Chapter 7

FORENSIC ANALYSIS OF A SIEMENS PROGRAMMABLE LOGIC CONTROLLER

Raymond Chan and Kam-Pui Chow

Abstract Programmable logic controllers are widely used in industrial control systems and supervisory control and data acquisition (SCADA) systems. As the potential of cyber attacks on programmable logic controllers increase, it is important to develop robust digital forensic techniques for investigating potential security incidents involving programmable logic controllers. This chapter focuses on the logging mechanism of a Siemens programmable logic controller, specifically the Siemens Total Integrated Automation Portal V13 program (Siemens TIA Portal, also called Siemens Step-7).

Keywords: Control systems, programmable logic controllers, forensic analysis

1. Introduction

Industrial control systems are vital to the operation of the critical infrastructure. Programmable logic controllers (PLCs), which are among the most commonly used components of industrial control systems, are used to monitor processes and perform control actions. Programmable logic controllers are usually connected to human-machine interfaces (HMIs) to enable remote real-time monitoring and control by human operators. Although modern industrial control systems have been used for several decades, little research has focused on forensic analysis methodologies for investigating security incidents involving control systems. The discovery of Stuxnet [10] in 2010 has significantly increased efforts to develop sophisticated and reliable forensic techniques for industrial control systems, including programmable logic controllers. These techniques are vital to understand the nature and scope of security incidents and attacks, extract evidence and potentially identify the perpetrators.

This chapter focuses on a Siemens programmable logic controller, namely the Siemens Total Integrated Automation Portal V13 program (Siemens TIA

M. Rice and S. Shenoi (Eds.): Critical Infrastructure Protection X, IFIP AICT 485, pp. 117–130, 2016.
DOI: 10.1007/978-3-319-48737-3_7

Portal, also called Siemens Step-7), one of the systems targeted by Stuxnet. In particular, this chapter discusses the information available in the Siemens TIA Portal that could support forensic investigations and presents a forensic analysis methodology for the Siemens programmable logic controller.

2. Related Work

Forensic analyses of industrial control systems are challenging due to the lack of access to and knowledge about proprietary devices and protocols [1]. Taveras [7] has proposed a high-level model for live SCADA system forensics. Spyridopoulos et al. [6] have discussed the implementation of logging capabilities in a typical SCADA system architecture to support forensic investigations. Eden et al. [3] have presented an incident response taxonomy that includes possible attacks and forensic analysis methodologies for SCADA systems. Wu et al. [9] have proposed a forensic capability for the Siemens S7 programmable logic controller that uses the Siemens TIA Portal to monitor changes to data. Patzlaff [5] has developed a forensic analysis framework for industrial control systems that covers programmable logic controllers as well as host computers and workstations.

SCADA network forensics is also a topic of considerable interest among researchers and practitioners. Kilpatrick et al. [4] have proposed a SCADA network forensic architecture for analyzing TCP/IP traffic between control devices. Valli [8] has developed a Snort intrusion detection system for SCADA networks, which is able to detect and respond to common network attacks.

Analyzing programmable logic controller firmware is also of value in incident response and forensic investigations. Basnight et al. [2] have discussed techniques for reverse engineering programmable logic controller firmware. However, the task is extremely time consuming and, due to the proprietary nature of the hardware and security mechanisms, it may not be possible to extract useful information for forensic investigations.

Little, if any, research has focused on practical approaches for performing forensic analyses on programmable logic controllers. To address the gap, this chapter presents a practical methodology for analyzing a Siemens programmable logic controller along with a computer workstation installed with the Siemens TIA Portal.

3. Forensic Analysis Methodology

Figure 1 summarizes the forensic analysis methodology. First, the forensic investigator identifies the computer workstations (PCs) and programmable logic controllers (PLCs) involved in the security incident. Next, evidence is extracted from the workstations and programmable logic controllers for analysis. Further analysis has to be performed on the workstations that have the Siemens Total Integrated Automation Portal V13 program (Siemens TIA Portal, also called Siemens Step-7) installed. The identified programmable logic controllers must be connected to the Siemens TIA Portal for further analysis. This section

