Programmable Logic Controller Forensics

Irfan Ahmed | University of New Orleans
Sebastian Obermeier | ABB
Sneha Sudhakaran and Vassil Roussev | University of New Orleans

Programmable logic controllers (PLCs) automate the control and monitoring of physical industrial and infrastructure processes such as power generation, gas pipelines, and water management. Due to the convergence of networking infrastructure, PLCs can be exposed to cyberattacks over the network with potentially catastrophic consequences. This article introduces the basic mechanisms by which various attacks can be detected, analyzed, and ultimately remedied.

Early industrial control system (ICS) environments were isolated deployments, not connected to other networks (such as the Internet or the corporate intranet), and their cybersecurity wasn't a concern. However, over the past two decades, these systems have become tightly integrated using commercial-off-the-shelf software and hardware, which brings substantial economic advantages. The unintended consequence of this network convergence is that IT vulnerabilities can ultimately provide attackers with direct access to ICS networks that are open to manipulation and attack. If attackers compromise a programmable logic controller (PLC), they can disrupt the normal control of a physical process and cause a catastrophe, such as environmental damage and loss of life. The recent cyberattack on the Ukrainian power grid shows such attacks are possible.[1]

Digital forensic investigation after a security breach or catastrophic event is crucial to answer questions about a cyberattack such as[2]

- What components were affected in the ICS environment?
- How was the attack executed? Who were the perpetrators, and what was their location?
- Were the attackers internal or external, and what was their motivation?
- Is the attack still active, and can it be repeated?

In this article, we focus on several common attack scenarios and discuss artifacts that a forensic investigation of an affected PLC can recover to help diagnose the attack. We summarize the available tools and methods for PLCs, both at the network and device level, that allow the extraction and analysis of the forensic artifacts.

PLC Primer

To understand cyberattacks on PLCs, it's essential to understand the PLC's role in an ICS environment, its interaction with other ICS components, and its hardware architecture at the device level.

PLC in an ICS Environment

Figure 1 presents an overview of a typical ICS environment. It consists of two major sections: control center

 Copublished by the IEEE Computer and Reliability Societies

and field sites. The control center runs ICS services such as the human–machine interface (HMI), engineering workstation, historian (a database application), and control server. The field site has sensors, actuators, and PLCs that are installed locally to monitor and control the physical processes. For instance, in a gas pipeline, a PLC monitors and controls the gas pressure. It obtains the current pressure of the compressed gas in the pipe. If the pressure exceeds a certain threshold, it opens a solenoid valve (an actuator) to release some gas, which reduces the gas pressure in the pipe.

An engineering workstation is used to configure and program a PLC. It has PLC vendor-specific programming software to write control logic that defines how the PLC should control a physical process. The logic is written in one of the languages defined in IEC 61131-3, such as ladder logic or instruction list. In the gas pipeline example, a PLC is programmed to maintain pressure in the pipeline between 40 and 50 PSI. Based on readings from the pressure sensor, if the gas pressure is more than 50 PSI, the PLC opens the solenoid valve to release some gas until the pressure is reduced to 40 PSI.

The control server or master terminal unit (MTU) and PLCs communicate with one another using ICS protocols, such as Modbus, PROFINET, and EtherNet/IP, and exchange data about the current status of a physical process, such as gas pressure and the solenoid valve's on/off state. When the data arrives at the control center, the HMI interprets and presents the data in a graphical user interface to a human operator. The HMI enables the operator to monitor the process remotely and make operational decisions to maintain safety and efficiency. For instance, the operator might choose to turn on the solenoid valve to release some gas even when the pressure is less than the threshold value. The historian stores the PLC data, which is used for viewing trends in graphical form for analysis.

