Chapter 4

ANALYSIS OF FIELD DEVICES USED IN INDUSTRIAL CONTROL SYSTEMS

John Mulder, Moses Schwartz, Michael Berg, Jonathan Van Houten, Jorge Mario Urrea and Alex Pease

Abstract A significant portion of the critical infrastructure relies on the proper operation of industrial control system (ICS) field devices. Unfortunately, security solutions for ICS field devices have not progressed sufficiently to address emerging threats. A primary shortfall is the ability to identify device components and analyze their lower level functionality. This paper describes the results obtained from hardware tear-downs of ICS field devices. The results demonstrate the ability to identify key components, analyze device firmware and examine backplane protocols – all necessary steps for the dynamic analysis and development of automated security solutions.

Keywords: Industrial control systems, device analysis, firmware analysis

1. Introduction

Industrial control system (ICS) field devices monitor and control physical processes in the critical infrastructure. Prior to Stuxnet [2], ICS security efforts focused primarily on human-machine interfaces (HMIs) and other supervisory control and data acquisition (SCADA) software. The high-profile attack, however, demonstrates the lack of security associated with ICS field devices. Meanwhile, there has been relatively little research focused on analyzing vulnerabilities associated with these critical assets.

Although many security solutions exist for analyzing software on commodity personal computers, limited tools are available and only a shallow understanding exists of the vulnerabilities related to ICS field devices. The vulnerabilities, however, do exist – initial research on ICS field devices has identified critical security flaws (e.g., hard-coded passwords extracted from firmware images, unauthenticated firmware uploads, multiple unauthenticated interfaces, and weak password hashing) [1, 3–6, 8]. As an example, in January 2012, a coalition of security researchers released a set of zero-day vulnerabilities targeting seven embedded ICS devices [7]. Therefore, it is imperative that future security research focus on the analysis of the firmware and hardware implementations.

This paper describes a process for analyzing ICS field devices. Specifically, the components of various programmable logic controllers (PLCs) are examined to derive fundamental attributes that relate to common design characteristics. Note that this analysis is not intended to identify specific vulnerabilities or design flaws. Rather, it seeks a deeper understanding of device design, which is a prerequisite for identifying attack surfaces. The approach is consistent with an adversarial viewpoint – no attempts were made to leverage inside knowledge or obtain vendor cooperation. The analysis of a device was performed by disassembling the device, cataloging its electronic components and creating a functional diagram. The electrical connectivity between the pins was then examined to verify device interconnections. After characterizing the hardware, several techniques were used to extract the firmware, including locating firmware updates, connecting via debug ports and reading the flash memory contents using a chip programmer. Finally, methods were explored for examining the backplanes used for communications between PLC subcomponents.

2. PLC Overview

This paper primarily focuses on PLCs; however, the methodology and results are applicable to a wide range of embedded devices. Indeed, many types of automation equipment have similar control capabilities. For example, the primary automation components used in electrical substations are remote terminal units (RTUs). RTUs perform data aggregation and protocol conversion for the other devices in a substation. Many RTUs, sometimes called real-time automation controllers or substation controllers, execute the same control logic as a PLC. Note that PLC logic is usually written in a language defined in IEC 61131-3 such as Ladder Logic or Instruction List. The primary differences between PLCs and RTUs are in their target markets and configuration details (e.g., input and output specifications).

Modular PLCs, as demonstrated in Figure 1, comprise discrete modules that are connected via a backplane. The processor module reads values from the communications and input/output (I/O) modules, interprets and executes the control logic, and writes values to the communications and I/O modules. The communications modules dissect complex communications-protocol-specific code. Because some control system protocols are extremely complex, communications modules may possess significant processing power and intelligence. I/O modules convert signals between low voltage (3.3V DC or 5V DC), low current (milliamps) control logic and high voltage (24+V DC), high current (amps) process control. Additionally, analog I/O modules contain analog-to-digital converters and digital-to-analog converters.