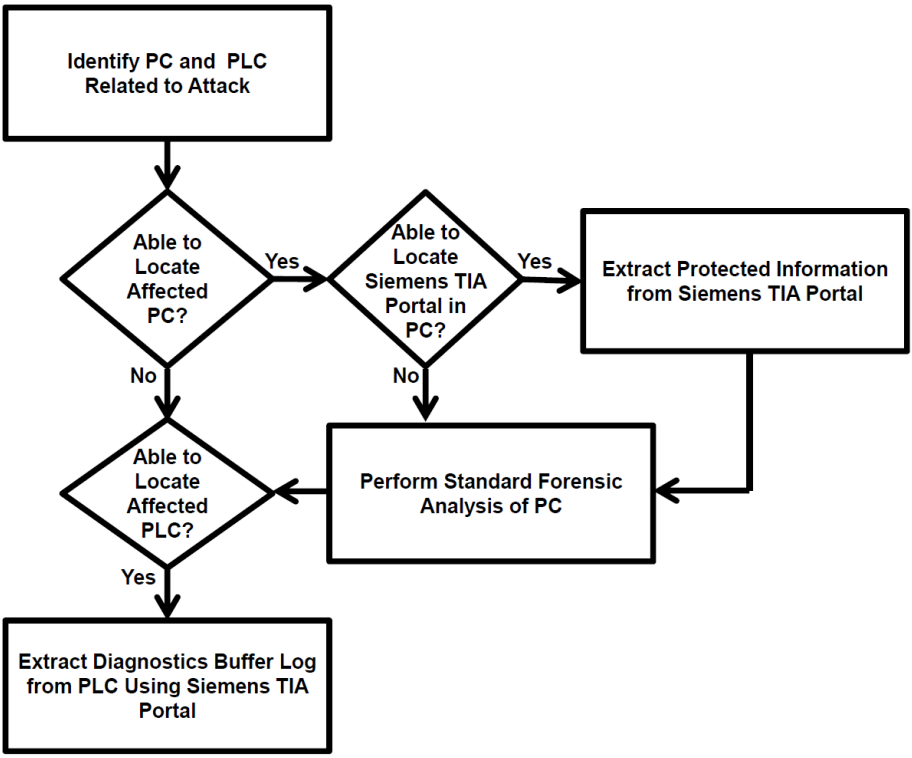

Figure 1. Forensic analysis methodology.

focuses on the forensic examination of the Siemens TIA Portal installed on a workstation and a method for examining the diagnostics buffer in an affected programmable logic controller.

3.1 Analyzing the Siemens TIA Portal

The Siemens TIA Portal is an integrated development environment for configuring and developing programs, and monitoring the status of the Siemens programmable logic controller. As such, it provides valuable information for forensic investigations.

3.2 Analyzing the `PEData.plf` File

The `PEData.plf` project file is generated by the Siemens TIA Portal. It contains information about the programmable logic controller program and configuration. The `PEData.plf` file is generated when a new programmable logic controller project is created. Because, the `PEData.plf` file records the programmable logic controller information in plaintext, any forensic tool (e.g., WinHex) can be used to examine the information.

PEData.plf

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
004C1FB0	00	00	FF	FF	FF	FF	01	00	00	00	04	00	00	00	00	00
004C1FC0	00	00	FF	51	00	00	00	00	00	00	00	10	00	00	00	00
004C1FD0	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00	30
004C1FE0	00	00	00	14	05	00	00	00	00	00	00	E3	05	00	00	01
004C1FF0	00	00	00	DE	06	B1	53	76	F8	D1	08	14	05	00	00	00
004C2000	00	00	00	E3	05	00	00	01	00	00	00	C7	A2	66	DC	76
004C2010	F8	D1	08	FF	31	00	00	00	00	00	00	00	0E	00	00	00
004C2020	00	00	00	00	00	00	00	00	01	00	00	00	00	00	00	00
004C2030	10	00	00	00	09	00	00	00	12	17	00	00	14	00	00	00
004C2040	00	00	00	00	FF	0A	24	24	43	4F	4D	4D	49	54	24	24
004C2050	28	04	69	DC	76	F8	D1	08	FF	0A	23	23	43	4C	4F	53
004C2060	45	23	23	23	01	6F	51	72	77	F8	D1	08	FF	27	04	00
004C2070	00	3A	10	20	00	9A	1B	00	00	00	00	00	00	33	00	00
004C2080	00	01	00	00	00	00	00	08	02	3C	00	00	00	00	00	00
004C2090	00	53	00	00	00	00	00	00	00	C5	00	00	00	0B	01	00
004C20A0	00	D2	03	00	00	00	00	00	00	17	00	00	00	0D	00	00
004C20B0	00	0C	00	00	00	01	0A	50	4C	55	53	42	4C	4F	43	4B
004C20C0	41	00	00	00	72	00	00	00	01	00	00	00	3A	10	20	00
004C20D0	01	00	00	00	00	00	00	00	29	00	10	00	00	00	00	00
004C20E0	00	00	00	00	00	00	00	10	01	00	00	00	01	01	00	00
004C20F0	00	C1	CB	42	8F	B9	14	D0	BC	93	C6	E3	58	44	0F	43
004C2100	A3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004C2110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004C2120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
004C2130	00	00	46	00	00	00	E3	09	00	00	00	00	00	00	1C	00
004C2140	00	00	31	00	00	00	F6	00	00	00	00	00	00	00	15	00
004C2150	00	00	00	00	00	00	00	00	00	00	00	00	00	10	01	00

Figure 2. Hex representation of the `PEData.plf` file.

Analysis of the file contents revealed that all the project actions are stored in the project file. Whenever the programmable logic controller program is modified, a PLUSBLOCK is appended to the end of the project file. The PLUSBLOCK provides information about the changes made to the programmable logic controller program. By comparing the variable tags and the ladder logic regions in the PLUSBLOCK, a forensic investigator can obtain details about the modifications made to the programmable logic controller program.

Figure 2 shows the file structure of the `PEData.plf` file; the actions and modifications are marked as COMMIT. It is possible to reconstruct the changes that have been applied to the project file.

Figure 3 shows the MAC times (last modification time, last accessed time and creation time) of the `PEData.plf` file. The information is used to determine the modification time, last access time and creation time of the project.

Figure 4 shows the locations of the ladder logic program and tags in the `PLUSBLOCK` file. This information also enables a forensic investigator to examine the changes made to the programmable logic controller program.

