Detecting PLC Control Corruption via On-Device Runtime Verification

Luis Garcia, Saman Zonouz
Department of Electrical & Computer Engineering
Rutgers University
Piscataway, New Jersey 08854
{*l.garcia2, saman.zonouz*}*@rutgers.edu*

Dong Wei, Leandro Pfleger de Aguiar
Siemens Corporation, Corporate Technology
Princeton, New Jersey, 08540
{*dong.w, leandro.pfleger*}*@siemens.com*

***Abstract*—With an increased emphasis on the cyber-physical security of safety-critical industrial control systems, programmable logic controllers have been targeted by both security researchers and attackers as critical assets. Security and verification solutions have been proposed and/or implemented either externally or with limited computational power. Online verification or intrusion detection solutions are typically difficult to implement within the control logic of the programmable logic controller due to the strict timing requirements and limited resources. Recently, there has been an increased advancement in open controller systems where programmable logic controllers are coupled with embedded hypervisors running operating systems with much more computational power. Development environments are provided that allow developers to directly integrate library function calls from the embedded hypervisor into the program scan cycle of the programmable logic controller. In this paper, we leverage these coupled environments to implement online cyber-physical verification solutions directly integrated into the program scan cycle as well as online intrusion detection systems within the embedded hypervisor. This novel approach allows advanced security and verification solutions to be directly enforced from within the programmable logic controller program scan cycle. We evaluate the proposed solutions on a commercial-off-the-shelf Siemens product.**

I. Introduction

The security of Programmable Logic Controllers (PLCs) is increasingly becoming a vital issue in securing industrial control systems (ICS). There is an inherent difficulty integrating security into these PLCs as they are intended to be simple computing machines whose programs can be easily verified with the underlying physical systems they are controlling. Adding advanced security tools can compromise the time-sensitive operations as well as any general temporal attributes of the cyber-physical system.

The security of PLCs continues to receive an increased amount of attention in the wake of ICS-targeted malware. ICS-CERT reports that in FY 2015 [1], they responded to 295 reported incidents involving critical infrastructure in the United States. Most programming and operator commands are sent using insecure proprietary network protocols. Not only have proprietary protocols been reverse engineered, but open-source API's [2] have been released that allow programmers to develop invasive tools that can be used with malicious intent, such as PLCInject [3]. Additionally, open source packet dissectors have been developed for network protocol analyzers.

The reverse engineering of certain proprietary protocols has resulted in new protocols being developed with encrypted communication. Although these protocols can provide secure communication for the latest products, they are typically only supported by later devices while the legacy devices remain vulnerable to packet injection attacks.

Offline security solutions such as TSV [4] and [5] have been proposed as bump-in-the-wire verification mechanisms sitting between the operator/programmer interface and the PLC. These solutions have provided the ability to verify the programs downloaded to the PLC against temporal safety properties. Furthermore, models have been proposed for offline analysis of periodic traffic to and from a PLC [6]. These solutions were typically provided as external solutions, where more advanced processing systems are coupled with the PLC system to verify the programming inputs of the PLC. This allows for the advanced operations that require an abundance of memory such as the calculation of advanced physical properties of a system or processing the network traffic.

Modular embedded controllers introduced the concept of coupling a PLC with an embedded hypervisor. The hypervisors are typically much more advanced embedded operating systems than the actual PLC. APIs are provided for developing programs that can be directly integrated into the programming blocks of the PLC either synchronously or asynchronously through shared memory between the PLC and the hypervisor. Development environments are provided to generate programming blocks that can call an associated library function on the hypervisor, e.g., a DLL file on a Windows hypervisor, allowing the PLC to pass inputs and take in outputs from the library function within the main PLC scan cycle [7].

In this paper, we leverage these coupled environments to implement online security solutions directly integrated into the PLC. We first provide a novel approach to implementing a cyber-physical verification solution directly integrated into the scan-cycle of the PLC using the embedded hypervisor to perform advanced calculations of the underlying physical system. We then present an online monitoring solution that provides an IDS based on aforementioned security models of periodic PLC traffic.