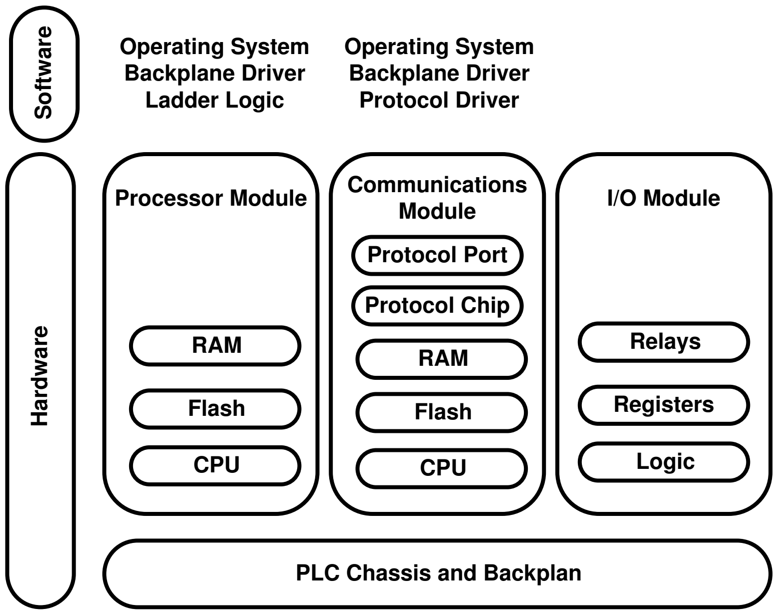

Figure 1. Generic modular PLC.

3. Hardware Analysis

PLC hardware is characterized by disassembling a device, photographing and cataloging the components, researching the parts, and determining the electrical connectivity between components. Although some embedded devices have minimal components that are easy to identify, even a simple PLC has many electronic parts. As such, a heuristic technique was used to discern the components of interest (e.g., memory and processor). The heuristic includes pin-count (higher is more interesting), type of chip package (ball grid array is often used for high-end components), proximity to other interesting parts, and chip markings. Although the heuristic technique approximates the significance of each component, it is adequate for this research effort.

Internet search queries were used for component identification, in particular, to correlate model numbers, brand markings, chip packages and pin counts. After identifying the components, a logical view of the PLC was derived. Component functions and connectivity were discerned using technical data sheets and tracings on the physical circuit card. Findings were verified using an ohmmeter to determine the connections between pins on different chips.

3.1 Example Device Analysis

An Allen-Bradley ControlLogic (Logix) PLC manufactured around 2005 is used to illustrate the methodology (Figure 2). It is one of the most popular

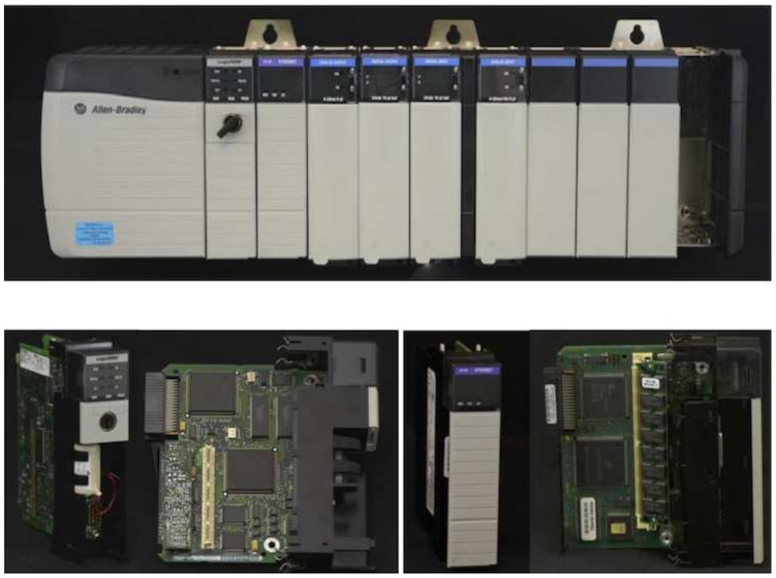

Figure 2. Allen-Bradley Logix PLC.