3.3 Analyzing the Program Block Metadata

Programmable logic controller programs are represented as program blocks in the Siemens TIA Portal. Each program block has its own metadata and

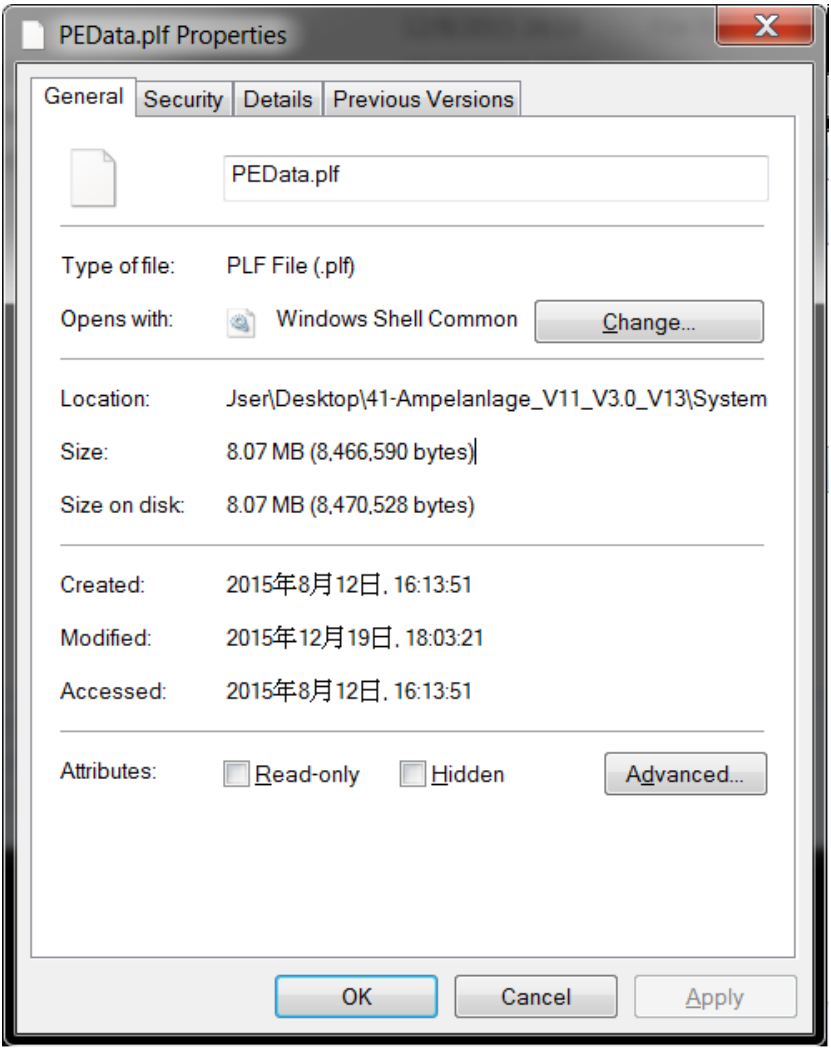

Figure 3. MAC times of the `PEData.plf` file.

attributes. The information includes the size of the binary file, last compilation date and last modified date of the programmable logic controller program.

After an incident occurs, a forensic investigator needs to verify the correctness of the timestamp information provided by the programmable logic controller. Several timestamps are provided by the Siemens TIA Portal. Figure 5 shows the compilation timestamp and the size of the program in memory.

Figure 6 shows several useful timestamps associated with a programmable logic controller program. The timestamps are:

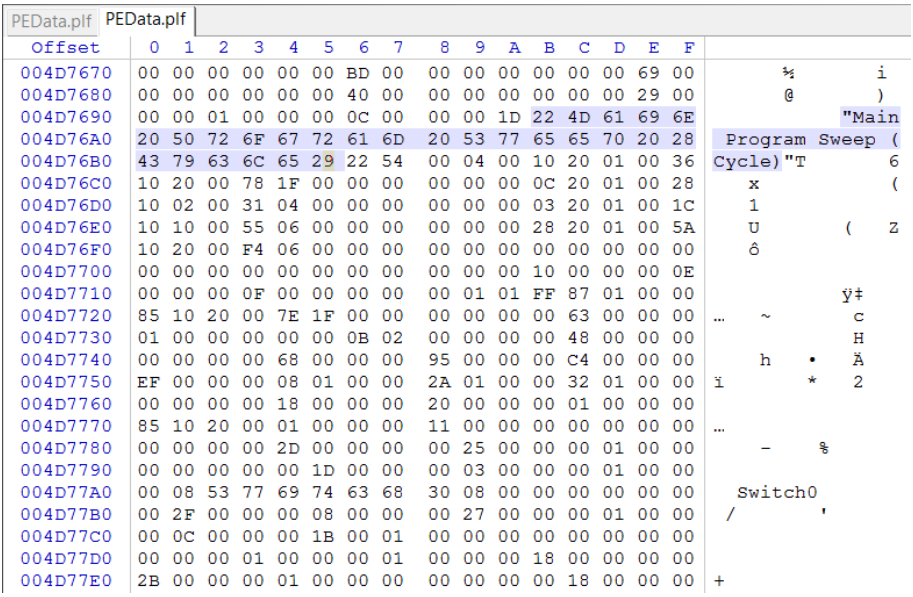

Figure 4. Storage locations of the ladder logic program and tags in `PEData.plf`.

