Received 15 August 2023; revised 12 October 2023; accepted 14 November 2023. Date of publication 23 November 2023; date of current version 14 December 2023. The review of this paper was arranged by Associate Editor Jin-Gang Lai.

Digital Object Identifier 10.1109/OJIES.2023.3335976

Security of Programmable Logic Controllers and Related Systems: Today and Tomorrow

WAEL ALSABBAGH [1,2] **(Graduate Student Member, IEEE), AND PETER LANGENDÖRFER** [1,2]

[1]IHP—Leibniz-Institut für Innovative Mikroelektronik, 15236 Frankfurt (Oder), Germany
[2]Brandenburg University of Technology Cottbus-Senftenberg, 03046 Cottbus, Germany

CORRESPONDING AUTHOR: WAEL ALSABBAGH (e-mail: alsabbagh@ihp-microelectronics.com).

This work was supported in part by the German Federal Office for Information Security through EMiL project under Grant 01MO23014C.

ABSTRACT Programmable logic controllers (PLCs) are indispensable in critical infrastructures and industrial control systems. The increasing demand for enhanced cost-effectiveness and production efficiency has driven automation manufacturers to integrate PLC-based applications and systems with external networks, such as Internet. Unfortunately, this connectivity has exposed systems to potential malicious attacks from motivated adversaries. Addressing this pressing issue necessitates a comprehensive summary of ongoing research related to PLCs and their related systems. This summary should classify these systems based on disclosed vulnerabilities, potential threats, and proposed security solutions, catering to both scientists and industrial engineers. While several recent surveys have reviewed and discussed PLC security and related topics, they often fell short of covering all essential aspects comprehensively. Furthermore, prior surveys tended to focus on analyzing vulnerabilities at the system level, overlooking the vulnerabilities specific to PLCs themselves. Consequently, their findings failed to effectively secure current operational systems or propose improved solutions for future PLC designs. In this article, we bridge this research gap by providing a detailed review of all aspects concerning the security of PLCs and related systems. This includes vulnerabilities, potential attacks, and security solutions including digital forensics. We aim to offer a precise analysis, addressing the shortcomings of previous studies. Finally, we conclude this article by presenting our recommendations tailored for PLC manufacturers, researchers, and engineers. We hope that these recommendations will contribute to the development of more secure PLCs in the future.

INDEX TERMS Cyberattacks, cybersecurity, digital forensics, programmable logic controllers, security solutions, vulnerabilities.

I. INTRODUCTION

Industrial control systems (ICSs) face a wide range of threats that target the physical processes controlled by programmable logic controllers (PLCs), as shown in many incidents like Stuxnet [1], Havex [3], TRITON [4], Black Energy [5], German Steel Mill [6], and others. PLCs serve as the last defense line of any ICS system. Consequently, if a PLC is compromised, it also compromises the entire physical process it controls, potentially leading to catastrophic incidents [38]. Each year, security experts identify and report vulnerabilities concerning PLCs and their related systems to the Common Vulnerabilities and Exposures (CVE[1]) database. These vulnerabilities undergo meticulous evaluation before specialists release advisories to inform the public about the potential threats. The responsibility of publishing these advisories lies with the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT[2]). To analyze the data effectively, we utilized the ICS-CERT website's research engine to specifically filter advisories related to PLCs. Subsequently, we extracted and compiled the number of PLC-related advisories reported annually, enabling us to generate a statistical report spanning from 2010 (the year of the first reported PLC-based system attack) to the end of 2022, as shown in Fig. 1.

[1][Online]. Available: https://cve.mitre.org/

[2][Online]. Available: http://www.ics-cert.org/

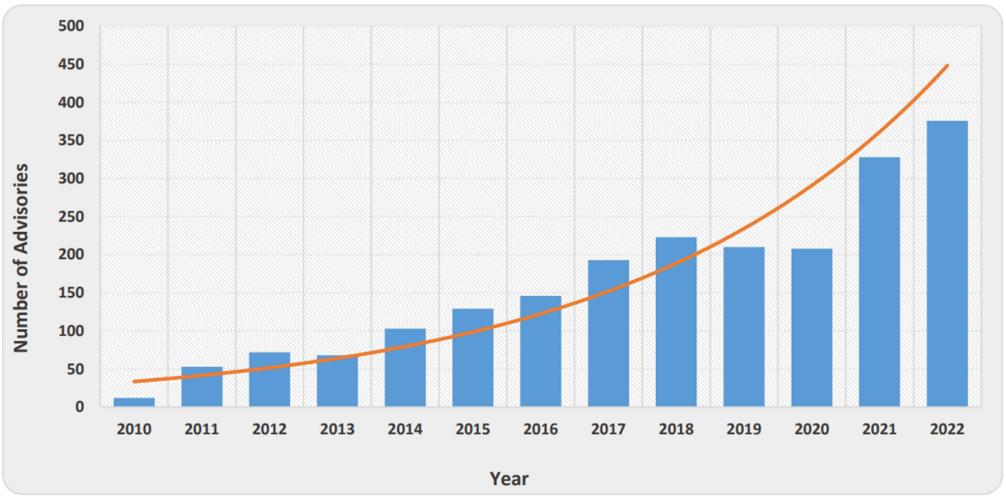

FIGURE 1. Number of advisories reported to ICS-CERT by year.

The data presented in our report indicate a significant and continuous increase in the interest surrounding the security of PLCs and their systems. Furthermore, the report shows clearly that the more identification of vulnerabilities, the more malicious attacks occur. Note that ICS-CERT advisories are only published when potential harm from attackers is detected, and most advisories contain multiple related vulnerabilities. As can be seen in Fig. 1, in 2022, there were almost 376 reported advisories, which is over 50 more than the previous year's 328 advisories and over 200 more than the number reported in 2012.

A deeper analysis to the reported vulnerabilities reveals that many of the weaknesses/entry points the attacker exploited were not novel, e.g., stack-based overflows, improper input validation, improper access control, etc. In addition, PLCs themselves often have, by default, vulnerabilities in program verification, firmware, and memory. We noticed that when adversaries gain access to a system's network, or the devices connected over the comprised network, they can exploit their vulnerabilities to conduct severe attacks such as command injection, control logic injection, firmware modification, memory corruption, replay attacks, and many others. The reported advisories showed also a notable increase in the sophistication of attack scenarios conducted, which have become more stealthy and complex. Consequently, there is an urgent need to develop a comprehensive defense mechanism against different cyberattacks. This holds true not only at the system level, like most of the other research works focused, but also directly at the PLC level. However, security measures like code verifying, firmware investigating, traffic monitoring, suspicious state checking, and more should be reconsidered as they could not yet entirely prevent the attacks. Note that applying any security solution to an ICS must meet the system's requirements it tries to protect. For instance, PLC-based systems that operate physical processes requiring real-time responses, processing continuously, interacting frequently, high availability, etc., should have security means that take those requirements into consideration . For all that, there are different challenges that engineers and research community encounter when it comes to implement security measures, which also need to be discussed and highlighted.

The digital forensic approach has recently introduced itself as a promising approach toward enhancing the security of PLCs and their related systems. Applying security schemes using digital forensic techniques can significantly increase the opportunities of disclosing exposed industrial devices, revealing ongoing attacks and detecting them at very early stage. Based on our study and findings, significant security recommendations were derived and suggested in hope to protect PLCs effectively in the future, ensuring the security of critical infrastructures (CIs) and ICSs.

A. COMPARING OUR SURVEY TO OTHERS

Our review to the previous research works showed that until this point, there are 29 surveys discussing PLC-related attacks, vulnerabilities, and security solutions [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [26], [27], [28], [29], [30], [31], [32], [36], [37]. However, upon closer investigation, we noticed that none of these surveys offered a comprehensive analysis encompassing all the critical aspects of PLC-based systems: vulnerabilities, attack scenarios, security detection solutions, and digital forensics, as depicted in Table 1. To make our table easier to read, we used certain symbols explained right below the table. The letter "I" is used to indicate that the survey has partly analyzed the corresponding security aspect, i.e., incomplete; "C" is used to indicate that the survey has fully analyzed the corresponding security aspect, i.e., comprehensive; "✓" is used to indicate that the survey has mentioned the corresponding security aspect; and "–" is used to indicate that the survey has not mentioned the corresponding security aspect.

In the following sections, we spotlight the differences between our survey and the previous ones.