lines of modular PLCs in North America. The PLC chassis is shown at the top of the Figure 2; the Logix 5555 processor is at the bottom left and the EtherNet/IP modules are at the bottom right. The PLC has a chassis with a power supply and slots for a control module, network communication modules and I/O modules. Figure 3 shows the logical diagram for the connections between the modules and components. This representation clarifies the roles of the various PLC components.

Annotated photographs help clarify the physical layout of the device components – with sufficiently detailed photographs, it is possible to trace the connections on the top and bottom layers of the printed circuit board. Figure 4 shows an annotated photograph of the Logix 5555 processor module and Figure 5 shows an annotated photograph of the EtherNet/IP module. The photographs are each linked to part lists in Tables 1 and 2, respectively. The part name references the data sheet descriptor. The manufacturer part number is a combination of the brand and part number based on the data sheet, when available, or trademarks found on a chip. Note that the markings are the same symbols that are found on the surfaces of the chips.

3.2 General Findings

The majority of field devices we analyzed used production chips from large manufacturers; only a few of the chips were variants produced for a specific vendor. Additionally, we discovered that configurations of flash memory and RAM are consistent with other embedded systems and typically use fairly sim-

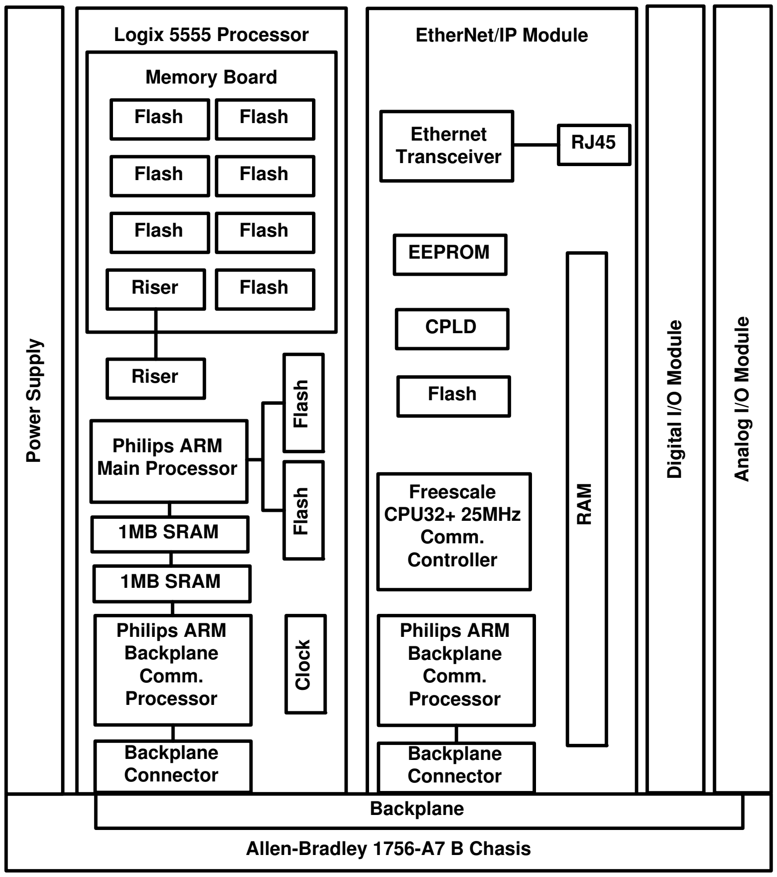

Figure 3. Logical component diagram of the Allen-Bradley Logix PLC.

ple two- to four-layer boards, whereas normal personal computer boards often have seven layers. Despite this simplicity, the interconnections can be difficult to discern primarily because the board layout is driven by efficiency.