Before providing further details of our solutions, it is important to note that Industrial Control Systems should always be secured using a holistic approach as outlined in security standards such as IEC 62443. The layered security architecture derived from IEC 62443 can be summarized by considering Plant Security, Network Security, and System Integrity as shown in Figure 1.

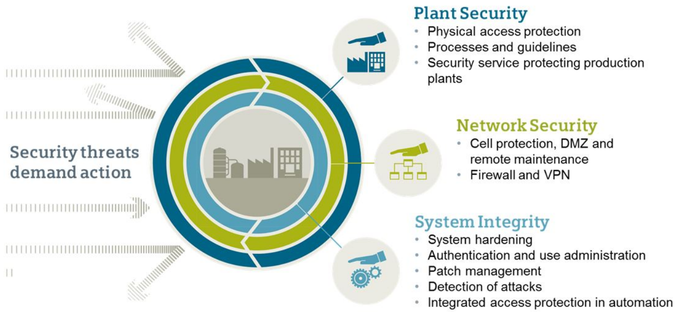
Plant Security
Physical access protection
Processes and guidelines
Security service protecting production plants
Network Security
Cell protection, DMZ and remote maintenance
Firewall and VPN
System Integrity
System hardening
Authentication and use administration
Patch management
Detection of attacks
Integrated access protection in automation
Security threats demand action

Fig. 1. The Concept of Defense-in-Depth

Security solutions as applied in an industrial context must take into account these layers of protection. For example, the solutions described in this paper are part of the “System Integrity” layer which supports “Detection of attacks”.

This paper is organized as follows. First, we provide a high-level overview of how our security solutions will be integrated into PLCs as well as our threat model in SectionII. Then we present a model for a cyber-physical verification solution that leverages the shared memory between the PLC and the embedded hypervisor in Section III. Next, we present a model for a passive intrusion detection solution within the embedded hypervisor that provides online modeling of the network traffic within the PLC in Section IV. We then show how we implemented and evaluated our security solutions in Section V. Finally, we present related work in Section VI and conclude in Section VII.

II. Overview

The two security solutions presented in this paper leverage the coupling of embedded hypervisors and PLCs. Figure 2 shows an overview of how both models would be integrated into the PLC. For our cyber-physical verification solution, programming blocks are generated and directly integrated synchronously or asynchronously into the main scan cycle of

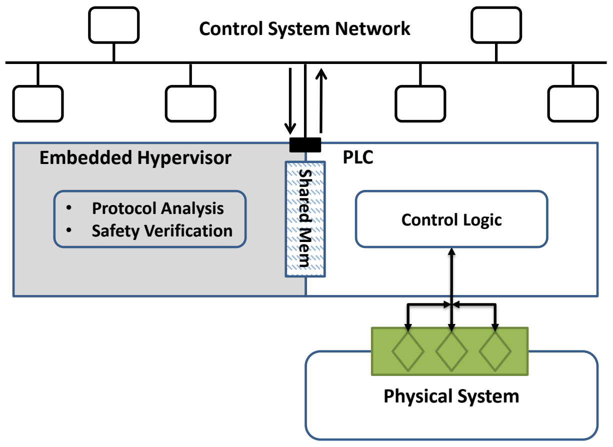

Fig. 2. System Overview. The coupled system communicates with the control system network. The PLC runs the control logic program that interfaces with the underlying physical system. The embedded hypervisor shares memory with the PLC and can run models with advanced calculations for protocol analysis and safety verification.