TABLE 1. Comparing Our Article to Prior Ones Related to the Theme: Security of PLCs and Related Systems

Year	Reference	Target Level	Vulnerabilities	Attacks	Security Solutions	Digital Forensics
2012	[10]	PLC	✓/ I	-	✓/ I	-
2016	[9]	System	✓/ C	✓/ C	✓/ C	-
2016	[11]	System	✓/ I	✓/ I	✓/ I	-
2016	[12]	System	-	-	✓/ I	-
2016	[24]	System	-	✓/ I	-	-
2017	[13]	System	-	-	✓/ C	-
2017	[17]	PLC	-	✓/ I	-	✓/ I
2017	[14]	System	✓/ C	✓/ C	✓/ C	-
2018	[15]	PLC	✓/ I	✓/ I	✓/ I	-
2018	[23]	System	-	-	-	✓/ C
2018	[16]	System	✓/ I	-	✓/ C	-
2019	[18]	System	-	✓/ C	-	-
2019	[19]	System	-	✓/ C	✓/ I	-
2019	[25]	System	-	-	✓/ I	-
2020	[20]	PLC	✓/ I	✓/ I	✓/ I	-
2020	[21]	System	-	✓/ C	-	✓/ C
2020	[22]	System	-	-	✓/ I	-
2021	[26]	PLC	-	✓/ I	✓/ I	-
2021	[27]	System	✓/ I	✓/ I	✓/ I	-
2021	[28]	System	-	-	-	✓/ I
2021	[29]	System	-	-	✓/ C	-
2022	[30]	System	✓/ C	✓/ C	✓/ C	-
2022	[36]	System	✓/ I	✓/ I	✓/ I	-
2022	[37]	System	✓/ C	✓/ C	✓/ C	-
2022	[31]	System	-	-	-	✓ I
2022	[32]	PLC	✓/ I	✓/ I	✓/ I	-
2022	[33]	PLC	✓/ I	✓/ I	✓/ I	-
2023	[34]	System	-	-	✓/ I	-
2023	[35]	System	-	-	-	✓/ C

(I) Incompletely Analysis; (C) Comprehensive Analysis; (✓) Mentioned; (-) Not mentioned.

1) INVESTIGATING THE SECURITY AT PLC LEVEL

The security of PLC-based systems has been primarily studied from two perspectives: the system level and the PLC level. After reviewing previous surveys, we found that most research papers focused on the overall system security, particularly the security of the supervisor control layer (see Fig. 4). The authors of most published surveys primarily concentrated on network security, with a specific emphasis on communication protocols [12], [18], [19]. In contrast, only seven papers delved into the security of the automatic layer and investigated the security of the PLC itself [10], [15], [17], [20], [26], [32]. However, these studies have not offered a comprehensive overview addressing all the aspects of the PLC security theme. In this article, we aim to shed light on critical security issues existing within the PLC itself, including areas like control logic, memory, firmware, input/output (I/O) pins, and more.

2) ADDRESSING ALL SECURITY ASPECTS

The surveys mentioned in Table 1 did not fully cover the prime four security aspects: vulnerabilities, attack scenarios, security solutions, and digital forensics. Furthermore, they did not include detailed discussions about the existing vulnerabilities in PLCs and their related protocols as well as systems. For instance, the attack classifications provided in previous studies like [26] and [32] were not fully elaborated upon, and some detection methods were only partially covered in one or a few papers, as done in [13], [16], and [22].

3) DEALING WITH ATTACKS AGAINST PRESENT PLC-BASED SYSTEMS

It is not secret that running industrial systems have vulnerabilities and exposed to different cyberattacks. These vulnerabilities can be attributed to either inadequate security considerations during the initial design of industrial components or improper configurations of application hardware and software. Previous works have not thoroughly addressed how existing systems can effectively deal with cyberattacks, particularly when it comes to analyzing security measures, which is one of the main issues we focus on in this article.

4) PROPOSING FUTURE PLC DESIGNS

Engineers face significant challenges when attempting to incorporate security measures into existing operational systems, particularly in the case of older PLCs that possess limited computational power, storage capacity, and bandwidth. As a result, it becomes imperative to introduce more sophisticated security strategies while designing future PLC devices to improve their overall security [29]. However, existing surveys on this subject have largely overlooked this aspect and failed to provide a foundational concept of how these future PLCs might look like. In contrast, our study delves into the ongoing research on digital forensic theme. Furthermore, we analyze also the development of security solutions and approaches. Through this exploration, we aim to offer valuable insights and directions for the forthcoming generation of PLC

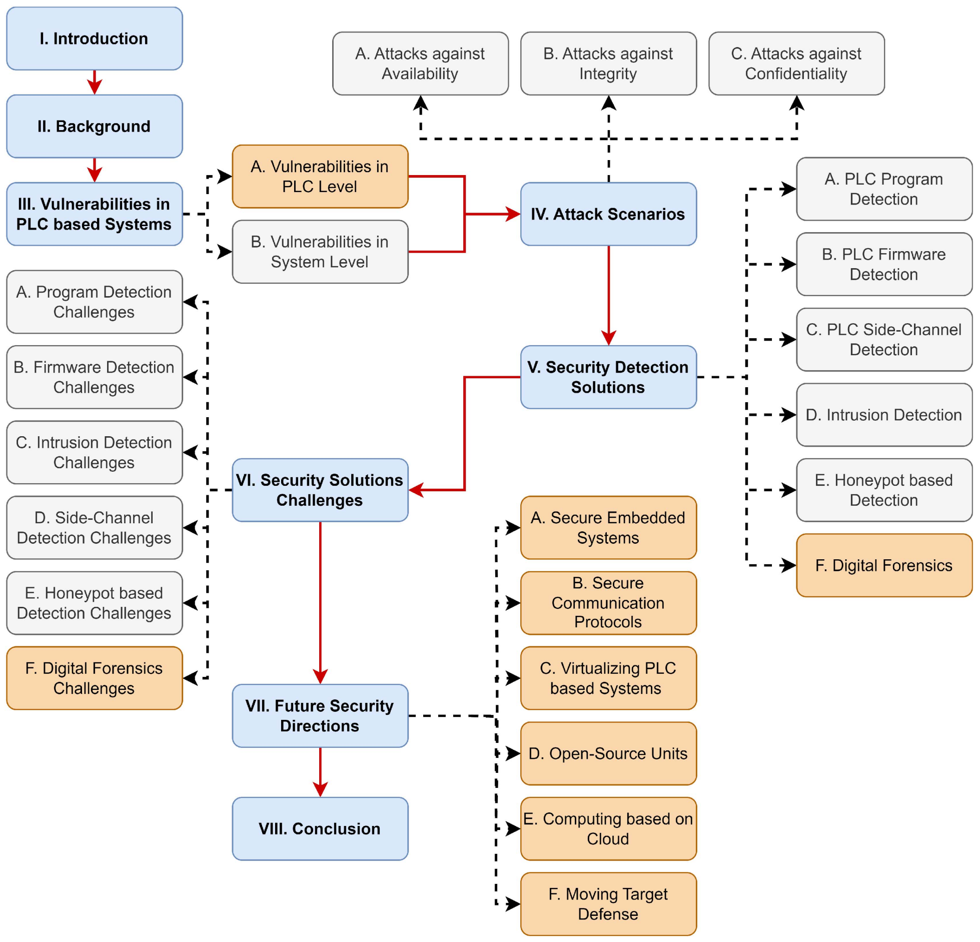
I. Introduction
II. Background
III. Vulnerabilities in PLC based Systems
A. Vulnerabilities in PLC Level
B. Vulnerabilities in System Level
A. Attacks against Availability
B. Attacks against Integrity
C. Attacks against Confidentiality
IV. Attack Scenarios
V. Security Detection Solutions
A. PLC Program Detection
B. PLC Firmware Detection
C. PLC Side-Channel Detection
D. Intrusion Detection
E. Honeypot based Detection
F. Digital Forensics
VI. Security Solutions Challenges
A. Program Detection Challenges
B. Firmware Detection Challenges
C. Intrusion Detection Challenges
D. Side-Channel Detection Challenges
E. Honeypot based Detection Challenges
F. Digital Forensics Challenges
VII. Future Security Directions
A. Secure Embedded Systems
B. Secure Communication Protocols
C. Virtualizing PLC based Systems
D. Open-Source Units
E. Computing based on Cloud
F. Moving Target Defense
VIII. Conclusion

FIGURE 2. Article structure diagram: Boxes in blue refer to main sections, boxes in gray refer to topics discussed in previous surveys, and boxes in orange refer to our contributions in this article.

devices, distinguishing our work from similar surveys like [9], [14], [30], and [37].