The most common processor architectures identified for ICS field devices were ARM, PowerPC and Motorola 68k. However, we also discovered that many devices are based on x86 processors, sometimes using commodity PC/104 form factor embedded computers to provide processing power. Multiple processor architectures on a single board are also common. For example, one device uses a Freescale PowerPC main processor and a separate ARM backplane communication processor. This is not a surprising configuration; however, deep analysis efforts require expertise in multiple architectures.

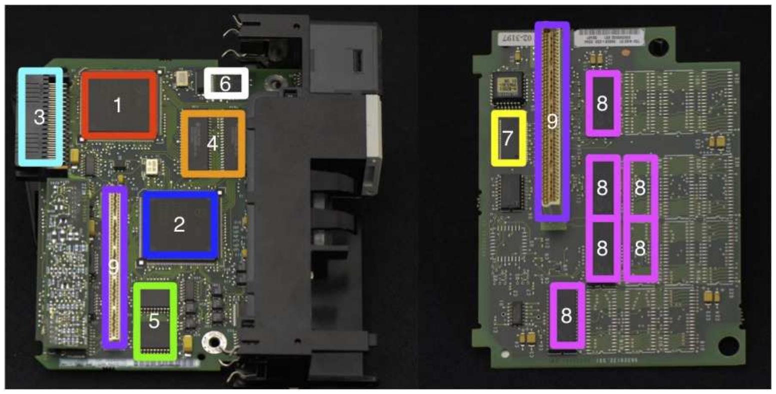

Figure 4. Allen-Bradley Logix 5555 processor module (annotations in Table 1).

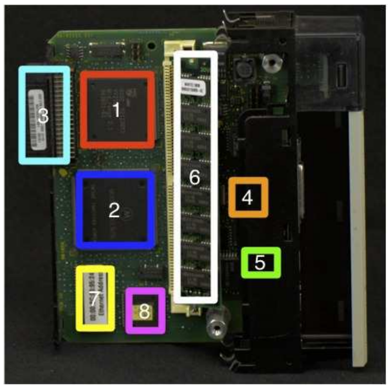

Figure 5. Allen-Bradley EtherNet/IP module (annotations in Table 2).

4. Firmware Analysis

The analysis of PLC device firmware differs from the typical analysis of software (binaries). In general, the toolset for personal computers is not well suited to firmware analysis. Additionally, it may require considerable effort to merely identify the processor and architecture of an embedded device.

The main challenges in understanding embedded devices stem from the diversity of components. Each device uses a different combination of processor architecture, embedded operating system, board design, backplane connector, protocol and logic representation. Many vendors use custom-built components, such as a system-on-a-chip (SOC) main processor, that do not have openly available documentation. The analysis is more difficult because SOCs and custom components are potentially unique, poorly documented and few analysts have experience with them. Unlike "normal" software analysis, the analysis of firmware can depend on chip-specific features. In the case of firmware, vulner-

Table 1. Allen-Bradley Logix 5555 processor module part list (see Figure 4).

	Part Name	**Manufacturer Part Number**	**Markings**
1	Backplane Communications Processor	NXP (Philips Semiconductors) VY21422E2 (Customer-specific product; Discontinued 31 Dec 2005)	PHILIPS ARM VY21422E Y43729Y1 03 KP0250 E MIDRANGE P3E 943631-64
2	Main Processor	NXP (Philips Semiconductors) VY21754A2 (Customer-specific product; Discontinued 31 Dec 2005)	PHILIPS ARM VY21754A Y35737Y1 08 KPr0224 A ARGUS-R2.1 943881-71
3	Backplane Connector		
4	1M High Speed SRAM	Hitachi HM621864HB	JAPAN 0133 HM621864HBLJP-20 00007NN0
5	4 Mb Single Supply Flash Memory	STmicroelectronics M29F040B	M29F040B 45K1 585200210 SINGAPORE
6	Y2K-Compliant Watchdog Real-Time Clock	Dallas Semiconductor DS1501 (May also be branded MAXIM)	DALLAS DS1501YEN 0247A6 045AM
7	32 MB CMOS 5.0V only, Uniform Sector Flash Memory	AMD AM29F032B (Made by Spansion)	AM29F032B -90EI 0113DPB H ©1998 AMD
8	Additional Flash Memory		
9	Riser to Memory Board		