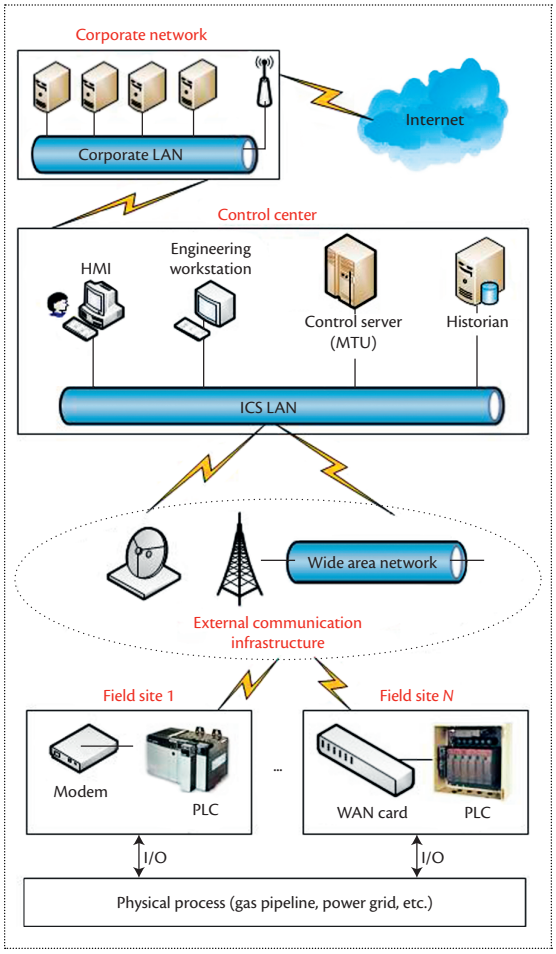

Figure 1. Overview of an industrial control system (ICS) environment. It consists of two major sections: control center and field sites. The control center runs ICS services such as the human–machine interface (HMI), engineering workstation, historian, and control server. The field site has sensors, actuators, and programmable logic controllers (PLCs) that are installed locally to monitor and control the physical processes.

PLC Architecture

Figure 2 depicts a typical architecture of a PLC that includes input and output modules, power supply, and memory such as RAM and EEPROM. The nonvolatile memory (EEPROM) stores firmware or an OS and control logic program (also called ladder logic). Input and output devices such as sensors, switches, relays, and valves are connected with the input and output modules. The PLC is connected with a physical process; input devices provide the current state of the process to the PLC, which the PLC processes through its control logic, and manipulate the physical process accordingly via output devices. In a ladder logic program, the input and output devices are referred to as *contacts* and *coils*.

Cyberattacks on PLC

PLCs' primary design requirements are safety, real-time response to changes in the monitored processes, and the ability to work in environments. They were never designed for resilience against network attacks of any kind. In a typical ICS scenario, the PLCs are connected to the control center and the center is connected to the corporate network, which is usually connected to the Internet through a demilitarized zone (DMZ). This connectivity chain exposes the PLCs to adversaries who can remotely launch pivot attacks to reach PLCs and manipulate their normal functions.

An ICS cyberattack involves communication protocols such as Modbus and DNP3, and exploits either the protocol specification or the vendor-specific

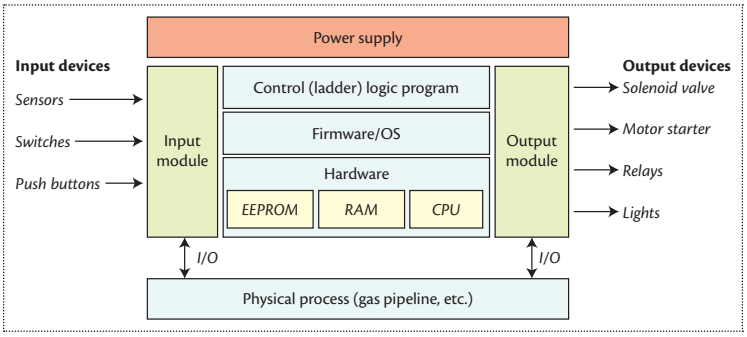

Figure 2. A typical PLC architecture. It includes input and output modules, power supply, and memory such as RAM and EEPROM (nonvolatile memory).

implementation of the protocol. In a typical scenario, the attacker penetrates an ICS network and interrupts, intercepts, modifies, and/or fabricates messages to manipulate PLCs and the network's ICS services.[3] For instance, the Idaho National Laboratory (INL) demonstrated a remote cyberattack on field devices (controllers) at the 2004 KEMA Control Systems Cyber Security Conference.[4] The attack was initiated at the Sandia National Laboratory and carried out remotely at INL. It exploited a newly discovered buffer overflow vulnerability in the Apache software at the control center, took control of field devices, and flipped the current state of a breaker.

