Volatility PLC Frame

Charles Beddingfield Digital Forensics, Spring 2020

Background

Attacks on critical infrastructure represent a growing threat to US National security. Critical infrastructure is defined as: “Critical infrastructure describes the physical and cyber systems and assets that are so vital to the United States that their incapacity or destruction would have a debilitating impact on our physical or economic security or public health or safety.”[1] The Department of Homeland Security Cyber Infrastructure Security Agency divides critical infrastructure into 17 sectors, including Chemical and Energy sectors.

A key part of securing industrial sectors from cyber attack is the protection of Industrial Control Systems (ICS), a broad classification for technologies that “systems that are used to monitor and control industrial processes.”[2] ICS are further divided into: Supervisory Control and Data Acquisition (SCADA) systems, often based on existing desktop operating systems; Control units; communication links, often Ethernet; and Remote Terminal Units, which are often a class of device known as Programmable Logic Controllers (PLCs). According to Lewis Folkerth for the SANS institute, “One common mistake in developing forensic techniques for ICS is to concentrate on the central server of a DCS or SCADA system.” Folkerth points out a dearth of forensic tools for the purpose of detecting and responding to the compromise of PLCs. [3]

PLC Frame attempts to address some of the difficulties associated with forensic examination of PLCs suspected of infection by malware. First, it takes as its starting point the defining characteristic of PLCs: the interaction and connection with sensors or actuators. Second, PLC Frame establishes a ‘mini-framework’ within Volatility. PLCs are notoriously diverse in their architecture and implementation. PLC Frame allows for a central plugin prioritizing the most important analysis steps, and is agnostic as to the specific target of examination. Third, in its current state of development, it leverages existing plugins to look for evidence of one of the more dangerous classes of PLC malware.

Programmable Logic Controllers as Digital Evidence

PLCs present unique difficulties for the forensic examiner. First, PLCs are a class of computing device, rather than a specific implementation. They are produced by a variety of manufacturers, and by definition require a level of customization to meet the needs of the industry consumer. In general, PLCs are input/output controllers with finite states based on cyclical logic. PLCs make use of input and output “pins” through which they interact with physical actuators and sensors. It is this cyber-physical nature of PLCs that makes them both attractive to attackers, and particularly dangerous. By manipulating the execution of the controller, an attacker can create real physical effects.

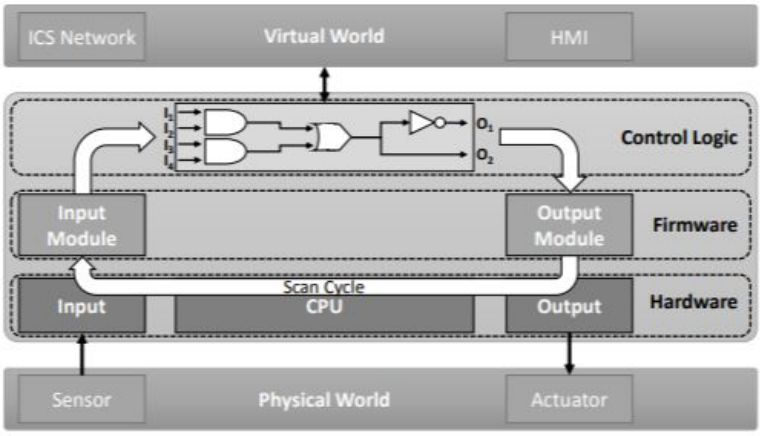
In a broad sense, most controllers used in industry process inputs and outputs in the following manner:

---

[1] https://www.dhs.gov/topic/critical-infrastructure-security

[2] https://www.kuppingercole.com/blog/williamson/ot-ics-scada-whats-the-difference

[3] Forensic Analysis of Industrial Control Systems, SANS Reading Room, Folkerth Lewis 2015



Source: Garcia, L. A., Brasser, F., Cintuglu, M. H., Sadeghi, A., Mohammed, O., & Zonouz, S. A. (2017). Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. *Proceedings 2017 Network and Distributed System Security Symposium*. doi:10.14722/ndss.2017.23313

In addition to diverse make, PLCs often ship with proprietary firmware. While there are standards for the logic programmed onto PLCs, the firmware can be proprietary and poorly documented.[4] Finally, PLCs make use of a variety of protocols to communicate with sensors and acuators.[5]

While these factors present barriers to a would-be attacker, there are documented cases of attacks on cyber-physical processes through PLCs.[6] [7] Furthermore, academic research of PLC-oriented malware has developed in recent years, and several types of malware classes have been theorized.[8] [9] This project will focus on a generalization of the way Stuxnet hooked into PLC functions to modify inputs and outputs, as well as communication with management devices.[10]

Many of the defensive methods for PLCs focus on detection of Malware on running devicesThis approach makes sense, as PLCs are often expected to operate with minimal interruption. Even for intrusion detection on running systems, many researchers have pointed out the inputs and outputs of a PLC present the most logical starting point for analysis.[11]

The same barriers to developing reliable attacks on PLCs also limit researchers and investigators' ability to detect such malware. Many existing forensic tools do not support analysis of evidence from PLCs. The diverse implementations of PLC firmware make the barrier