- **Block:** This is the latest modification time of the programmable logic controller program. It is either the last modification time of the program block interface or the code/data, depending on which entity was modified last.

- **Interface:** This is the latest modification time of the interface of the programmable logic controller program. The interface timestamp is updated whenever the interface is modified.

- **Code/Data:** This is the latest modification time of the program logic or metadata of the programmable logic controller program. The code/data timestamp is updated when the program or metadata of the program block are changed.

- **Binary:** This is the latest modification time of the metadata of the program block. It corresponds to the time when the compiled binary component was loaded into the programmable logic controller.

The compilation time (compilation timestamp in Figure 5) and last loaded time (load-relevant timestamp in Figure 6) enable a forensic investigator to ascertain when a program was compiled and loaded on the programmable logic controller. In a normal situation, the last loaded time should be after the compilation time. If the last loaded time is earlier than the compilation time, then the program was (possibly updated and) compiled, but not yet loaded on the programmable logic controller.

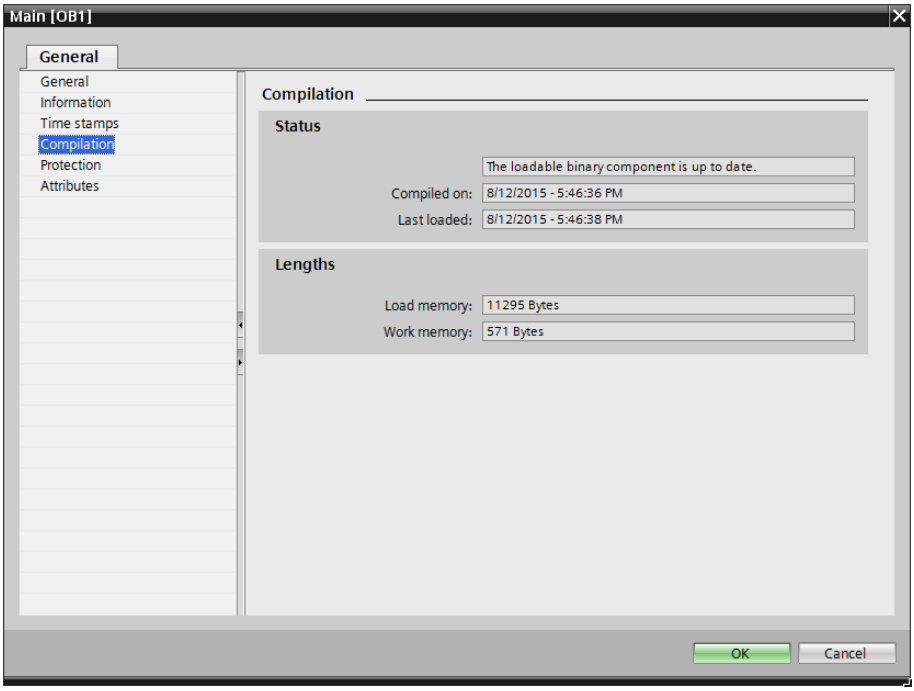

Figure 5. Program compilation information.

4. Analyzing the Diagnostics Buffer

The Siemens programmable logic controller has a diagnostics buffer that records its behavior and interactions with the Siemens TIA Portal. For each event, the diagnostics buffer records the timestamp, event id and detailed description of the event. Since the buffer is read-only, it is not possible for an attacker to modify its contents. Due to the limited memory in the programmable logic controller, the diagnostics buffer can only record about 1,300 to 3,000 recent events. During normal operation, the diagnostics buffer should be able to record an adequate number of programmable logic controller events for forensic analysis. During an investigation, a forensic professional should switch the programmable logic controller from the RUN mode to the STOP mode before examining the diagnostics buffer. Omitting this step may cause information about the earliest events to be overwritten by new events. Figure 7 shows the event log maintained by the diagnostics buffer.

4.1 Starting and Stopping the Controller

The Siemens programmable logic controller has three modes: (i) STARTUP; (ii) STOP; and (iii) RUN. When the programmable logic controller starts up with a pre-loaded program, the diagnostics buffer records the mode change