abilities in startup and interrupts are just as interesting as vulnerabilities in network code and application logic.

Table 2. Allen-Bradley Logix 5555 EtherNet/IP module part list (see Figure 5).

	Part Name	**Manufacturer Part Number**	**Markings**
1	Backplane Communications Processor	NXP (Philips Semiconductors) VY21086-2 (Customer-specific product; Discontinued 31 Dec 2005)	PHILIPS ARM 0102 y21230Y1 VY21086-Mid_range 2.1 943361-62
2	MC68360 QUad Integrated Communication Controller (QUICC)	Freescale Semiconductor MC68360 (CPU32+ architecture, 25 MHz)	MC68EN360CEM25L OK36E IQAC0049 KOREA
3	Backplane Connector		
4	Enhanced Ethernet Transceiver	Freescale Semiconductor MC68160A	MC68160 AFB HGR0045
5	5V Byte Alterable EEPROM	Xicor X28HC64	XICOR X28HC64JI-90 Cy0047
6	1M x 4-bit CMOS DRAM	Hyundai HY514400A (Eight total)	HY514400A LJ-60 9751C KOREA
7	5V FlashFile Memory	Intel 28F008SA (Sticker covering this chip shows Ethernet MAC address)	PA28F008SA 85 U0200321W (M)(C) '92 '96 Flash
8	High-Density EE CMOS Programmable Logic	Lattice Semiconductor MACH210A	Lattice MACH210A-10JE -12JI B023PE2

Analysis can proceed quite rapidly when the processor architecture is easy to determine. However, when the processor/operating system combinations are difficult to discern, the initial identification step in device analysis can take a significant amount of effort. To simplify the task, we developed a process for analyzing PLC device firmware that focuses on obtaining firmware images

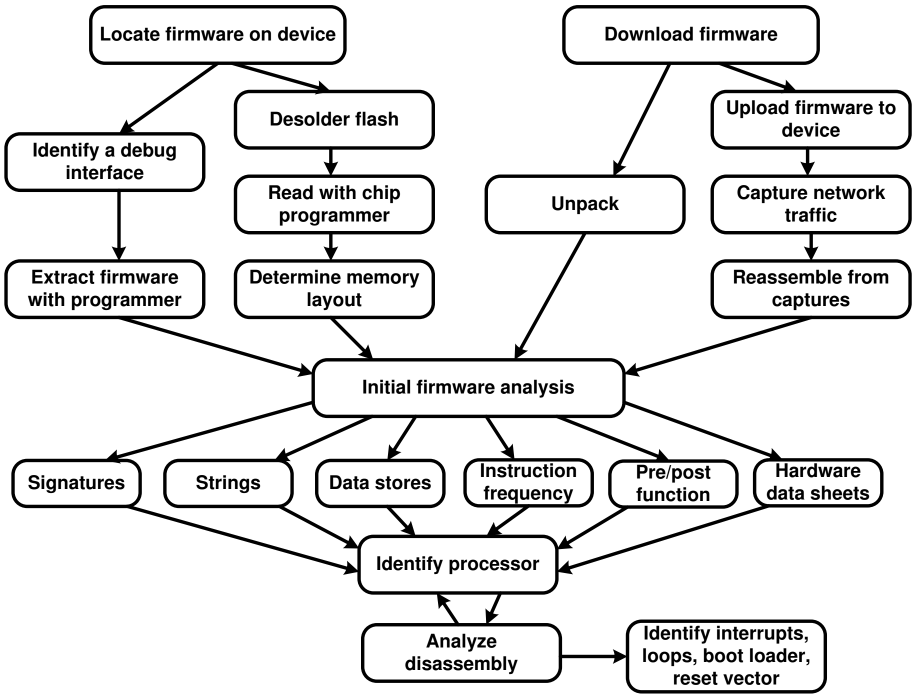

Figure 6. Process for analyzing embedded device firmware.