---

[4] https://plcopen.org/guidelines/guidelines
[5] https://resources.infosecinstitute.com/category/certifications-training/ics-scada/ics-protocols/#gref
[6] https://www.us-cert.gov/ics/content/cyber-threat-source-descriptions
[7] https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/
[8] Mclaughlin, Stephen. (2011). On dynamic malware payloads aimed at programmable logic controllers. 10-10.
[9] Govil, N., Agrawal, A., & Tippenhauer, N. O. (2017). On Ladder Logic Bombs in Industrial Control Systems. *Computer Security Lecture Notes in Computer Science*, 110–126. http://doi.org/10.1007/978-3-319-72817-9_8
[10] https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=ad4b3d10-b808-414c-b4c3-ae4a2ed85560&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments
[11] S. Zonouz, J. Rrushi and S. McLaughlin, "Detecting Industrial Control Malware Using Automated PLC Code Analytics," in *IEEE Security & Privacy*, vol. 12, no. 6, pp. 40-47, Nov.-Dec. 2014, doi: 10.1109/MSP.2014.113.

to designing these tools, and deploying them in a broad way, incredibly difficult. Analysis of volatile memory artifacts is further complicated by the unique setup of each ICS solution, and investigators must account for which artifacts represent potential manipulation of inputs and outputs.

Experimental Design

For this project, lack of access to open-source PLC software which could be virtualized presented the primary obstacle. The solution was OpenPLC, designed by Thiago Alves. OpenPLC is a completely open-source industrial control runtime for linux-based devices like Raspberry Pi. OpenPLC has been used for security research, and is maintained and easy to customize. Most importantly, OpenPLC is fairly well-documented, and can be easily virtualized.[12]

To demonstrate somewhat realistic ICS behavior, this project connected a virtualized instance of OpenPLC running on Raspberry Pi 4, and networked it to Factory I/O, a factory-floor simulator which can connect to a variety of PLC devices, and simulate actuator and sensor behavior.

Design Overview

PLC_Frame attempts to address the issues surrounding analysis of PLC volatile memory by abstracting the process, and giving users a tool to configure for specific ICS implementations. First, the user encodes in a JSON file information about the firmware, the communication protocol in use, and unique identifiers for sensors and actuators to refine the search. Next, PLC Frame takes the configuration file and returns to the framework a “strategy class” with functions specific to the given implementation. Finally, the mini-framework executes a set of analysis steps tailored to the ICS solution, and attempts to reveal artifacts of malware infections.

This project leveraged refined output from existing plugins like linux_netstat and linux_plthook to show how an investigator or a technician could quickly determine if a particular PLC shows evidence of malware attempting to modify cyber-physical processes. Because the mini-framework abstracts the actual functions and plugins used, PLC Frame is intended to be extensible within the broader Volatility project. Specialists from, say Siemens or Allen Bradley could add configuration classes to the ‘resources’ file of PLC Frame, and researchers can customize functions for the abstracted analysis steps. Most importantly, a technician investigating potential infection of an industrial process will not need to know particulars of memory analysis of their particular device. By modifying the configuration file, the technician could quickly extract salient artifacts from a suspect device.

Results

PLC Frame currently focuses on extracting evidence of malware which modifies values to and from the IO devices attached to a PLC. The design of OpenPLC presents its own set of challenges to this approach. As noted above, PLC firmware typically includes modules for

[12] Alves, Thiago & Morris, Thomas. (2018). OpenPLC: An IEC 61131-3 Compliant Open Source Industrial Controller for Cyber Security Research. Computers & Security. 78. 10.1016/j.cose.2018.07.007.

particular input output systems. Values are written to memory locations corresponding to manufacturer specifications[13] OpenPLC, on the other hand, is an executable written in C++, which means it relies heavily on resources delivered by the underlying operating system at runtime to communicate with a sensor and actuator server. These complications are magnified in a virtualized instance.

For these reasons, PLC_Frame blurs the distinction between the runtime and the operating system, into a general "firmware". Of the existing classes of malware specific to PLCs, PLC Frame addresses malware which hooks into functions in the OpenPLC runtime communication libraries, and attempts to modify parameters (register and coil values).[14] On proprietary industrial control devices, this is a relatively novel threat. For a runtime executing on a Linux OS, this is an established and respected attack methodology. For this reason, when analyzing OpenPLC on a Linux OS using Modbus TCP to communicate with devices, PLC Frame can strategically use existing Volatility plugins to find all processes connected to pre-configured remote devices (our factory IO server). It then leverages linux_plthook by Georg Wicherski to examine the ELF file involved in network communication.[15]

PLC_Frame inherits limitations of the broader Volatility Framework. It is currently built to analyze PLC memory images from Linux-based devices. However, the availability of open-source PLC firmware solutions, based on more familiar operating systems is growing.[16] [17] Volatility currently supports dozens of memory profiles. As manufacturers shift to more standardized platforms, and as new, dangerous malware causes physical damage, Volatility will have good reason to support analysis of these systems.

Conclusion

PLC_Frame lays the groundwork for a practical, modular plugin which facilitates the addition of tools to analyze the dizzying variety of industrial control solutions. Its limited scope, to communication hooking presents a possible model for other plugins for additional malware artifacts by abstracting details of analysis of specific software. Finally, it demonstrates the validity of this approach by addressing a realistic threat to the OpenPLC runtime communicating over TCP Modbus.

---

[13] https://instrumentationtools.com/plc-memory-mapping-io-addressing/

[14] Reeves, J., Ramaswamy, A., Locasto, M., Bratus, S., & Smith, S. (2012). Intrusion detection for resource-constrained embedded control systems in the power grid. *International Journal of Critical Infrastructure Protection, 5*(2), 74-83. doi:10.1016/j.ijcip.2012.02.002

[15] https://sites.google.com/site/bletchleypark2/malware-analysis/malware-technique/rootkit/plt-hook-detection-with-volatility?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1

[16] https://www.rexygen.com/pigeon-plc-rb100-rb300-rexygen-platform/

[17] https://www.industrialshields.com/