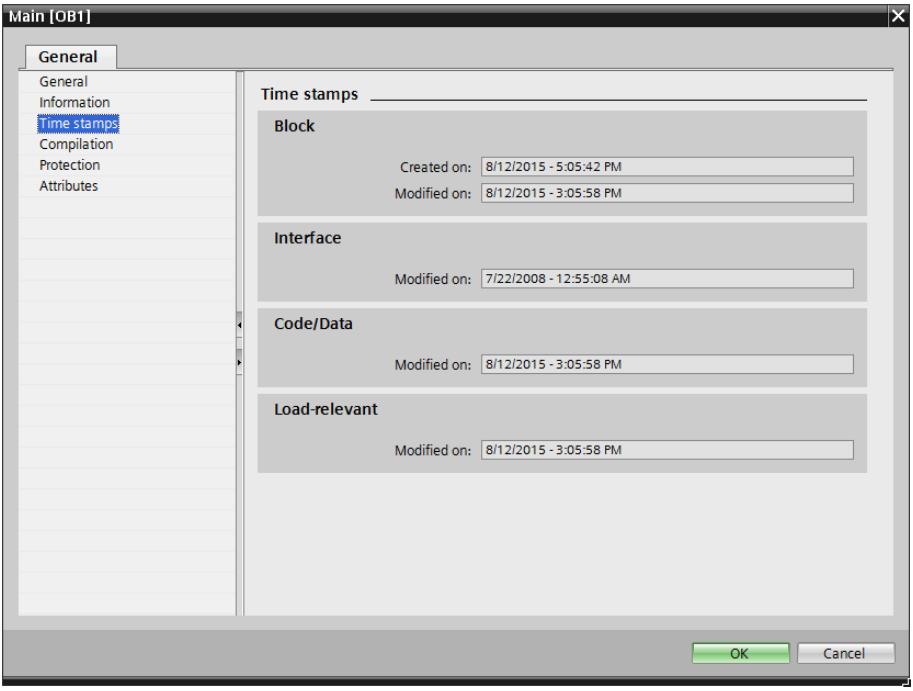

Figure 6. Program timestamp information.

from STARTUP to RUN. If no program has been pre-loaded, then the mode change from STARTUP to STOP is recorded. The Siemens TIA Portal can send commands to the programmable logic controller to change its mode. Two events are recorded after a STOP command is sent to the programmable logic controller and three events are recorded after a RUN command is sent to the programmable logic controller. Figure 8 shows the diagnostics buffer event log after the STOP and RUN commands are sent to the programmable logic controller.

4.2 Uploading a New Program

In order to upload a new program to the programmable logic controller, the Siemens TIA Portal has to change the programmable logic controller from the RUN mode to the STOP mode, overwrite the existing program with the new program and change the programmable logic controller mode to STARTUP. After the program is successfully uploaded to the programmable logic controller, the Siemens TIA Portal issues a WARM RESTART command to change the mode from STARTUP to RUN, thereby enabling the newly-installed program to execute. The diagnostics buffer records a total of seven events until the upload action is completed. Figure 9 shows the corresponding event log in the diagnostics buffer.

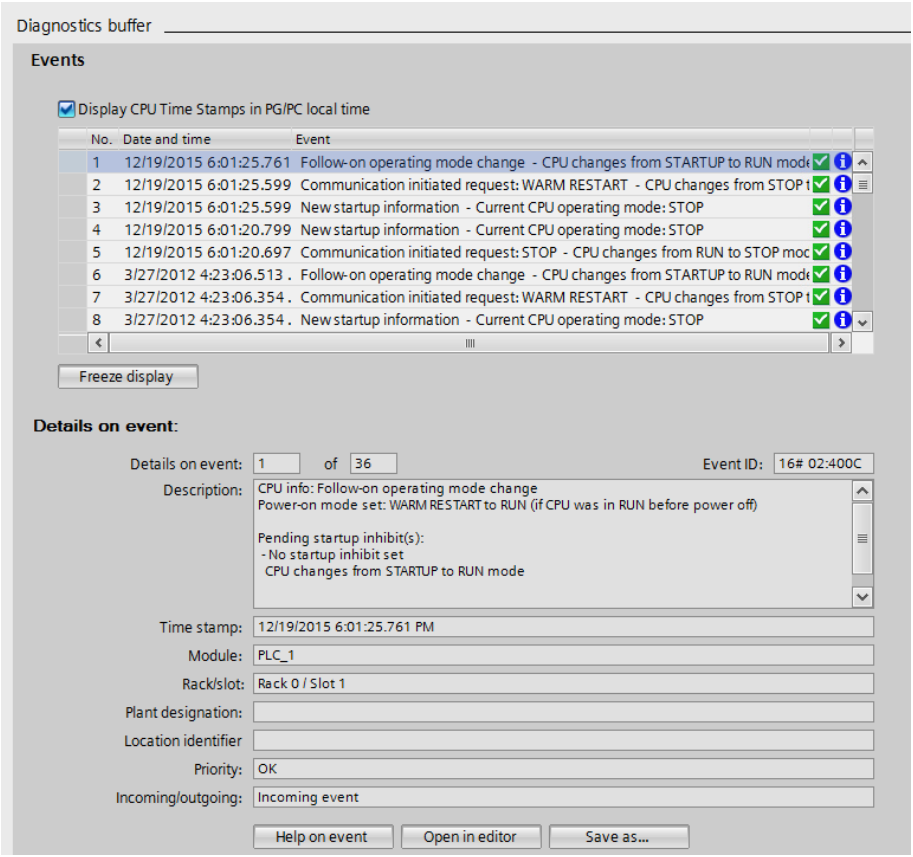

Figure 7. Diagnostics buffer event log.

4.3 Analyzing Engineer-Defined Events

Other information is available that may be useful to determine the state of the programmable logic controller. For example, an engineer can set the diagnostics buffer to log events related to the running status of the programmable logic controller that are helpful in investigating incidents. Figure 10 shows examples of engineer-defined events that can be logged by the programmable logic controller.

5. Case Study

This section describes a hypothetical, albeit realistic, security incident in which the proposed forensic analysis methodology is used to conduct the investigation.