and identifying the processor architecture. The firmware analysis process is outlined in Figure 6.

4.1 Acquiring Firmware

We investigated four methods for acquiring firmware:

- Read directly from the device using a debug port.
- Read directly from flash memory using a chip programmer.
- Unpack firmware update files.
- Capture network traffic during a firmware update.

Each method for obtaining firmware has inherent difficulties. Connecting via debug ports (e.g., JTAG) works for some devices; however, it is often the case that the interfaces are disabled or non-standard protocols are used. Reading firmware from flash memory consistently worked to obtain the firmware, but unusual memory layouts can make reassembling the entire firmware extremely difficult.

Vendor firmware updates were often not available for the devices of interest to allow the unpacking of update files. In situations where the updates were available, the firmware contents had to be extracted and reconstructed. Additionally, firmware obtained by capturing network traffic from an update or backup mechanism had to be extracted from the data sections of multiple network packets. Whether they are captured on disk or reconstructed from network captures, the firmware updates obtained from vendor websites often did not represent the actual layout and configuration of the firmware on the devices.

4.2 Identifying the Device Architecture

Identifying the processor and memory architecture of a device was one of the more challenging tasks. In some cases, processors can be identified by their chip markings. However, even when a processor is identified, the memory architectures can vary considerably, especially the flash memory located on the microcontroller, the separate flash and memory boards with a field-programmable gate array (FPGA) interface, and the traceable direct connections between microprocessor and memory. Depending on the design or manufacturing date of a device, it may be possible to guess the likely processors based on their popularity at the time, the relationships between companies, and the processors used in similar devices.

In some cases, we identified the processor type by comparing byte patterns to function calls, register usage and instruction frequency for a likely processor. For example, at the beginning of a function call there is often a store instruction to preserve non-volatile registers and a load instruction is often present at the end of a function. After a processor is tentatively identified, the result can be verified by comparing the firmware code with other code for the processor. The start of function calls, register usage and instruction frequency often facilitate this verification.

5. Communication Backplane Analysis

Analyzing the firmware for every PLC or field device would be extremely time-consuming due to the wide range of hardware and firmware used even within a single product line. However, network protocols are common to a wide range of PLCs. In fact, we discovered that modular PLCs appear to use variants of common network protocols for backplane communications between modules. This means that the backplane presents an avenue for analyzing a wide range of PLCs. Note, however, that backplane analysis is not a replacement for firmware analysis; rather, the two approaches are complementary.

5.1 Identifying Physical Properties

Identifying the pin spacing and layout provides the initial structure of the backplane and allows an analyst to create custom connectors for data collection. A continuity tester and voltmeter can be used to identify various signals on the PLC backplane, which provides insight into the likely locations of ground pins and some power pins. It is necessary to identify the voltage on each pin to avoid damaging the logic analyzer and to further narrow the pins of interest. Additionally, a voltmeter can be used to identify the operating voltage of each pin on the backplane through startup and normal operation of the PLC.

5.2 Analyzing Pin Logic

After identifying the backplane pins and physical properties, a logic analyzer may be used for deeper analysis. The logic threshold and sampling rate are required to obtain clean captures of normal backplane traffic. These are determined through trial and error. Captures from the logic analyzer may be reviewed to form hypotheses about the use of each pin. Some possible signals of interest include clock-enable, end-of-frame, frame-header, clock and data. The packet timing is first determined by measuring the length of packets and gaps between packets. The backplane traffic is then monitored under several different hardware configurations by removing and reordering modules in the PLC. From these captures, it is possible to identify the modules that created the various packets.