The forensic artifacts of such attacks can exist at both network and device levels.

Network-Level Analysis

The common categories of PLC cyberattacks at the network level include reconnaissance, man-in-the-middle (MITM), and denial-of-service (DoS) attacks.[5–7]

Reconnaissance. Reconnaissance is the preliminary step of information gathering that precedes the actual attack. It involves the identification of supporting function codes, assigned PLC addresses, make, model, firmware, and so on.[7] It can be achieved by either passively eavesdropping or actively querying the PLC. Passive reconnaissance is hard to investigate because it rarely leaves any traces on the network, whereas active querying does.

To identify assigned PLC addresses, attackers can scan the whole address range for the PLCs by sending them the request messages with a commonly supported function code. If a response is received for an address, it exposes the address used by a PLC. This approach is quite noisy as it leaves a significant number of messages containing unknown PLC addresses in the network traffic log. For example, a full address scan using the Modbus protocol requires 247 messages—the number of allowable addresses.

Once attackers know a PLC's address, they can perform a similar attack to explore the supporting function codes in the PLC. Attackers send messages with all possible function codes to the PLC and, based on the response, can determine the valid function codes. For instance, a Modbus message has a 1-byte field for function code. If a PLC doesn't support a function code, it sends an exception response message with exception code 0x01, indicating an illegal function code. Just like the previous attack, this one also leaves a lot of forensic artifacts in the network traffic in the form of exception response messages. To reduce the network footprint, attackers might attempt to explore a small number of function codes that are essential to launch an attack. For example, code 0x05 in Modbus is used to alter a coil value, which typically directly controls a physical process.

If an attack involves exploiting a specific PLC vulnerability, the reconnaissance will aim to gather the PLCs' specific make, model, and software versions to identify a vulnerable one. The protocols might support functions for extracting device-specific information. In Modbus, function code 0x11 reveals the device's current run status and additional device-specific information. The presence of such traffic in the network log is a useful indicator of reconnaissance.

Man in the middle. In an MITM attack, an attacker is positioned as a mediator of all communication between ICS services in a control center and PLCs in the field sites. In other words, all the traffic passes through the attacker's machine, allowing him or her to eavesdrop and to manipulate message content at will. For instance, in a gas pipeline, the attacker could fabricate a message that instructs the PLC to open the solenoid valve. When the PLC sends the current state of the valve to the control center, the attacker could modify the data in the message to represent the state of the valve as closed. When the HMI receives the manipulated data, it shows the human operator that the valve is in the closed position, which is the normal state of the valve. Thus, an MITM attack provides a measure of stealth that can keep the victim in the dark indefinitely.

One approach to place the attacker in the MITM position is Address Resolution Protocol (ARP) spoofing. The attacker modifies the ARP tables (used for IP/MAC resolution) of the victim's PLC and the computer running the HMI to associate their IP addresses with the attacker's MAC address. After modification, both the PLC and the computer start using the attacker's MAC address in the messages exchanged with each other, redirecting the messages to the attacker's machine.[5]

Assuming that a forensic investigator has the correct IP/MAC association, these types of attacks are identifiable by analyzing a network traffic log for inconsistent

IP/MAC associations. Another source of data is the PLC's RAM content and the computer from which the ARP tables can be extracted.