B. MOTIVATION

Despite the fact that many surveys discussed the security of PLCs and their related control systems, the research community still lacks a comprehensive review that addresses all these aspects together concentrating on the PLC device individually as well as the connections with other industrial devices operating in the same network. Therefore, it is essential to first understand how malicious adversaries have successfully exploited vulnerabilities in PLCs using different attack techniques, and also how existing security solutions have defended against those attacks. In this article, we aim to shed more light on the missing issues in the previous surveys and provide security recommendations that help in achieving more secure and promising PLC-based systems in the future, with the hope that our findings will give a further boost to the research community.

C. SCOPE OF THIS ARTICLE

Between 2010 and 2022, several academic works, related to the security of PLCs, have been published. Most of these contributions are publicly available on academic research databases and digital libraries, e.g., IEEE Xplore, ACM, DBLP, Science Direct, Springer Link, Microsoft Academic, etc. We manually extracted those publications using specific keywords such as review, survey, overview, vulnerabilities, supervisory control and data acquisition (SCADA), ICS, PLC, attacks, mitigation, detection, security, and forensics. In addition, we reviewed the references of these located papers to identify other relevant works. In total, we found more than 6000 papers published in mainstream journals and conferences. For this article, we have carefully selected a short list of 235 papers to provide a comprehensive and coherent review. In the following sections, we outline the criteria and terms we considered while selecting the literature for this article.

1) STUDY INVESTIGATES THE SECURITY OF PLCS

We center our attention on exploring vulnerabilities, attack scenarios, security solutions, and digital forensics related to PLCs and their systems.

2) STUDY PLAYS AN IMPACTFUL ROLE IN THE RESEARCH COMMUNITY

In our discussion, we focus on highly referenced papers within the realm of PLC-based systems. These papers shed light on

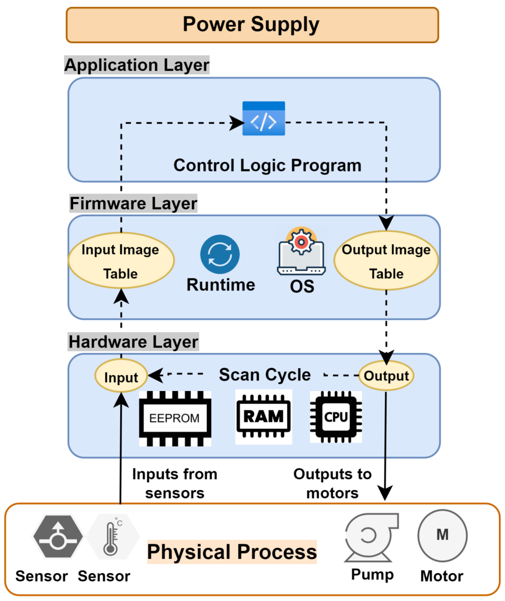

FIGURE 3. Typical PLC architecture.

the latest vulnerabilities that the authors have identified and exploited to inflict damage on the target systems. To ensure precision, we excluded the works with fewer than 20 citations, unless they introduce novel attack methodologies or reveal new vulnerabilities.

3) STUDY DISCOVERS A NEW DIRECTION FOR FORTHCOMING RESEARCH

We take into consideration the papers that suggest new paths for further research or potential detection schemes or directions to be considered.

D. CONTRIBUTION

The primary contributions we introduce in this work are as follows.

1) COMPREHENSIVE REVIEW

In comparison to the already published surveys that are related to the security of PLCs, our work demonstrates a comprehensive review, encompassing four aspects: vulnerabilities, threats, security solutions, and digital forensic. Based on our findings, we suggests security recommendations that help in strengthening the overall security of our PLC-based systems.

2) DUAL INVESTIGATIONS ON PLCS

Our analysis to the weak/entry points takes a dual perspective, examining both the PLC itself and its entire system. When considering the PLC, we primarily address the program, memory, and firmware aspects. For the system as a whole, we concentrate on specific-vendor software and protocols, as well as other components connected to the controller.

3) SECURITY APPROACHES FOR DEPLOYED PLCS

In relation to currently running PLC systems, this article introduces various security approaches that are categorized based on the detection object, i.e., program, firmware, fingerprint, intrusion, and honeypot-based detection.

4) DIGITAL FORENSIC APPROACHES ARE INCLUDED

We delve into the methodologies, challenges, and implementations of digital forensic approaches, specifically those are dedicated to protect PLC-based systems.

5) CLASSIFYING SECURITY SOLUTIONS CHALLENGES

Besides that our article provides a deep analysis to vulnerabilities, attacks, and security solutions related to PLCs, it reviews the challenges that implementing each security approach encounters, classifying them based on the security solutions introduced in this work.

6) FUTURE SECURITY DIRECTIONS

Looking ahead to more secure future systems, we suggest our six security recommendations summarizing our findings in this article.

The rest of this article is organized as follows (see also Fig. 2). Section II introduces the basic background of PLCs and their related protocols and security requirements. An intensive vulnerabilities analysis is conducted in Section III, while the attack scenarios are discussed in Section IV. We review the existing security solutions in Section V and discuss the security solutions challenges in Section VI. Six future security directions are suggested in Section VII. Finally, Section VIII concludes this article.

II. BACKGROUND

In this section, we provide an overview of a typical PLC architecture, its operational environment, communication protocols, related systems, and the security demands specific to PLC-based systems.

A. PLC ARCHITECTURE

Fig. 3 depicts the typical architecture of a standard PLC. It is a digital device designed to control machines, industries, and plants through programmed instructions. In its simplest form, a PLC consists of various components. These include a power supply, input and output modules, an operating system (OS), and memory components like random access memory (RAM) and electrically erasable programmable read only memory (EEPROM). The PLC also features an interface for uploading and downloading user programs to and from the engineering workstation (EWS). The OS and the user-specific program are stored in the EEPROM. Input devices, such as sensors and switches, provide real-time data about the physical process to the PLC. The PLC processes this information through its control logic and drives the physical process accordingly using output devices like motors and valves. To program the

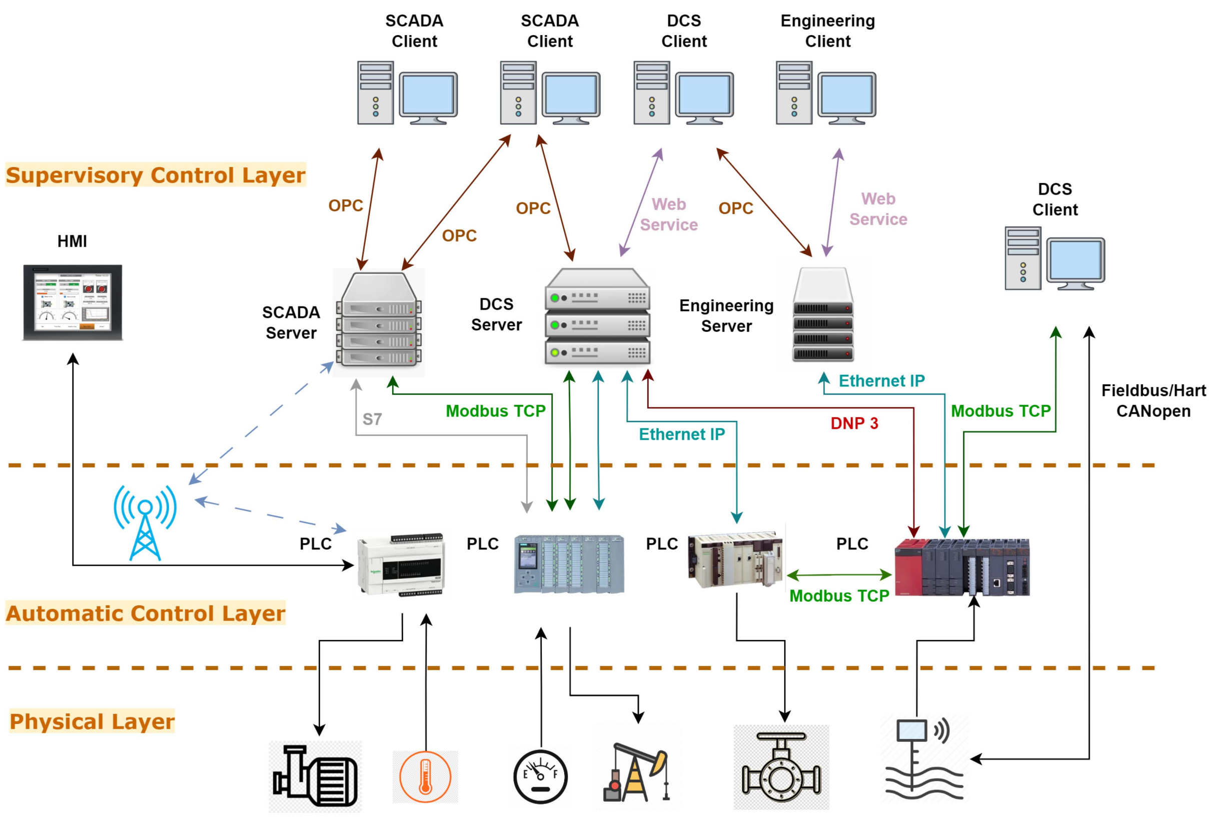

FIGURE 4. SCADA system architecture.