An engineer was dismissed from his position at a water supply company as a result of unsatisfactory performance. On December 26, 2015, the company

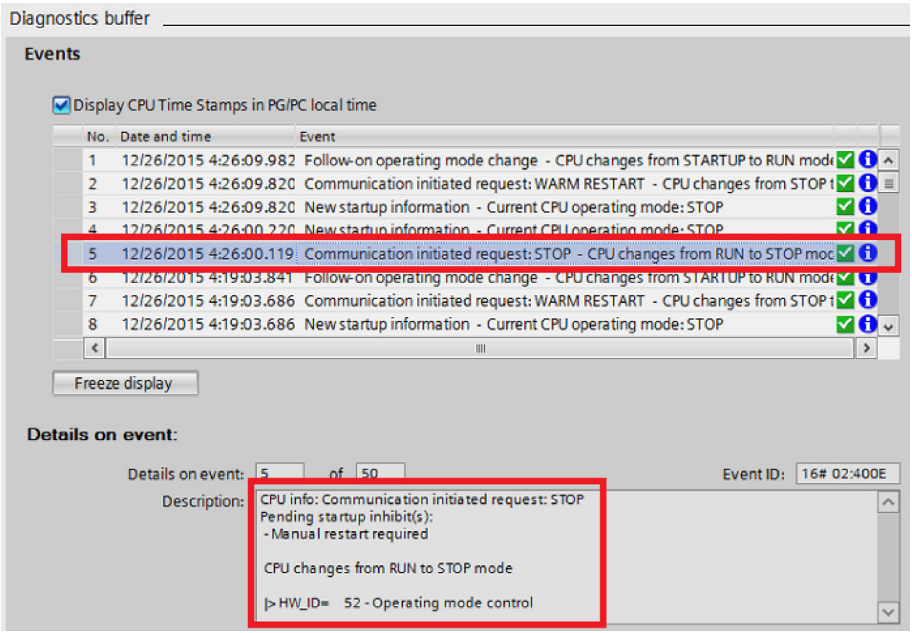

Figure 8. Event log generated by sending a STOP command.

incident response team received a call that the water supply system was down due to a malfunctioning programmable logic controller. The incident response team was requested to investigate whether or not the system was attacked.

After examining the CCTV recording, the dismissed engineer was identified as the primary suspect. Specifically, he was seen to access a workstation with the Siemens TIA Portal installed. The incident response team believed that the engineer had modified the programmable logic controller program, which caused the water supply system failure. The incident response team was tasked with identifying the actions performed by the engineer and their times by examining the digital traces left in the Siemens TIA Portal and the diagnostics buffer of the programmable logic controller.

The incident response team hypothesized that the engineer modified the program in question using the workstation and then uploaded the program to the programmable logic controller. The team performed the following actions and retrieved the following evidence according to the proposed methodology:

- The incident response team examined the workstation used by the engineer, which had the Siemens TIA Portal was installed. In the Siemens TIA Portal, the team discovered that the last modification time of the programmable logic controller program was 8/12/2015 3:05:58 PM, the compilation time was 8/12/2015 5:46:36 PM and the last loaded time was 8/12/2015 5:46:38 PM. These timestamps enabled the team to identify when the program had been modified and uploaded to the programmable

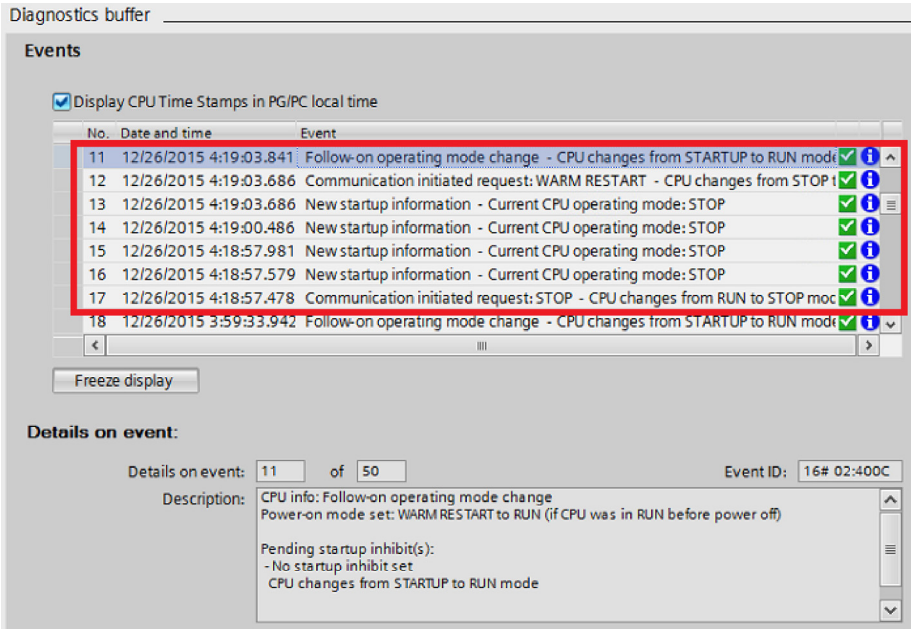

Figure 9. Event log generated by uploading a program.