Denial of service. A typical PLC informs the control center periodically about the current state of the physical process being monitored and controlled. A DoS attack on a PLC is designed to interfere with its normal processing by denying it the ability to either communicate with other ICS components or execute control logic programs. A typical DoS attack on communication involves packet flooding, which exhausts the bandwidth of the communication link between the PLC and the control center.[5]

Device-Level Analysis

The common categories of attacks on the PLC device include DoS, command injection, and memory corruption.[7–10]

Denial of service. A DoS attack on a PLC device either targets a vulnerability in a PLC component, such as firmware, or exploits a normal function in a malicious manner. For instance, a malformed packet, if not properly handled by a PLC, could cause the PLC to crash, preventing it from executing the control logic.

Another attack involves altering a PLC operational mode from RUN to PROGRAM, which stops the PLC execution of the logic program. The PLC normally operates in RUN mode; PROGRAM mode is used to transfer new ladder logic to the PLC. The mode switch is needed to facilitate the normal update of the control logic, but it could also be misused to attack the system. Detection of such malicious activity is readily possible by examining the patterns of communication sent to the PLC; the investigator needs an appropriate tool to automate this process.

Command injection. Command injection attacks are performed by injecting unwanted code into a computer system, thereby gaining unauthorized control over the system. These attacks are normally executed in PLCs by overwriting the ladder logic program. Thomas H. Morris and Wei Gao identify three types of command injection attacks: malicious state injection, malicious parameter injection, and malicious function code injection.[7]

The malicious state injection attack sends a malicious command to a PLC to transition the state of a physical process to an abnormal state. For instance, changing the coil state from 0x0 to 0x1 alters the state of the corresponding actuator.

The malicious parameter injection attack changes a device set point. For instance, the threshold setting of the gas pressure is changed from 50 to 300 psi. The pipe might be damaged if it can't bear 300 psi.

The malicious function code injection attack changes the configuration of the PLC to an unexpected state such as restart of communication.

Memory/firmware corruption. For more than a decade, memory corruption attacks such as buffer overflow have been common in ICS networks and PLCs.[8]

Luis Garcia and his colleagues presented Harvey, a rootkit that infects a PLC's firmware to control all inputs and outputs.[9] Harvey resides at the firmware level, intercepts the firmware's control and information flow, and modifies the input and output data arbitrarily.

PLC Forensic Challenges

Several challenges must be overcome to investigate a suspect PLC efficiently and effectively. These include the lack of network forensic tools, the PLC's resource constraints and potentially remote location, and the proprietary nature of most hardware and software used to program and maintain it.

Local Access to a PLC

A PLC can be located at a remote region that's effectively unreachable for a forensic investigator. If a PLC is compromised and local access to a PLC is difficult, the investigation tools and techniques that require physical access to the PLC won't work.

Resource-Constrained PLC Device

PLCs are designed to run continuously for control and monitoring and are resource constrained with limited computing power and memory storage. Thus, the use of a forensic tool must not interfere with a PLC's availability, which is challenging for an investigator who wants to run such forensic tools in an unfamiliar ICS environment.

PLC Data Acquisition

A PLC is connected to I/O devices such as sensors and actuators. The amount of data generated by these devices can be quite significant. For instance, in power grid stations, sensors can carry out measurements up to 4,000 times per second. It's challenging for an investigator to capture and analyze such a large amount of data.

PLC components include RAM and EEPROM, which store the firmware. To identify any malicious modifications in the firmware, its acquisition is inevitable. The current techniques require local access to a PLC and utilize JTAG (Joint Test Action Group)/UART (universal asynchronous receiver–transmitter) type interfaces for EEPROM/RAM data acquisition. RAM content contains the PLC's current state and artifacts

Table 1. Classification of unknown file types based on content.

File type	Classification
0x4C	Simple Mail-Transfer Protocol configuration
0x49	Ethernet configuration
0x22	Ladder logic—control logic program
0x4D	DNP3 configuration
0x95	Routing information
0x47	DF1 (asynchronous byte-oriented protocol) configuration

of forensic relevance including the current ladder logic program, ARP table, and so on. Unfortunately, no forensic techniques are available to remotely acquire the firmware and RAM contents over the network.