5.3 Translating Backplane Traffic into Bytes

After the data signals have been identified, the analysis of the transmitted data can begin. Logic analyzer software can be used to export the backplane traffic captures to the comma-separated value (CSV) format. The presence of a signal (i.e., voltage above the determined threshold) is represented as a binary one and the absence (i.e., voltage below the threshold) is represented as a binary zero. A simple script can be used to parse the CSV file, translate the binary signals into bytes and identify the header and data sections. If the backplane sends bytes in parallel, it is necessary to identify the order of data pins. This is accomplished by testing different combinations and searching for data bytes that match known patterns (e.g., ASCII, low digits or network protocol headers).

5.4 Analyzing the Backplane Protocol

In order to understand the parser byte output, it is necessary to analyze the backplane protocol. The first step is to determine if the protocol is openly documented. The protocol specification identifies the required fields and unique field values, which can help categorize packets. If software implementations of the protocol are available, they can be used to determine the structure of conversations and packets. Additionally, a comparison between network packet captures and backplane traffic captures can provide insight into the use of some fields; this is especially useful when the captures were taken during the same time period. In our experience, PLC backplanes are often based on network protocols used by PLCs to communicate with external entities.

5.5 Dissecting Backplane Traffic

The final step in backplane analysis is to dissect the backplane traffic. A protocol traffic dissector (e.g., Wireshark) can automatically identify some fields in request and response packets. However, most traffic dissectors do not handle backplane protocols. Therefore, it may be necessary to develop a custom dissector program for analysis.

We have developed a dissector program that consists of a pre-processing script and a protocol field identifier. The pre-processing script reads in binary dumps, identifies packet boundaries, removes collection timestamps and writes ASCII-encoded hex values. The protocol field identifier reads in ASCII-encoded hex values, removes stray line noise, identifies known headers and payloads, and prints a human-readable summary of the packets.

6. Conclusions

Attack techniques that target PLCs will continue to grow in sophistication. As security mechanisms are developed to protect the application layer, attackers will begin to exploit lower levels of abstraction. It is imperative that the security community prepare for this threat and develop automated tools and techniques. The techniques described in this paper can be leveraged to develop automated tools for performing dynamic analyses of PLCs used in the critical infrastructure.

Acknowledgements

This research was supported by the Laboratory Directed Research and Development (LDRD) Program at Sandia National Laboratories, Albuquerque, New Mexico. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy National Nuclear Security Administration under Contract No. DE-AC04-94AL85000.

References

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

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

[3] Industrial Control System Cyber Emergency Response Team (ICS-CERT), Schneider Electric Quantum Ethernet Module Multiple Vulnerabilities, ICS-CERT Alert 11-346-01, Department of Homeland Security, Washington, DC, 2011.

[4] H. Moore, Shiny old VxWorks vulnerabilities, Metasploit (`community.rapid7.com/community/solutions/metasploit/blog/2010/08/p2/shiny-old-vxworks-vulnerabilities`), August 2, 2010.

[5] D. Peck, Security testing, vulnerabilities and exploits in operating systems used in control system field devices, *Proceedings of the SCADA Security Scientific Symposium*, 2010.

[6] D. Peck and D. Peterson, Leveraging Ethernet card vulnerabilities in field devices, *Proceedings of the SCADA Security Scientific Symposium*, 2009.

[7] D. Peterson, Project Basecamp at S4, Digital Bond Blog, Digital Bond, Sunrise, Florida (`www.digitalbond.com/2012/01/19/project-basecamp-at-s4`), January 19, 2012.

[8] R. Santamarta, Reversing industrial firmware for fun and backdoors, Reversemode (`reversemode.com/index.php?option=com_content&task=view&id=80&Itemid=1`), December 12, 2011.