PLC, control logic programs are used. These programs perform specific tasks based on the input readings and output conditions. There are five PLC programming languages defined in IEC-61131 [7]: ladder diagram (LD), structured text (ST), sequential function chart (SFC), instruction list (IL), and function block diagram (FBD) . The firmware in the PLC serves as the OS, facilitating the exchange of interactive data between the physical world and the device.

B. PLC RUNTIME OS

PLCs operate on a real-time OS, specifically designed to execute certain tasks through cyclic repetition of command sequences in extremely short intervals, typically in milliseconds. Each execution cycle, known as a scan cycle, consists of four main steps. In the first step, the central processing unit (CPU) reads data from connected sensors and stores the obtained values in a data table or an input image. Next, the logic execution updates the inputs of the running program with the newly acquired sensor values. Following that, the control logic is executed, and the output statuses are updated accordingly [32]. The fourth step deals with communication tasks, facilitating data exchange with devices connected to the PLC. After completing the communication scan, the PLC enters a maintenance phase. During this phase, various internal tasks are performed e.g., updating internal clocks, managing memory, and other essential system maintenance activities. Although the user is not informed about this maintenance sequence, it regularly runs in the background as a crucial part of the PLC's functioning.

C. COMMUNICATION PROTOCOLS

A protocol refers to a collection of rules governing how devices in a network communicate with each other. In the context of PLCs, communication protocols are essential for establishing connections with remote I/O devices, remote control devices, and engineering software. PLC communication protocols are mostly proprietary, i.e., each manufacturer has his own protocols. However, these proprietary protocols have undergone scrutiny and reverse engineering by both academic researchers and industry experts. This is due to the fact that these protocols allow programming software to access the physical memory of PLCs [219]. As a result, efforts have been made to understand and study these protocols in more detail.

D. PLC-BASED SYSTEMS

In the context of PLC systems, a prime example we can give is the SCADA system. In this regard, we present the architecture of a SCADA system, along with an overview of industrial communication protocols commonly utilized in current PLCs. Fig. 4 depicts a contemporary SCADA system, which is structured into the following three layers.

1) *Supervisory control layer:* It is responsible for overseeing and managing monitoring operations. Its primary function is to collect data from various sources.

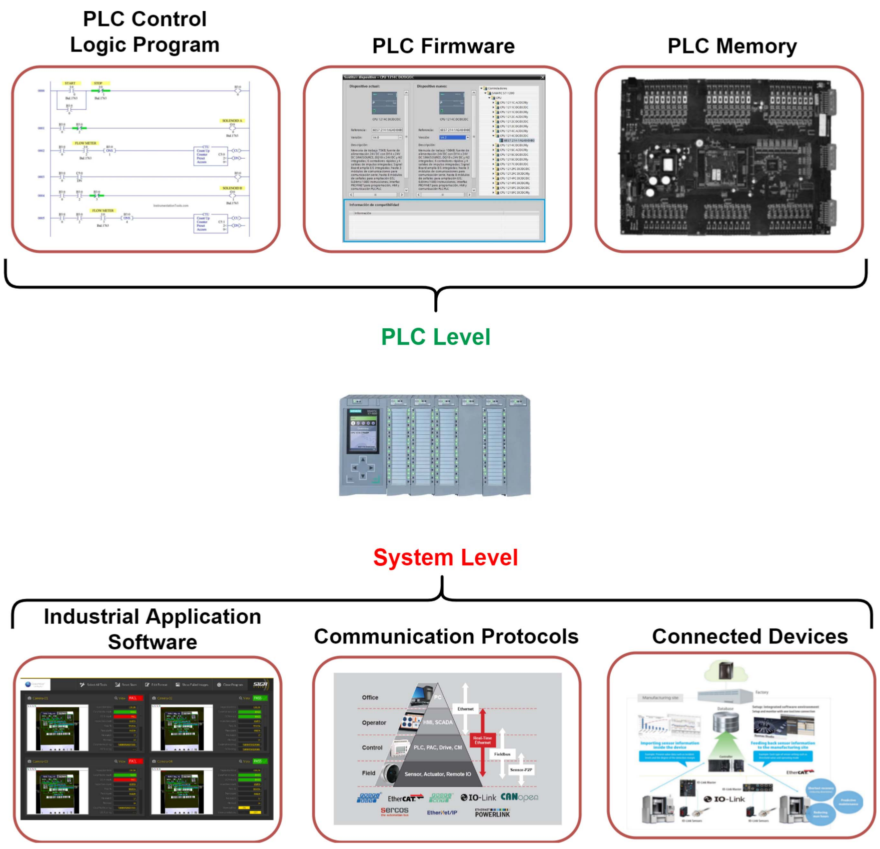

FIGURE 5. **Classification of PLC- and system-level vulnerabilities.**

2) *Automatic control layer:* This layer aims at regulating physical processes. Thus, it operates with the help of control commands. It is where PLC devices are situated, handling the execution of control instructions.
3) *Physical layer:* At the lowest level, this layer contains the physical devices that run the machinery. It is under the control and interacts with the upper layers.

Fig. 4 clearly shows that PLCs control actuators/motors and sensors/switches. Simultaneously, other devices are connected with the PLCs, e.g., human–machine interface (HMI), EWS, historian, and control servers. The HMI communicates with the PLC using certain ports, allowing it to read/write data directly from/to the connected PLC. On the other hand, the EWS is responsible for reading and writing programs and configurations from/to the PLC.

E. SECURITY REQUIREMENTS FOR PLC-BASED SYSTEMS

In contrast to typical information technology (IT) system security, PLC-based systems and ICSs have distinct priorities, focusing on availability, integrity, and confidentiality. Addressing security concerns in PLC-based systems requires considering specific requirements, which are as follows [9]:

1) ensuring high availability for each layer within the system;
2) maintaining the integrity of industrial processes;
3) sustaining continuous operations over an extended operational life span;
4) managing complex interactions and cooperation between the layers and with the physical world;
5) providing hard real-time responses;
6) managing different and wide-distributed components;
7) supporting multiproprietary communication protocols;
8) utilizing significant numbers of legacy subsystems.

III. PLC-BASED SYSTEMS VULNERABILITIES

This section analyzes and discusses the vulnerabilities affecting both the PLC individually and its related systems. It is worth mentioning that the comprehensive list of vulnerabilities is extracted from the 235 papers that we reviewed. These vulnerabilities are classified into two levels, i.e., the PLC level and the system level, as illustrated in Fig. 5.

At the PLC level, vulnerabilities primarily exist in programs, firmware, and memory. On the other hand, at the system level, vulnerabilities are mostly found in industrial application software, communication protocols used in the industrial environment, and connected devices. Moving forward, we analyze the vulnerabilities in each level.

A. VULNERABILITIES IN PLC LEVEL

The widespread knowledge is that the original design of PLCs did not adequately consider security aspects [38], [99]. Since PLCs are extensively used in various infrastructure sectors, replacing the existing legacy PLCs would be extremely challenging and almost unfeasible. As a result, these factors have given rise to numerous attacks specifically aimed at exploiting various vulnerabilities in the design of those legacy PLCs.

1) VULNERABILITIES IN PLC PROGRAMS

Control logic programs are typically written using one of the five programming languages mentioned earlier. However, these programs often contain critical flaws due to the way they are designed. Such vulnerabilities can compromise the integrity and availability of PLCs, either intentionally or accidentally. Programmers may unknowingly create backdoors for potential adversaries or inherit PLC programs with dormant threats due to a lack of professional knowledge and skills. Some common issues include the use of duplicated instructions, snooping, missing certain coils or outputs, and bypassing or denial of service (DoS) [40], [41]. For instance, the ladder logic program itself is susceptible to malware insertion because it lacks proper authentication before being downloaded into PLCs. This makes it possible for malware, such as Ladder Logic Bombs demonstrated in [42] to be embedded in LD code with a dormant state that can be activated at any time, posing a significant threat. Serhane et al. [41] further explored vulnerabilities and bad code practices in LD programming that may lead to bugs and potential exploitation by attackers. Furthermore, Valentine [43] illustrated how adversaries can install a jump to a subroutine function and manipulate the intercommunication between different ladders in an LD code. All these factors contribute to the security risks associated with LD programming in PLCs.