the PLC that share memory with a library on the embedded hypervisor. The threat model for this solution assumes that memory protection mechanisms are in place that can limit PLC clients to write to designated areas of memory. As we will detail in section III, these designated areas are treated as temporary buffers before the data along with the system state is verified within the embedded hypervisor and then forwarded to a destination buffer. Therefore, this model assumes that an attacker cannot circumvent this mechanism by directly writing to the destination buffer. If the proprietary protocol in question has been reverse-engineered, then the attacker might have the ability to remotely program the PLC and dictate the control flow of the program. The second IDS solution allows online intrusion detection from within the PLC. The threat model assumes that hypervisor is inaccessible, i.e., cannot be tampered with, and that the hypervisor shares the same Ethernet channel as the PLC. This allows the embedded hypervisor to directly monitor all traffic coming into the PLC Ethernet port and to model the PLC from within the embedded hypervisor. Additionally, in both cases, the threat model assumes that a secure reporting mechanism is in place. Although the solutions provide detection mechanisms and active verification solutions, they do not emphasize secure reporting mechanisms to the operators and/or programmers. Actionable items upon intrusion are outside of the scope of this paper.

III. Cyber-physical Verification within a PLC Scan Cycle

Previous bump-in-the-wire verification solutions have been implemented in order to symbolically verify the logical programs download to a PLC against temporal safety properties. However, these solutions rely heavily on the soundness and

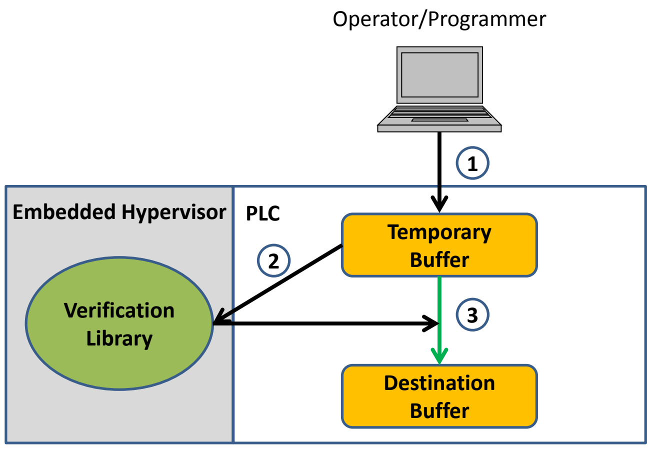

Fig. 3. CPS Verification: (1) PG/Client/HMI writes to temporary buffer; (2) A function in the verification library is called to verify this value; (3) If value doesn't violate safety constraints, the value in the temporary buffer is transferred to the destination buffer

completeness of their external, offline verification solutions. There is an inherent difficulty in defining and verifying cyber-physical safety properties given the variety of inputs in a typical PLC program and the complexity of the underlying physical invariant properties. Similarly, IDS models are passive external security solutions. Previously proposed models seem to have only been implemented for offline traffic analysis. In both cases, there is no active verification of values written to memory in the PLC. PLCs support memory protection and access control, but several programs still provide PLC clients with the capability of modifying the variables that represent discrete attributes of the cyber-physical solution.

Using PLCs coupled with embedded hypervisors, active cyber-physical verification solutions of values written to memory can be implemented and directly integrated into the scan cycle of a PLC. Our solution leverages this coupling to verify values written to areas of memory in the PLC. A high-level overview and control flow of a sample solution is presented in Figure 3.

The solution works by restricting writes to the memory in the PLC to designated temporary buffers in memory. When a write to the temporary buffer is detected, the functional programming block associated with the embedded hypervisor library function is invoked and passes the system state to the embedded hypervisor. The written value is verified against previously defined temporal safety properties based on the underlying physical model and the current system state. If the value written to the temporary buffer doesn't violate any safety or security constraints, the embedded hypervisor will return a signal to the PLC that allows this value to be forwarded to the destination memory buffer. Otherwise, the transfer will be blocked and a notification can be raised to the operator that an unsafe command has been issued.

