and A. de Medeiros, Process mining and security: Detecting anomalous process execution and checking process conformance, *Electronic Notes in Theoretical Computer Science*, vol. 121, pp. 3–21, 2005.

[15] J. Wang, Petri nets for dynamic event-driven system modeling, in *Handbook of Dynamic System Modeling*, P. Fishwick (Ed.), Chapman and Hall/CRC, Boca Raton, Florida, pp. 24-1–24-17, 2007.

[16] T. Wu and J. Nurse, Exploring the use of PLC debugging tools for digital forensic investigations of SCADA systems, *Journal of Digital Forensics, Security and Law*, vol. 10(4), pp. 79–96, 2015.

[17] K. Yau and K. Chow, PLC forensics based on control program logic change detection, *Journal of Digital Forensics, Security and Law*, vol. 10(4), pp. 59–68, 2015.

[18] K. Yau and K. Chow, Detecting anomalous programmable logic controller events using machine learning, in *Advances in Digital Forensics XIII*, G. Peterson and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 81–94, 2017.

[19] K. Yau, K. Chow and S. Yiu, A forensic logging system for Siemens programmable logic controllers, in *Advances in Digital Forensics XIV*, G. Peterson and S. Shenoi (Eds.), Springer, Cham, Switzerland, pp. 331–349, 2018.

[20] K. Yau, K. Chow, S. Yiu and C. Chan, Detecting anomalous behavior of a PLC using semi-supervised machine learning, *Proceedings of the IEEE Conference on Communications and Network Security*, pp. 580–585, 2017.

[21] W. Yew, PLC Device Security – Tailoring Needs, White Paper, SANS Institute, Bethesda, Maryland (`sansorg.egnyte.com/dl/aN9oVirLPG`), 2021.