2) VULNERABILITIES IN PLC FIRMWARE

PLCs rely on firmware, which acts as an embedded OS, such as Microware OS-9, VxWorks, and Microsoft Windows, to achieve their computing objectives. Despite their complexity, the current firmware used in PLCs suffers from security weaknesses and susceptibility to attacks, much like the original OSs. An example is the Backhoff CX5020 utilizing Windows CE 6.0 Plus, which possesses exploitable flaws [44]. Surprisingly, many similar vulnerabilities were discovered in typical microprocessor-based devices. Consequently, attacking PLCs may not require exploiting specific vulnerabilities, but rather getting access to the controller and manipulating its regular operation. Such vulnerabilities could lead to firmware modification attacks and other disruptive actions that impact the normal functioning of PLCs. For instance, Basnight et al. [51] scrutinized Allen-Bradley ControlLogix PLCs, specifically the ControlLogix 161 PLC firmware, and uncovered weaknesses in the firmware update validation that facilitated firmware counterfeiting. Another research group [52] found security issues with source and data authentication in firmware uploads for both Koyo and Rockwell Automation PLCs. Schuett et al. [53] conducted research involving the extraction and analysis of firmware images to identify execution paths. The findings allowed the repackaging of firmware with a malicious attack, triggering a DoS attack by combining control commands.

3) VULNERABILITIES IN PLC MEMORY

PLCs are comprised of two types of memories: main and register memory. The first stores the control logics, while the latter serves as a temporary memory for processing the control logics, refreshed by the CPU in each scan cycle. Sandaruwan et al. [45] found out that critical variables influencing the main logic are stored in the register memory. Surprisingly, certain personal computers within the PLC network might have permission to access the register memory of a PLC, allowing adversaries to write malicious instructions, as if they were legitimate ICS operators. As a result, a potential attack scenario arises: if an attacker gains entry to a PLC and injects malicious values into the register memory, memory corruption attacks become feasible. Rais et al. [46] conducted forensic analysis on Allen-Bradley PLCs at the hardware level, and they found that stolen information caused the PLC to crash. Furthermore, the authors exposed memory dumps of the tested PLCs, which could potentially be exploited in firmware modification attacks.

B. VULNERABILITIES IN SYSTEM LEVEL

To effectively manage and oversee the physical processes of an ICS, seamless coordination among all components and layers is essential. For instance, the PLCs engage with the industrial application software through communication protocols, enabling data transfer and control commands among various ICS devices across different layers. However, from a security perspective, there are risks wherein attackers could commandeer a running PLC by exploiting potential vulnerabilities. These vulnerabilities include manipulating the communication protocols, compromising the integrity of the industrial application software, and infecting connected devices. In the following subsections, we elaborate on each type of those vulnerabilities in more detail.

1) VULNERABILITIES IN COMMUNICATION PROTOCOLS

Although the current network protocols effectively facilitate communication between PLCs and other industrial devices, they suffer from critical security shortcomings that leave them vulnerable to malicious manipulations. Our assessment has identified three prevalent vulnerabilities in industrial communication protocols, outlined as follows.

1) *Absence of authentication:* This allows malicious users to gain unauthorized privileges and manipulate protocol packets without any identification mechanism in place.
2) *Absence of authorization:* Adversaries are capable of exploiting the order of executing PLC code to send packets to others, potentially leading to harmful consequences.
3) *Absence of encryption:* This exposes transparent data, enabling malevolent users to capture and misuse it for their harmful endeavors. For instance, the S7 protocol has been found to lack authentication, leading to attacks on Siemens PLCs, as highlighted by Alsabbagh and Langendörfer [38], [39], [48], [49], [50]. Moreover,

TABLE 2. **Attack Scenarios in Existing Works**

Attack	Target	Vulnerabilities	Security Goal	Reference
Firmware Modification	PLC firmware layer	Firmware Updates	Availability	[52], [55]
Memory Corruption	PLC I/O Memory	Memory Layout	Availability	[56]–[58]
Denial of Service	Service or resources	PLC protocols and ports	Availability	[59]–[62]
HMI-Exploited Attack	Service or resources	insecure communication channels	Availability	[76], [91]–[94]
Time Delay Injection Attack	Service or resources	insecure networks	Availability	[95]–[98]
Payload Attack	PLC control logic	Authorization	Integrity	[63]–[65]
Control flow Attack	control logic operation	Execution flow of processes	Integrity	[41], [43]
Configuration Modification Attack	PLC parameters	Bad code practice	Integrity	[74]
Injection Attack	PLC control logic	insecure protocols	Integrity	[38], [39], [49]–[51], [67]–[72], [82], [83], [85], [101]–[103]
I/O pin Control	PLC I/O pins	I/O pin flaws	Integrity	[73], [74]
MitM Attack	PLC control logic, I/O, Ports, Passwords	Insecure protocols, poor authentication	Confidentiality	[38], [39], [45], [49]–[51], [75]–[81], [84]–[87]
Eavesdrop Attack	Read critical data	insecure networks	Confidentiality	[39], [75], [82], [101]
Reconnaissance attack	Gather information, e.g., addresses and function codes	Insecure networks	Confidentiality	[39], [99]
Brute-force Attack	Authentication and poor authentication protocols	Insecure networks	Confidentiality	[38], [75], [82], [84], [88], [90]
Replay Attack	PLC control logic, I/O, Ports, Passwords	insecure protocols, poor authentication	Confidentiality	[38], [39], [45], [49]–[51], [75], [76], [79], [82], [84]–[90]

these vulnerabilities can be exploited to execute replay attacks, man-in-the-middle (MitM) attacks, control logic injection attacks, and many others.

2) VULNERABILITIES IN APPLICATION SOFTWARE

Any PLC specific-vendor software is primarily comprised of three components: a software to program PLCs, a software to configure the network, and a software to manage SCADA systems. When these software are exploited, the devices that use them also become vulnerable. Attackers gain the ability to upload malicious code, alter PLC settings, and access critical data, among other actions. An illustrative case of industrial application software compromise is seen in the Stuxnet malware [1]. Notably, the Stuxnet attackers targeted the Iranian nuclear plant and leveraged at least four zero-day vulnerabilities, making it a unique and sophisticated attack [1]. Furthermore, Leverett and Wightman [47] demonstrated that CoDeSys, a third-party programming software, could modify PLC code. The software the author used ensured the integrity of executing the control process, which potentially opened the door for further malicious exploits, e.g., injection attacks.

3) VULNERABILITIES IN CONNECTED DEVICES

The proper functioning and dependability of PLC operations might be compromised by incorrect statuses and fraudulent I/O from connected devices, such as communication processors, I/O modules, and HMIs. Serhane et al. [41] highlighted that HMIs and historians have become more susceptible to security breaches due to the rise in remote access, particularly Internet-facing systems. Consequently, this increased vulnerability has attracted numerous adversaries seeking to exploit exposed connected devices that possess vulnerable initial access, thereby infecting networks. As a result, PLC programs are adversely affected by receiving false commands and deceptive I/O values from these compromised devices [78].

IV. ATTACK SCENARIOS

In this section, we delve into the attack scenarios used by adversaries once they exploit the vulnerabilities mentioned in Section III. Through our study to the prior research, we have identified 15 distinct types of attacks. To better organize these attacks, we have classified them into three categories based on the confidentiality integrity availability (CIA) security model's properties [155]: attacks against availability, integrity, and confidentiality.

In Table 2, we provide a summary of our analysis to the previous studies that discussed different attack scenarios aimed at PLC-based systems.

A. ATTACKS AGAINST AVAILABILITY

These attacks target the obstruction of authorized users' access to data or their ability to utilize specific resources when required. The group comprises five distinct attack scenarios, which are as follows: firmware modification, memory corruption, DoS, delayed, and HMI-exploited attack. In the following subsections, we overview each attack scenario, providing a more comprehensive review of their characteristics.

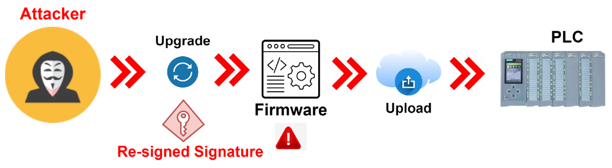

FIGURE 6. **Firmware modification attack.**

1) FIRMWARE MODIFICATION ATTACK

Fig. 6 depicts this attack scenario. The validation process for firmware updates lacks robust security measures, making it susceptible to unauthorized adversaries who can modify the firmware with malicious code.