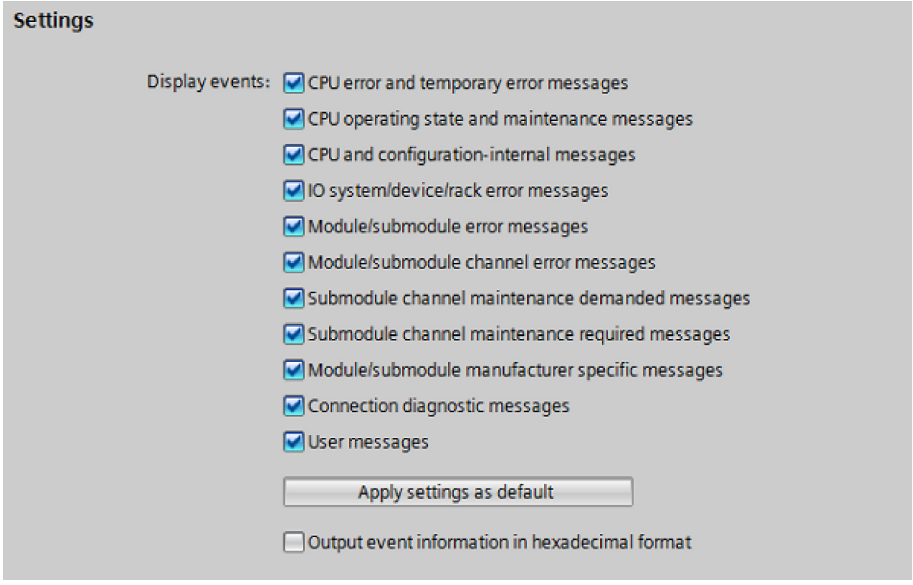

Figure 10. Engineer-defined events recorded by the diagnostics buffer.

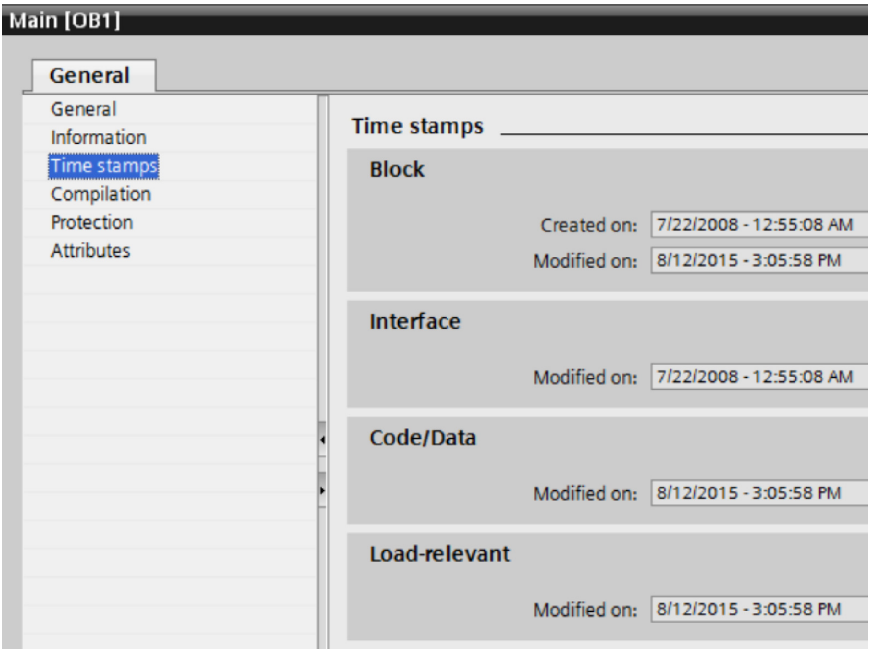

Figure 11. Program block timestamps in the Siemens TIA Portal.

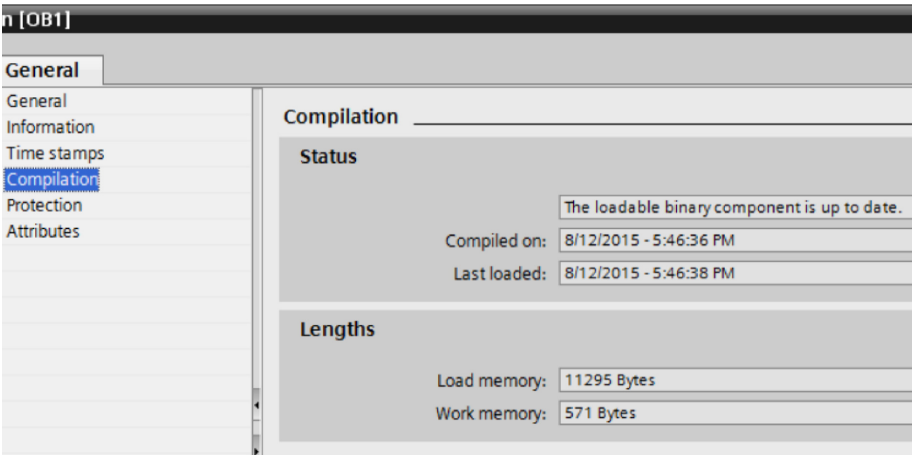

Figure 12. Program compilation/uploading timestamps in the Siemens TIA Portal.

logic controller. Figures 11 and 12 show screenshots of the program block information in the Siemens TIA Portal.