Proprietary Closed Source Firmware

PLCs generally run closed source, vendor-specific, proprietary firmware, making it difficult to develop generalized forensic solutions for PLCs or to modify the firmware to add functionalities for PLC forensic readiness.

Inadequate ICS Network Forensic Tools

Remote attacks on PLCs involve the ICS network. A PLC might support different ICS protocols depending on the physical process. For instance, DNP3 is popular in the power sector. Unfortunately, network forensic tools aren't available to support a large number of ICS protocols, analyze network traffic logs to identify different types of cyberattacks especially on PLCs, and extract forensic artifacts. For instance, if an adversary transfers a ladder logic program in a PLC, a network forensic tool is required to extract and analyze the program from a network traffic log.

Insufficient Logging

Logging is critical for the re-creation and correlation of events. Unfortunately, PLCs have limited or no logging capabilities. The logging is geared toward process disturbance, not security breaches, and thus isn't useful for forensic investigation. The logs exist for a limited time and are overwritten frequently because PLC memory is small. In the control center, the historian stores the values of process and control variables received from a PLC. However, historian data is limited and depends mostly on MTU configurations. The historian doesn't store the current state of a PLC's ladder logic and firmware.

PLC Forensic Tools and Methodologies

PLC forensics is mostly an unexplored territory, currently limited to ICS network protocols.

Network Level

We recently developed a network forensic tool to analyze the network traffic log of an ICS protocol, PCCC (Programmable Controller Communication Commands).[11] The prototype tool can extract the low-level representation of a control ladder logic program, PLC system configurations such as Simple Mail-Transfer Protocol (SMTP) client, and the state of input and output devices. The protocol contains file type and file number fields for the function codes 0xA2 and 0xAA for reading and writing to a PLC, respectively. Interestingly, network traffic analysis reveals several unknown file types, which we classified based on their content. Table 1 presents the file types along with the description of their content. SMTP client information such as username, password, and to/from email addresses is transferred in ASCII characters over the network.

Ken Yau and Kam-Pui Chow proposed a Control Program Logic Change Detector, which derives a set of detection rules using a ladder logic program.[12] The rules are then utilized to analyze network traffic of the PLC running the ladder logic program and identify any unexpected (potentially anomalous) control variable values.

Tim Kilpatrick and his colleagues presented a network forensics architecture in which network packets at certain strategic locations were captured and followed by reconstructing network events from the captured packets. The architecture has two major components—agents and the data warehouse.[13] The agents capture packets from network traffic from specific locations in an ICS network. The architecture also has a synopsis engine to create a summary for each packet. The packet is then passed to the data warehouse. The data warehouse digitally signs the summary of each packet and forwards it to appropriate agents. The digitally verified summary is later used as evidence for network forensic capabilities.

Craig Valli proposed using the Snort intrusion detection system (IDS) for the forensic investigation of Modbus and DNP3 vulnerabilities and attacks.[14] He created a step-by-step procedure for forensic analysis including creating environment scanning for anomaly detection as a first step. Next, production and replay for each vulnerability is performed, then each vulnerability is analyzed, resulting in the creation of an IDS rule set. Finally, testing of each rule set is performed.

Amit Kleinmann and Avishai Wool analyzed the network traffic of Siemens S7 PLC and observed that the traffic is highly periodic.[15] Based on the observation, they proposed an IDS that utilizes a deterministic finite automaton to model the network traffic accurately. The IDS achieves 99.26 percent accuracy.

Device Level

A forensic investigation requires the acquisition and analysis of data from suspicious PLCs.

Data acquisition. For live data acquisition, both volatile and nonvolatile data should be acquired. Although ICS protocols have function codes to read specific data types, such as input, output, counter, and timer, from a live PLC, these function codes can't be used to acquire the entire RAM and EEPROM content.

We analyzed two different PLCs (referred to as PLC A and PLC B) from a major vendor regarding their forensic capability. (The make and model of the PLCs are anonymized to protect proprietary information.) The main focus is on evaluating the possibility to easily acquire RAM and nonvolatile RAM (NVRAM) content, then dump and investigate the file system.