This type of attack involves the adversaries downloading a re-signed firmware containing the malicious code and then injecting it into the PLC to create a hidden backdoor. The consequences of such attacks can be catastrophic, leading to significant failures and severe consequences. To exploit PLC firmware vulnerabilities, attackers typically employ reverse engineering techniques to infer the firmware update validation method. They analyze this method to find weaknesses that facilitate firmware modification and counterfeiting. By exploiting these weaknesses, attackers create a counterfeit firmware sample with malicious code, which they upload and execute on the target PLC.

Basnight et al. [51] noted that to modify a firmware, attackers need to apply reverse engineering techniques on the binary firmware to obtain certain functions through disassembly. Detecting and verifying a malicious firmware running on PLCs proved challenging, making PLC firmware modification a stealthy and hard-to-detect threat. Garcia et al. [54] developed a PLC rootkit, called Harvey, designed to exploit power grid systems. This rootkit infected the firmware of PLCs, allowing it to tamper with all the inputs and outputs of the PLCs in an arbitrary manner. Another research group [52] investigated the functionality of firmware upgrades and updates supported by PLCs. They demonstrated a firmware modification attack scenario against Koyo and Rockwell Automation PLCs, showing how malicious firmware could be uploaded into the Ethernet cards of these field devices.

2) MEMORY CORRUPTION ATTACK

PLC memory attacks involve unauthorized manipulation of critical memory data in PLCs, as shown in Fig. 7.

Our analysis to these attacks has revealed multiple vulnerabilities e.g., buffer overflows, incorrect mappings between memory addresses and protocol elements, etc. Thus, once attackers get access to a target network, they can maliciously manipulate different types of data stored in the compromised PLC memory, e.g., control, configuration, and decision-making data. Those manipulations can be done by overwriting specific memory locations relevant to I/O operations, thus tampering with set-point variables. In their research, Robles-Durazno et al. [55], [56] focused on attacking the PLC input memory. The authors devised an attack approach that pushes

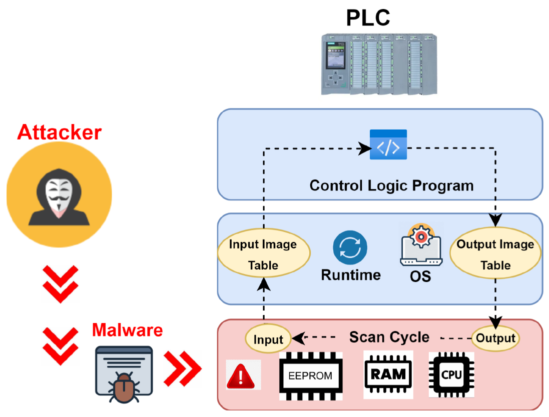

FIGURE 7. **Memory corruption attack.**

carefully crafted packets to the input memory. To this end, the authors assumed that the attackers already had control network access. Afterward, three PLC memory corruption attacks were proposed. All targeted different locations of the PLC memory.

The first attack involved overwriting bytes of memory allocated to external sensors. Similarly, the second attack affected the memory associated with outputs. The third attack targeted the working memory to modify set-point variables, including high or low alarms, and values in control systems. Furthermore, Zubair et al. [57] were successful in injecting a malicious program that modified a table entry in the RAM memory, effectively redirecting a built-in call to a malicious function. Notably, they managed to conceal their attack by erasing any traces from the protocol mapped space after its execution.

3) DOS ATTACK

A major security vulnerability in most PLCs lies in their design, where they accept requests from any Internet Protocol (IP) or media access control (MAC) address. Consequently, one of the primary attacks to ICS systems is a DoS attack. A DoS attack is technically not a specific attack type but rather a goal that involves various attack methods. The objective of a DoS attack is to disrupt the availability of a service or resource, hindering legitimate access to authorized resources and interfering with their intended utilization.

Tacliad et al. [58] identified a particular scenario of DoS attack conducted through the Ethernet/Industrial Protocol Fuzz, targeting the Programmable Controller Communication Command (PCCC) service. They found that using an invalid data file type caused data reading failures. Moreover, DoS attacks are often IP-oriented attacks. Another research [59] demonstrated different DoS attack scenarios by spoofing IP addresses between both S7 PLCs and their software i.e., Total Integrated Automation (TIA) Portal. In the same way, Sayegh et al. [60] performed IP packet flooding on the PLC's ports; precisely, they launched DoS attacks between two connected devices (PLCs and HMIs) using four different methods. Furthermore, the DoS attack presented in [61] showcased that an

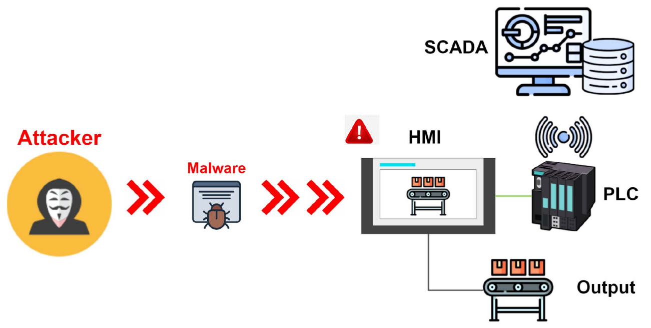

FIGURE 8. HMI-exploited attack.

attacker can use various types of packets, such as Synchronize (SYN) packets, S7Communication (S7Comm) packets, User Datagram Protocol (UDP) packets, Transmission Control Protocol (TCP)/IP packets, and more to exploit PLCs.

4) HMI-EXPLOITED ATTACK

The HMI device helps human operators to observe and control industrial processes. It allows them to monitor the process state, adjust control settings, and intervene manually during emergencies. In addition, operators can configure control algorithms and parameters in the connected PLCs. Unfortunately, these functionalities also make the HMI an attractive target for potential attackers who seek to exploit vulnerabilities in both the HMI's software and hardware components (see Fig. 8).

Common vulnerabilities include memory corruption, weak credential management, lack of proper authentication/authorization, code injection bugs, etc. Researchers like Kleinmann et al. [91], Rosa et al. [90], Hu et al. [92], and Alsabbagh et al. [75] have investigated various attack scenarios against real HMIs. Kleinmann et al. [91] demonstrated attacks by hijacking the communication channels between the HMI and PLCs. By manipulating the traffic, they were able to present a fake view of the industrial process to the operator, leading to potential damage to the system. Rosa et al. [90] crafted deceptive Modbus frames transmitted between PLCs and an HMI, creating a false view for the SCADA operator. Hu et al. [92] presented a sophisticated multistage semantic attack against ICS, which could bypass existing intrusion detection systems (IDSs). By hijacking the communication channels between the HMI and the remote PLC, they manipulated measurement data and control instructions while presenting a fake view of the data to the HMI to conceal the malicious activity. Alsabbagh et al. [75] introduced a stealthy false command injection attack using a database containing real Modbus request–response pairs between PLC and HMI devices. By skillfully managing communication flows, they tricked the SCADA operator, showing them fake views, while the PLC processed malicious commands from the attacker. These studies highlight the importance of strengthening the security measures around HMIs to protect CI from potential cyber-physical attacks.

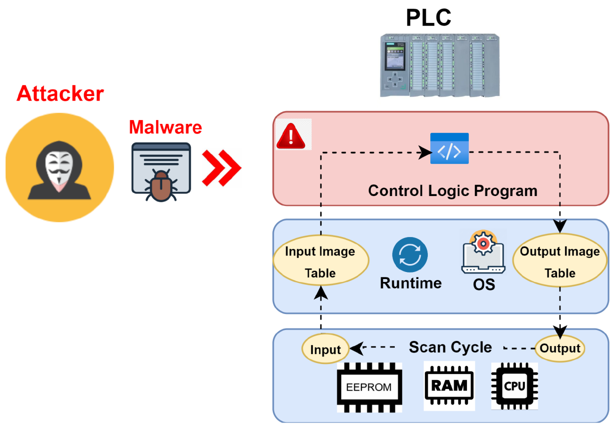

FIGURE 9. Payload attack.

5) TIME-DELAY INJECTION ATTACK

This type of attack primarily focuses on exploiting weaknesses within the communication links of the targeted system, resulting in the loss of critical information and potentially leading to unstable operating conditions [95]. In a PLC-based system, delays in packet transmission across the control network can cause a degradation in system performance and stability. Larsen [96] termed this attack as a "stale data attack," where the attacker manipulates the timing of encrypted packets on the associated network to create a discrepancy between the physical and logical states of the process. As a consequence, the PLC may be forced into an arbitrary state. To illustrate, Lou et al. [97] conducted a time-delay injection attack on a power plant control system, successfully introducing delays in the transfer of control commands over the control network. Similarly, Korkmaz et al. [94] executed a successful time-delay injection attack on a SCADA system. They first exploited a network vulnerability and then employed a traffic shaping tool to intentionally introduce random delays within the targeted control network.