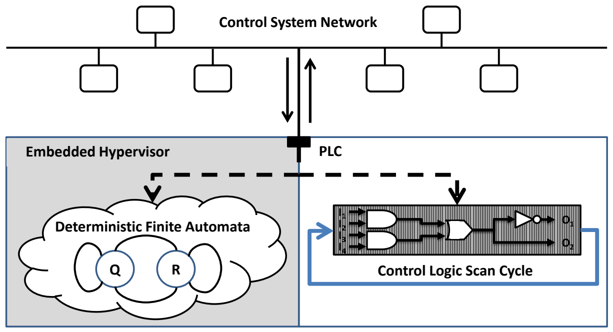

Fig. 4. Passive IDS implementation within the embedded hypervisor. The PLC and the hypervisor listen on the same port. The hypervisor maintains a model of the traffic for anomaly detection, such as the expected queries (**Q**) and responses (**R**) in an ethernet protocol, in parallel to the PLC running the control logic program.

The purpose of interacting with the embedded hypervisor is to provide the ability of advanced calculations of the underlying physical system model. For example, if a PLC is controlling significant components of an electric power grid, e.g., circuit breakers and tap changers of transformers, the embedded hypervisor can take care of running optimal power flow equations to determine the impacts of a particular action in real time, e.g., opening/closing a circuit breaker.

IV. Automaton-based Controller Anomaly Detection

The embedded hypervisors can also be used to implement online IDS from within the PLC. IDS solutions have been proposed for modeling PLC traffic for the purpose of detecting malicious packets. Our solution is based on the deterministic finite automaton (DFA) solution presented in [8] and [6]. Figure 4 presents a simple DFA example of Modbus traffic.

In this system, an expected periodic traffic pattern is a sequence of four packets: a first query (Q_1), a response to the first query (R_1), a second query (Q_2), and a response to the second query (R_2). If a subsequent packet represents the next expected state in the pattern, then we have a *Normal* transition from one DFA state to the next. If the subsequent packet is the same as the current packet, then we have a *Retransmission* and the DFA remains in the same state. If the subsequent packet is not the expected packet and is within the subset of $\{Q_1, R_1, Q_2, R_2\}$, then we have a *Miss* and the DFA state transitions to the state of the subsequent packet. If the subsequent packet is not the expected packet and is not within this subset, then we have an *Unknown* and the DFA transitions to the beginning of the pattern sequence. An *Unknown* transition is the worst type of transition and can generally be expected to be an intrusion. Further details of the DFA algorithm as well as its application to specific PLC Ethernet protocols can be found in the aforementioned papers.

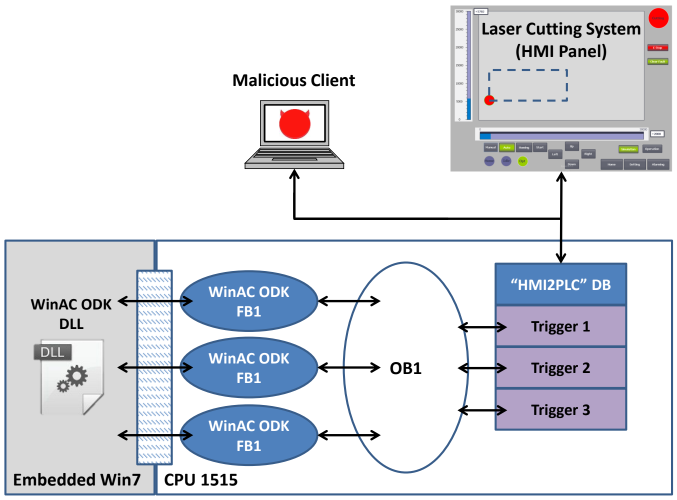

Fig. 5. CPS Verification Solution. WinAC ODK implementation allows the main scan cycle programming block, OB1, to invoke the automatically generated functional programming blocks, FB, associated with the verification library functions of the DLL located in the embedded hypervisor. The data block, "HMI2PLC" DB, can be written to by the legitimate HMI panel of our cyber-physical system or by a malicious client on the network.

V. Evaluations

In this section, we present our evaluations and implementations of the two proposed security solutions. Both of our solutions were implemented using the SIMATIC ET 200SP Open Controller, CPU 1515 PC. The PLC has a hypervisor with Windows Embedded 7E 32-Bit. We used SIMATIC WinAC Open Development Kit (ODK) to implement both of our solutions. The WinAC ODK provides an API for Microsoft Visual Studio that allows developers to generate DLLs with desired library functions to be stored on the embedded hypervisor, while also generating the associated programming blocks that are directly downloaded to the PLC and can interface with the DLL through shared memory.