Table 2 summarizes the evaluation results. A positive smiley means it's easy to extract the type of information; that is, the device supports the forensic analyst by providing extraction functionality.

A challenge in analyzing PLCs is extracting memory content. Whereas PLC B at least dumps the dynamic RAM (DRAM) to a flash card if it crashes (mainly for debugging purposes), PLC A doesn't offer any support. However, neither device supported a DRAM dump while in operation.

There's little support for extracting PLC A's NVRAM (PLC B had none). Apparently, the only possibility is to extract parts of the NVRAM through the PLC's real-time OS.

PLC A contains a compact flash card that stores the entire file system. The card can be removed by opening the device, and also contains the boot image in the boot sector of the card and the entire configuration information. PLC B's built-in flash storage contains only the firmware; the actual applications running on the controller exist only in RAM. In the event of a power failure, a backup battery keeps the RAM contents alive for a certain period of time. The compact flash slot can be used to upgrade a controller's firmware or load applications onto the controller; or if a compact flash card is inserted into the controller when it crashes, a dump of the RAM of the controller will be written to the compact flash card.

The difficulty in obtaining the PLC databases is connected to the difficulty of acquiring the flash storage. While PLC A exposes the complete device database, PLC B stores all information in its RAM, which, as pointed out, is very difficult to obtain.

Log files can be easily extracted for both PLCs, and network statistics can also be extracted using the PLC configuration tool.

In summary, both investigated devices have a different design and architecture and use distinct tools for configuration and management. This results in differences in the capabilities to extract specific information, data artifacts, and memory content. In general, we recommend that device manufacturers consider forensic capabilities early in their device life cycle.

Table 2. Evaluation of two programmable logic controllers' (PLCs') forensic capabilities.

Artifact	PLC A	PLC B
Dynamic RAM	☹	😐
Nonvolatile RAM	😐	–
Flash storage	☺	😐
Databases	☺	☹
Log files	☺	☺
Network statistics	☺	☺

Data Analysis

Saman A. Zonouz and his colleagues proposed combining symbolic execution and model checking to analyze the code bounds of malicious injected PLC code.[10] Symbolic execution of PLC program code analyzes the causal relationship between different inputs and the resulting execution of program code. Model checking validates the PLC code of the intended control system against the specification of a model ICS. This combined approach can be employed for forensic purposes to identify which areas have caused PLC malicious code injection trials, and which part of the code was the likely cause for such execution. Thus, model checking can indicate that aberrant behavior has occurred, whereas symbolic execution can point to the root cause of the execution.

Lucille McMinn and Jonathan Butts proposed a firmware verification tool for forensically analyzing the trials of altering firmware code to gain access over the ICS network by any unauthorized person. This is done by either comparing the firmware under test with a good firmware version or analyzing the data captured from the PLC and identifying whether the underlying PLC's firmware is modified.[16]

Industrial control systems support critical infrastructure services and have life cycles that span decades. This introduces challenges that could not have been foreseen and need to be addressed by other means. PLCs, which execute an ICS's control logic, are particularly vulnerable to network attacks; therefore,

developing the means to quickly diagnose and analyze such attacks is critical.

Unlike the forensic analysis of a conventional IT system, which has seen more than three decades of active development, PLC forensics is still at an early stage of development. This is partially due to the specialized nature of ICS and the prevalence of proprietary and poorly documented protocols. Nonetheless, over the past decade, researchers and practitioners have adapted several core methods and tools from IT forensics and security monitoring to the ICS world. In particular, reverse engineering has resulted in the public documentation of several ICS protocols, network analysis tools have been adapted to parse and interpret such protocols, and an emerging body of work is targeting the direct analysis of PLC memory content. ■