B. ATTACKS AGAINST INTEGRITY

In this category, we elaborate five different attack scenarios as follows: payload, injection, I/O pin control, control flow, and configuration modification attack. The following subsections provide a detailed explanation of each attack scenario.

1) PAYLOAD ATTACK

PLCs interact with various hardware components and offer firmware support for executing control logic programs, often referred to as "payload" programs. Lately, there has been a growing interest among attackers in targeting these payload programs (see Fig. 9).

The reason behind this is that they can directly inject harmful payload programs into PLCs. Once they acquire the necessary privileges, attackers can even modify alerts related to payload changes, effectively concealing their actions [62]. Consequently, engineers are unable to identify malicious payload programs through real-time integrity inspection using

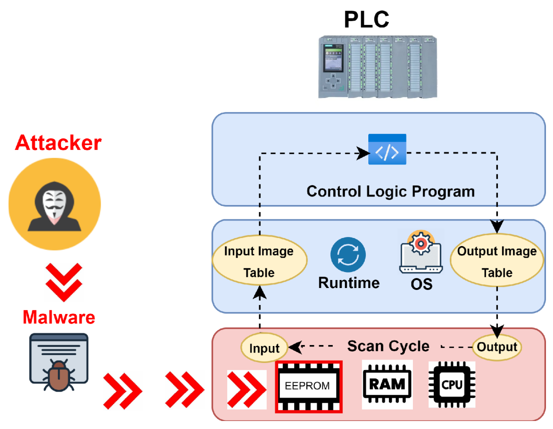

FIGURE 10. **Control flow attack.**

specific-vendor PLC software. As a result, such attacks can be seen as alterations to control logic or the direct insertion of malicious code into the PLC system. To explore the capacity of generating malevolent payloads, McLaughlin et al. [63] delved into the challenge of developing PLC malware capable of creating dynamic payloads based on insights gained from observing processes within the control system. The researchers successfully crafted a dynamic payload that triggered unsafe behaviors within the PLC. In a subsequent study, McLaughlin and McDaniel [64] introduced a tool called SABOT. This tool automatically maps control instructions within a PLC to a specification of the desired behavior of the target control system, provided by an adversary. This mapping process reconstructs enough internal layout semantics of the PLC to instantiate arbitrary malicious PLC code.

2) CONTROL FLOW ATTACK

This class of attacks involves exploiting a vulnerability related to memory corruption, like buffer overflow. This vulnerability allows attackers to execute code of their choice (see Fig. 10).

Initially, they evade critical security checks such as secure boot or authentication methods. Once that is done, they proceed to execute malicious code segments without any conditions. Serhane et al. [41] demonstrated how attackers can compromise functions, manipulating specific values of operands, or introducing empty branches and jumps. They discussed the potential of using infinite loops through jumps and utilizing timers to trigger a branch with malicious instructions at a time chosen by the attacker. Valentine [43] outlined an attack scenario where an adversary inserts a jump to a subroutine function and alters communication between two or more ladder components in LD code. The author illustrated that an attacker who gains access to the engineering station could implant their malicious code at a wrongly labeled location. This would lead to multiple errors before the code reaches its intended destination upon the return command. Beresford [74] identified various protocol vulnerabilities in Siemens PLCs that could be exploited by an adversary to execute remote code attacks. Schuster et al. [156] conducted experiments revealing an attacker's ability to elude detection methods for control flow attacks by manipulating executable module code sequences within the target program. Davi et al. [157] introduced several techniques for control flow attacks that can bypass conventional detection mechanisms. Specifically, they demonstrated that attackers can capitalize on vulnerabilities in a program's binary code to create extensive chains of gadgets. This approach effectively undermines detection mechanisms designed to counter control flow attacks.

3) INJECTION ATTACK

The majority of PLCs tend to accept messages without restrictions when received over networks that use insecure communication protocols, such as S7Comm for Siemens S7-300 and S7-400 PLCs, and PCCC for Allen-Bradley MicroLogix 1400 PLCs. Moreover, some PLC vendors provide open-source function libraries that help in establishing TCP/UDP communications that eventually bring a significant risk of exploitation by malicious attackers. This creates a scenario where unauthorized individuals who gain access to the control network can insert harmful data or code into PLCs, thereby gaining complete control over the targeted PLC(s) and causing severe disruptions to the entire control system. A good example of such an attack is the "denial of engineering operations" (DEO) [101]. The authors introduced three attack injection scenarios. The first two attack scenarios utilized a MitM approach to manipulate network traffic during attempts to retrieve control logic from an infected PLC. In the first attack, the adversary removed the infected code from the packets to conceal the infection, while in the second attack, the attacker replaced specific control logic instructions in the packets with irrelevant data. In the third attack, the attacker crafted a malicious control logic that could run on a PLC but caused the software to crash when attempting to obtain the control logic from the PLC. Alsabbagh and Langendörfer [38], [39], [48], [49], [50], [81], [82] presented various injection attack scenarios targeting SIMATIC S7 PLCs. They demonstrated the feasibility of exploiting vulnerabilities in S7Comm and S7CommPlus protocols, allowing for the modification of control logic programs in S7 PLCs from different families. Furthermore, the authors managed to hide their ongoing attacks from the ICS operators using various techniques, including employing fake PLCs, manipulating certain blocks, and crafting specific packets. The compromised PLCs caused unsafe states within the systems they targeted. Qasim et al. [66] introduced an automated framework called Similo for control logic forensics in ICSs. As part of their experiments, they conducted DEO attacks similar to those in [101] and managed to hide malicious control logic within Allen-Bradley MicroLogix 1100 and 1400 PLCs. Klick et al. [99] demonstrated that a knowledgeable adversary with access to a PLC, specifically S7-300 PLCs, could download and upload native code to it, provided that the code was composed of MC7 bytecode. The authors developed

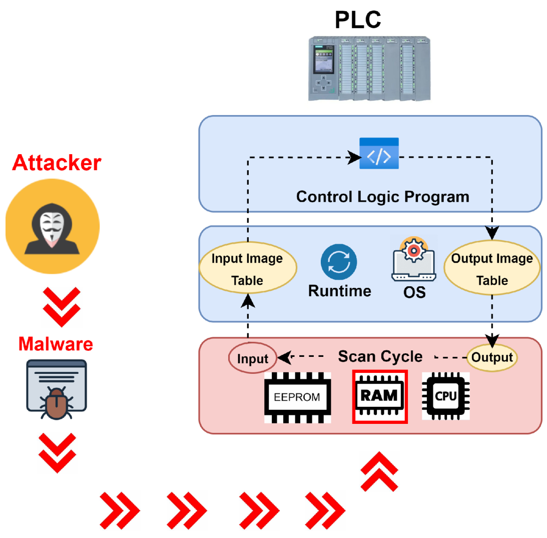

FIGURE 11. I/O pin control attack.

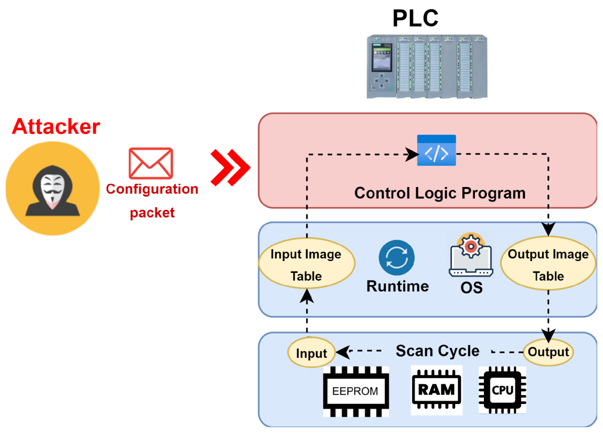

FIGURE 12. Configuration modification attack.

an attacking tool named PLCinject, designed to inject malicious MC7 bytecode into PLCs. Using PLCinject, Spenneberg et al. [100] introduced a worm called PLC-Blaster, capable of spreading from one PLC to another by copying itself and adapting the next targeted PLC to execute the worm alongside the active control logic program. The worm was designed to execute appropriate code at the start of each execution cycle and incorporated various antidetection mechanisms to intensify its impact. Another form of injection attack pertains to false data. McLaughlin and Zonouz [68] performed an injection attack, precisely a false-data scenario against individual PLCs. Their attack tool first analyzed I/O traces to internally build a logic model of the target devices and then performed a sequence of false input data to achieve desired outputs. Another work [69] introduced a false-data attack that involved constructing a certain model by collecting a so-called fault-free I/O traces. The achieved model was then used to generate false sequences for injecting into exploitable sensors. Fritz et al. [70] used Petri nets (PNs) to model a false-data attack in order to maliciously change sensor measurements in a discreet manner to alter state variables. Employing stealthy techniques for injection attacks, Yoo and Ahmed [71] suggested a false-data attack through two scenarios: fragmentation and noise padding attack. Both the scenarios aimed at using certain network packets to manipulate the logic of the target PLC.