A. Cyber-physical Verification Solution

The previous simple scenario was directly integrated into a cyber-physical simulation program. Figure 5 provides an overview of the cyber-physical system used in our solution.

The associated physical system in this scenario is a laser-cutting tool that places materials onto a cutting platform and cuts a particular shape specified by the operator. Typically the HMI reads from and writes to a specific DB, which we labeled "HMI2PLC". We developed an attack scenario in which a hacker using a Snap7 client to inject malicious packets that alter this DB.

We integrated the WinAC ODK functions directly into the main cyclic programming block, OB1. Table I provides the safety specifications of our sample security solution.

The first safety specification states that the system should not receive a manual direction signal–moving the cutter up, down, left, or right–while the system is in "Auto" mode, meaning that the cutting should be automatic. When OB1 detects a direction signal, a call to the associated WinAC ODK function is triggered. The WinAC ODK function will then check the relevant status bits and, if there is a violation of the safety specification, it will raise an alarm (e.g., a notification will be raised on the HMI panel). The second safety specification states that the laser-cutter's homing position (i.e., the position the cutter returns to when it has finished a full cutting-cycle) cannot change while the system is not in "Auto" mode and the system is not in "Idle" mode, which is just the mode that indicates the cutter is standing idle. If OB1 detects that either the X- or Y-coordinate of the homing position setting has changed, it will invoke the associated WinAC ODK function in the same manner to verify the change against these safety rules. If the WinAC ODK function detects a violation, it will raise a signal that forces the system to finish the current cutting cycle and stop production until the operator acknowledges the intrusion. The final specification just states that the cutting speed of the laser cannot change while in "Auto" mode and while the "Cutting" indicator is true. If OB1 detects a change in the cutting speed, it will invoke another WinAC ODK function that issues and Emergency Stop signal if the rule was violated.

Although these rules could have been easily implemented using simple ladder logic or STL programming, they serve as place holders for advanced calculations for physical equations. Our goal was to demonstrate a highly-coupled PLC security solution. Furthermore, these solutions can be directly integrated into the scan cycle timing and allow developers to account for the security solution in their timing specifications. The associated programming blocks can be invoked synchronously or asynchronously depending on the safety/operational requirements of the scan cycle.

This IDS relies on the assumption that the proprietary protocol is not reverse-engineered. If the PLC's programming protocol(s) are reverse engineered, a hacker who is able to establish a programming connection to the PLC can just program any blocks to overwrite or skip over the security implementation.

B. Online Automaton-based Anomaly Detection Solution

Our IDS solution implements an online analysis using T-Shark [9] to inspect every packet from within the embedded hypervisor. Using our knowledge of the S7-comm protocol from David Nardella's analysis, we built our solution on top of an already existing S7-comm Wireshark dissector plugin. We directly integrated the DFA IDS directly into the packet-dissection so that every packet over S7-comm protocol is processed through our model.

TABLE I
SAFETY SPECIFICATIONS FOR LASER-CUTTING SYSTEM

Trigger Signal	**Safety Conditions**	**Violation Response**
Manual Direction Click(↑, ↓,←, →)	!(Auto)	Notification
Home Position Changed	!(Auto) && !(Idle)	Stop Production
Cutting Speed Changed	!(Auto) && !(Cutting)	Emergency Stop

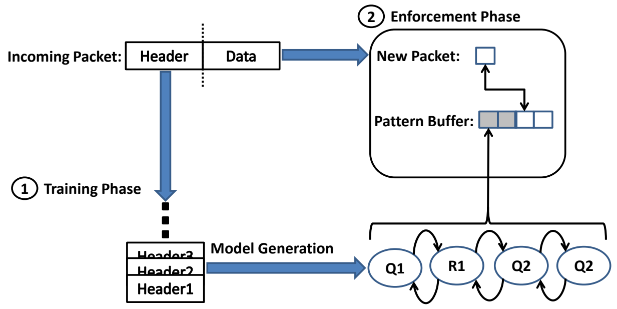

Fig. 6. Deep packet inspection from within PLC. The solution first trains the model by queueing the headers of the first 1500 incoming packets and generating a DFA based on the training set, usually as a sequence os queries, **Q**, and responses, **R**. This DFA will then be used to enforce intrusion detection within the hypervisor