References

1. R.M. Lee, M.J. Assante, and T. Conway, *Analysis of the Cyber Attack on the Ukrainian Power Grid*, tech. report, SANS Inst., Washington, DC, 2016.
2. I. Ahmed et al., "SCADA Systems: Challenges for Forensic Investigators," *Computer*, vol. 45, no. 12, 2012, pp. 44–51.
3. S. East, "A Taxonomy of Attacks on the DNP3 Protocol, *Critical Infrastructure Protection, IFIP Advances in Information and Communication Technology*, vol. 311, Springer, 2009, pp. 67–81.
4. J. Weiss, *Protecting Industrial Control Systems from Electronic Threats*, Momentum Press, 2010.
5. B. Chen, "Implementing Attacks for Modbus/TCP Protocol in a Real-Time Cyber Physical System Test Bed," *IEEE Int'l Workshop Technical Committee on Communications Quality and Reliability* (CQR 15), 2015, pp. 1–6.
6. P. Huitsing et al., "Attack Taxonomies for the Modbus Protocols," *Int'l J. Critical Infrastructure Protection*, vol. 1, Dec. 2008, pp. 37–44.
7. T.H. Morris and W. Gao, "Industrial Control System Cyber Attacks," *Proc. 1st Int'l Symp. ICS and SCADA Cyber Security Research* (ICS-CSR 13), 2013, pp. 22–29.
8. C. Bellettini and J.L. Rrushi, "Combating Memory Corruption Attacks on SCADA Devices," *Proc. 2nd. Ann. IFIP Int'l Conf. Critical Infrastructure Protection* (IFIP 08), revised papers, 2008, pp. 141–156.
9. L. Garcia et al., "Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit," *Proc. 24th Ann. Network and Distributed System Security Symp.* (NDSS 17), 2017.
10. S.A. Zonouz et al., "Detecting Industrial Control Malware Using Automated PLC Code Analytics," *IEEE Security & Privacy*, vol. 12, no. 6, 2014, pp. 40–47.
11. S. Senthivel, I. Ahmed, and V. Roussev, "SCADA Network Forensics of the PCCC Protocol," *Proc. 7th Ann. Digital Forensics Research Conf.* (DFRWS 17), 2017, pp. S57–S65.
12. K. Yau and K.-P. Chow, "PLC Forensics Based on Control Program Logic Change Detection Works," *J. Digital Forensics, Security and Law*, vol. 10, no. 4, 2015, pp. 59–68.
13. T. Kilpatrick et al. "Forensic Analysis of SCADA Systems and Networks," *Int'l J. Security and Networks*, vol. 3, no. 2, 2008, pp. 95–102.
14. C. Valli, "Snort IDS for SCADA Networks," *Proc. 2009 Int'l Conf. Security and Management* (SAM 09), 2009, pp. 618–621.
15. A. Kleinmann and A. Wool, "Accurate Modeling of the Siemens S7 SCADA Protocol for Intrusion Detection and Digital Forensics," *J. Digital Forensics, Security and Law*, vol. 9, no. 2, 2014, pp. 37–50.
16. L. McMinn and J. Butts, "A Firmware Verification Tool for Programmable Logic Controllers," *Critical Infrastructure Protection*, of *IFIP Advances in Information and Communication Technology*, vol. 390, Springer, 2012, pp. 59–69.

Irfan Ahmed is an assistant professor of computer science at the University of New Orleans. His research interests include industrial control system security, digital forensics, and malware detection and analysis. Ahmed received a PhD in computer science from Ajou University. He's an associate member of the American Academy of Forensic Sciences. Contact him at irfan@cs.uno.edu.

Sebastian Obermeier is the Group Research Area Manager Software at ABB. His research interests include IT security for industrial control systems and database technology. Obermeier received a PhD in computer science from the University of Paderborn. Contact him at sebastian.obermeier@ch.abb.com.

Sneha Sudhakaran is a PhD student in the Department of Computer Science at the University of New Orleans. Her research interests include industrial control system security and forensics. Contact her at ssudhaka@my.uno.edu.

Vassil Roussev is a professor in the Department of Computer Science at the University of New Orleans. His research interests include network and mobile security, digital forensics, big data security, privacy, and usable security. Roussev received a PhD in computer science from the University of North Carolina at Chapel Hill. Contact him at vassil@cs.uno.edu.