- The incident response team proceeded to identify the programmable logic controller with the abnormal behavior. After connecting to the pro-

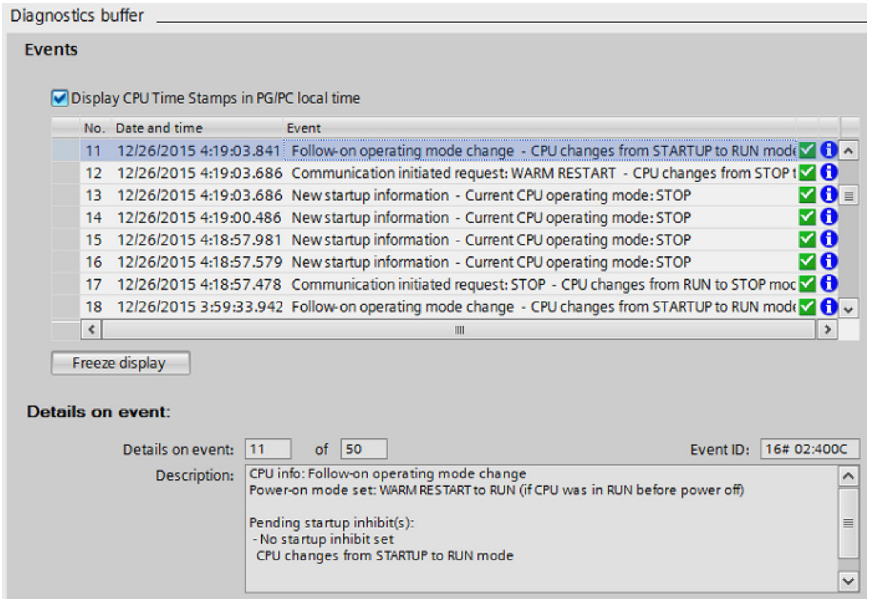

Figure 13. Diagnostics buffer in the malfunctioning programmable logic controller.

grammable logic controller, the team extracted the event log in the diagnostics buffer and discovered that the programmable logic controller had been stopped and the modified program was executed on 26/12/2015 4:18:57 AM. The information provided by the diagnostics buffer proved that the program was modified on 8/12/2015, which matched the date of the water supply system failure. Figure 13 shows a screenshot of the diagnostics buffer in the malfunctioning programmable logic controller.

Using the proposed forensic analysis methodology, the incident response team successfully extracted the programmable logic controller program modification time and compilation time from the Siemens TIA Portal to confirm when the program had been changed. By retrieving the event log from the diagnostics buffer in the programmable logic controller, the incident response team confirmed the time when the programmable logic controller was restarted. Upon comparing the program retrieved from the malfunctioning programmable logic controller with the original program, the incident response team was able to discover exactly how the program was modified and exactly what caused the programmable logic controller to malfunction. All the events were placed on a timeline by comparing the MAC times corresponding to the original program and the modified program.

6. Conclusions

The discovery of Stuxnet in 2010 has significantly increased efforts to develop sophisticated forensic techniques for industrial control systems. These

techniques are vital to understand the nature and scope of security incidents and attacks, extract evidence and potentially identify the perpetrators. The proposed methodology for performing forensic analyses of programmable logic controllers is effective and practical. In particular, it focuses on a Siemens programmable logic controller along with a computer workstation installed with the Siemens TIA Portal, one of the systems targeted by Stuxnet. Future research will extend the forensic analysis methodology to cover other programmable logic controller models and their associated firmware and software.

References

[1] I. Ahmed, S. Obermeier, M. Naedele and G. Richard, SCADA systems: Challenges for forensic investigators, *IEEE Computer*, vol. 45(12), pp. 44–51, 2012.

[2] Z. Basnight, J. Butts and T. Dube, Analysis of programmable logic controller firmware for threat assessment and forensic investigation, *Journal of Information Warfare*, vol. 12(2), 2013.

[3] P. Eden, A. Blyth, P. Burnap, Y. Cherdantseva, K. Jones, H. Soulsby and K. Stoddart, A forensic taxonomy of SCADA systems and approach to incident response, *Proceedings of the Third International Symposium on ICS and SCADA Cyber Security Research*, pp. 42–51, 2015.

[4] T. Kilpatrick, J. Gonzalez, R. Chandia, M. Papa and S. Shenoi, An architecture for SCADA network forensics, in *Advances in Digital Forensics II*, M. Olivier and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 273–285, 2006.

[5] H. Patzlaff, D7.1 Preliminary Report on Forensic Analysis for Industrial Systems, CRISALIS Consortium, Symantec, Sophia Antipolis, France, 2013.

[6] 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*, 2013.

[7] P. Taveras, SCADA live forensics: Real time data acquisition process to detect, prevent or evaluate critical situations, *Proceedings of the First Annual International Interdisciplinary Conference*, pp. 253–262, 2013.

[8] C. Valli, Snort IDS for SCADA networks, *Proceedings of the International Conference on Security and Management*, pp. 618–621, 2009.

[9] T. Wu, J. Pagna Disso, K. Jones and A. Campos, Towards a SCADA forensics architecture, *Proceedings of the First International Symposium on ICS and SCADA Cyber Security Research*, pp. 12–21, 2013.

[10] K. Zetter, *Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon*, Crown, New York, 2014.