Figure 6 shows an overview of our IDS implementation. T-Shark will listen on port 102, i.e., the shared PLC Ethernet port, for incoming packets and our plugin will dissect any packets that utilize the S7-comm protocol. As in [6], we have a learning stage (where the periodic pattern is learned) and an enforcement stage (where each packet is checked against the learned pattern). The symbols of our DFA are solely based on the headers of the S7-comm PDU. We split any multi-reads or multi-writes into individiual symbols. For example, if one packet specifies a write to 18 variables, we will split that packet into 18 separate symbols with the same prefix. Initially, we set our max pattern length to be 1500 symbols, with a validation window size of 6000 (these numbers taken from the latter paper). Therefore, our plugin will queue the first 6000 packets (which are assumed to be benign). Starting at a pattern length of 2 and increasing up to 1500, we check to see which pattern length best fits the periodic data. A pattern's performance is essentially determined by the number of Normals over the total number of transitions $(Normals + Misses + Retransmissions + Unknowns)$.

Once we select an appropriate pattern, we can then set this pattern as our DFA. Each subsequent symbol is checked against this DFA and any Misses, Retransmissions, or Unknowns will be reported accordingly. In our solution, we had the program write a portion of memory that would signal an alarm whenever an Unknown symbol was detected. Furthermore, we had to modify our validation window size and max pattern length as the simulation program generated much more symbols than 1500 in one cycle.

We reinforced the IDS solution by ensuring that Retransmission packets were valid. Because the DFA solution discards the actual data values being written to variables, an attacker could generate a packet that has the same symbol as a previous packet and manipulate the data. Because the pattern is periodic, the attacker can then find a way to inject the packet so that it lands in the sequence just before or after the same packet in the pattern. The DFA solution would simply identify this packet along with the extra acknowledgement packet as Retransmission symbols (since there will most likely be two acknowledgement packets in a row). To resolve this issue, we simply keep a data buffer that holds the data of the previous packet. If the current packet is identified as a Retransmission, we just compare the two data buffers and make sure nothing has been changed. Although this does not mitigate the case for Misses (as the data would not be expected to be the same), we can guarantee that valid Retransmissions are benign.

In addition to not being able to validate Miss packets, there are a couple of limitations with this IDS solution. First, it relies on the data being highly periodic. For fully automated systems where there is little to no human interaction, our IDS solution would have an extremely high intrusion-detection accuracy. However, most industrial control systems involve operators who use HMI panels to send commands to the PLCs. The simulation program was designed to simulate an operator that starts the cutting process between 1 and 10 seconds every cycle. This operation will generate one symbol, i.e., the packet the operator sends to start the cutting process. This symbol will almost always be identified as a Miss since the operator starts the process at a different point in the pattern sequence every time. This false positive could most likely be mitigated by adapting the learning process to the application-specific pattern. There are many ways to modify the algorithm by incorporating supervised learning. As a stand-alone, unsupervised process, though, our algorithm can only guarantee that Normal, Retransmission, and Unknown packets will be properly identified and handled accordingly. However, the goal of this solution was to present a sample IDS solution that can be embedded within the PLC. Having an advanced embedded hypervisor coupled with the PLC allows the system to provide online deep-packet inspection.

VI. RELATED WORK

In this section, we will present several related verification and security solutions for PLCs. It is worth noting that our solutions emphasize the ability to verify and secure the PLC from within the device, not the security models themselves.

We first review works related to the guidelines associated with securing control systems. In [10], NIST guideline security architectures are presented for ICS with respect to supervisory control and data acquisition systems, distributed control systems, and PLCs. Similar guidelines for the energy industry are presented in [11] and [12]. [13] and [14] argue that compliance with these standards provide a false sense of security.

We now discuss previous security and verification solutions presented for control systems. TSV [4] presented an external bump-in-the-wire verifier for process controller code downloaded to the PLC. Mohan et al. [15] introduced a monitor that dynamically checks the safety of plant behavior. Offline intrusion detection solutions have been proposed to model PLC traffic as a deterministic finite automaton in [8] and [6]. Another model based intrusion detection was proposed in [16]. In all cases, the security solutions were implemented as external solutions as opposed to within the PLC. Avatar [17] provides a framework to support dynamic security analysis of embedded systems firmware. However, the firmware resides below the control logic level and security/verification solutions cannot be easily integrated into the scan-cycle of the PLC. In general, our solution focuses more on application-level security solutions. [18] uses mathematical analysis techniques to evaluate various aspects, such as safety and reliability, of a given control system, but focuses on accidental failures and not malicious actions. PLC vendors themselves typically use basic security mechanisms with a single privilege level [4].

VII. Conclusions

In this paper we presented two security models for PLCs that leverage the advanced computational power of embedded hypervisors that are coupled with PLCs. We evaluated implementations of both models on a real PLC on a simulated cyber-physical system with unpredictable operation.

Acknowledgement

The authors would like to express their gratitude to George Trummer, Stefan Woronka, Ben Collar and Frank Garrabrant for their insightful feedbacks and constructive suggestions. This material is based upon work supported by Siemens as well as the Department of Energy under Award Number DE-OE0000780.

References

[1] US ICS-CERT. (2015) ICS CERT Monitor, November/December 2015. https://ics-cert.us-cert.gov/monitors/ICS-MM201512.

[2] D. Nardella. (2015) Snap7 Overview. http://snap7.sourceforge.net/.

[3] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing plcs-a new back orifice."

[4] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A trusted safety verifier for process controller code." in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2014.

[5] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated plc code analytics," *Security & Privacy, IEEE*, vol. 12, no. 6, pp. 40–47, 2014.

[6] A. Kleinman and A. Wool, "Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics," *The Journal of Digital Forensics, Security and Law: JDFSL*, vol. 9, no. 2, p. 37, 2014.

[7] Siemens. (2009) SIMATIC Windows Automation Center RTX Open Development Kit (WinAC ODK). https://cache.industry.siemens.com/dl/files/966/35948966/att_82094/v1/winac_odk_user_manual_en-US_en-US.pdf.

[8] N. Goldenberg and A. Wool, "Accurate modeling of modbus/tcp for intrusion detection in scada systems," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 2, pp. 63–75, 2013.

[9] Wireshark. (2016) tshark. https://www.wireshark.org/docs/man-pages/tshark.html.

[10] National Energy Regulatory Commission, NERC CIP 002 1 - Critical Cyber Asset Identification, 2006.

[11] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ics) security," *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

[12] R. Carlson, J. Dagle, S. Shamsuddin, and R. Evans, "A summary of control system security standards activities in the energy sector," *Department of Energy*, p. 48, 2005.

[13] J. Weiss, "Are the nerc cips making the grid less reliable," *Control Global*, 2009.

[14] L. Pietre-Cambacédes, M. Tritschler, and G. N. Ericsson, "Cybersecurity myths on power control systems: 21 misconceptions and false beliefs," *Power Delivery, IEEE Transactions on*, vol. 26, no. 1, pp. 161–172, 2011.

[15] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, "S3a: secure system simplex architecture for enhanced security of cyber-physical systems," *arXiv preprint arXiv:1202.5722*, 2012.

[16] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proceedings of the SCADA security scientific symposium*, vol. 46. Citeseer, 2007, pp. 1–12.

[17] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti, "Avatar: A framework to support dynamic security analysis of embedded systems' firmwares." in *NDSS*, 2014.

[18] W. M. Goble, *Control systems safety evaluation and reliability*. ISA, 2010.