4) I/O PIN CONTROL ATTACK

Fig. 11 represents a high overview of this attack scenario. The control of pins within embedded devices is governed by specific electrical logic that corresponds to unique physical addresses known as registers. To illustrate, the "Input Enabled" logic designates a pin's function as an input. In the context of PLCs, their logical registers are also linked to mapped registers, all of which are overseen by the OS. This process of managing the mapped registers through software is referred to as pin control. From a security perspective, malicious attackers have the potential to carry out I/O pin control attacks, compromising the legitimacy of authorized operations and manipulating interactions with the physical environment. An illustrative instance of such attacks was presented by Abbasi et al. [73]. The authors assumed that they had obtained root access and necessary preliminary knowledge. They devised an I/O manipulation attack with a discreet approach. Their method involved using the debug registers of the PLC to interrupt specific packets, e.g., read and write. Notably, they emphasized that their attack would go undetected by the PLC's runtime software. Building upon prior investigations, Abbasi et al. [72] further demonstrated in a subsequent study how an attacker could disrupt the integrity and accessibility of PLCs' I/O by exploiting specific pin control operations. This attack allowed the perpetrator to gain control over the physical processes typically managed by the PLC, all while evading detection by both the PLC runtime and monitoring personnel such as HMI operators. Importantly, this approach did not necessitate altering the PLC control logic as suggested in previous works [63], [64], which would typically be supervised by a host-based IDS.

5) CONFIGURATION MODIFICATION ATTACK

These attacks enable an adversary to change crucial settings or files of a PLC, such as control logic and network communication setups, in order to manipulate the process being controlled. Fig. 12 shows this attack scenario.

These malicious actions can be executed through network communication or interactions with hardware and software components. For instance, in PLCs using protocols like DNP3 or Profinet, configuration parameters play a key role. These parameters determine whether the PLC operates as a slave or a master, the allocation of protocol addresses, and the data points involved, including the extent of communication with additional slave or master devices. If an adversary gains control over a workstation within a CI network and gains access to configuration files related to the target PLC, he can modify these files and replaces them with altered versions.

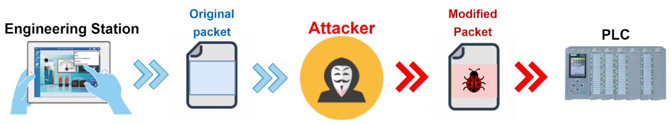

FIGURE 13. Replay attack.

This manipulation grants the attacker control over the PLC, potentially resulting in harmful activities affecting the entire system. Furthermore, the attacker can load his customized configuration onto both master and slave PLCs. Depending on the device, configuration files might be uploaded via Ethernet or serial connections. Sometimes, the attacker can even upload the original configuration previously installed by an engineer. This scenario allows adversaries to analyze the uploaded configuration, make modifications, and subsequently upload the altered configuration back into the control process. A good example of such an attack scenario is Stuxnet. Falliere et al. [2] reported that the Stuxnet malware was utilized to tamper with the programming of PLCs, specifically impacting a portion of the uranium enrichment process.

C. ATTACKS AGAINST CONFIDENTIALITY

In this group, we categorize five types of attacks: MitM, replay, eavesdropping, bypass, and brute-force attack. In the following, we provide a more elaborate depiction of each attack scenario.

1) MITM ATTACK

In MitM attacks, malicious adversaries position themselves between two devices, e.g., PLC and PLC, PLC and HMI, etc., or between devices and control centers. These attacks exploit the lack of encryption and or authentication mechanism in the existing communication protocols, allowing the two ends of the communication to interact legitimately. More concerning, the attacker can manipulate messages to compromise data confidentiality. The Address Resolution Protocol (ARP) poisoning technique is commonly employed in MitM attacks. This method involves associating the victim IP addresses with attacker's MAC address in the ARP tables. The MitM approach was used against different S7 PLC families [38], [39], [45], [48], [49], [50], [74], [76], [77], [81], [84], [85], [86], [88], [102], intercepting all data packets exchanged between the engineering station and the PLC. In another instance, Lim et al. [103] utilized an MitM scenario in order to exploit TRICON PLCs. The authors interrupted and altered certain configuration setting packets, rerouting the traffic through the victim controllers. Furthermore, Grandgenett et al. [102] developed an MitM attack to selectively manipulate Common Industrial Protocol data and commands transmitted between PLCs (RSLogix 5000) and Web server (EtherNet/IP) module.

2) REPLAY ATTACK

This kind of attacks involves taking advantage of a system's operation by resending certain valid messages (see Fig. 13). These messages are typically contained within packets that are recorded by attackers from a prior communication session using an MitM approach. The goal of this attack is first to bypass successfully any authentication mechanism applied, even if attackers do not fully understand the communication protocols. Beresford [74] was among the first researchers who demonstrated a replay attack against S7 PLCs (S7-1200). His work sparked interest among researchers focused on PLC security [38], [45], [77], [81], [83], [89]. Following vulnerabilities in the S7Comm protocol, a new protocol called S7CommPlus was developed with a built-in protection mechanism against replay attacks. However, several research works [49], [50], [82], [84], [85], [86], [88] revealed that the S7CommPlus protocol still have exploitable weaknesses. Researchers managed to break the encryption algorithms, allowing them to successfully carry out replay attacks against S7 PLCs.

3) EAVESDROPPING ATTACK

An eavesdropping attack takes place when an attacker intercepts, deletes, or alters data being transmitted between two or more endpoints within a system. This technique, also referred to as sniffing or snooping, exploits insecure network communication to gain unauthorized access to data in transit among devices. In a study by Ayub et al. [87], an eavesdropping attack was executed as part of their investigations into the authentication methods employed by different PLCs. The researchers demonstrated that attackers who have access to the control network can not only read but also manipulate any messages being exchanged over the network. Similarly, Alsabbagh et al. [75] intercepted packets exchanged between a PLC and HMI, modifying these packets to hide their ongoing attack from the operator of the ICS. In another instance, Sushma et al. [83] substituted control packets in the network with crafted packets as a component of their control logic injection attack. Consequently, the ICS operator was deceived into receiving a false control logic program, while the PLC executed a modified program. Hui et al. [86], [88] also adopted the eavesdropping attack strategy to pilfer an S7 session from the ICS operator and successfully establish communication with the targeted S7 PLC. The researchers deliberately dropped specific S7 packets from the network, preventing the intended destination from receiving accurate data. This facilitated the authors' ability to carry out more severe attacks against the targeted device.

4) RECONNAISSANCE ATTACK

Reconnaissance attacks have the primary objective of collecting information related to control systems and devices. These attacks involve activities such as mapping the network structure and identifying specific characteristics of the devices. These characteristics encompass details like the manufacturer, model number, network protocols, system addresses, memory layout, and so on. Previous studies, such as [39], [74], [81], and [99], have introduced specialized tools for scanning industrial environments. These tools are designed to extract vital information about PLCs and the systems they are associated with. For example, Alsabbagh and Langendörfer [81] introduced a tool called a PNIO scanner. This scanner is capable of exploring industrial networks and detecting any devices that support the Profinet protocol. The authors also devised an S7 scanner, specifically for compromised S7 PLCs, e.g., S7-300 and S7-400. This scanner gathers crucial data from these PLCs, including software and hardware blocks, block counts, program sizes, and more. In a similar vein, Beresford [74] conducted an eavesdropping attack using Wireshark (network protocol analyzer). By capturing specific packets, he managed to uncover important data, which he then leveraged for subsequent attacks, such as replay attacks. Another contribution comes from Klick et al. [99], who presented a scanner based on the Simple Network Management Protocol. This scanner focuses on revealing significant details about exposed S7 PLCs, including information like product type, model number, hardware and software firmware versions, and other pertinent data.

5) BRUTE-FORCE ATTACK

A brute-force attack involves attempting to deduce login details, credentials, and encryption keys by systematically trying out every conceivable combination of characters or numbers until the correct one is identified. This approach is primarily employed to uncover passwords for PLCs or HMIs, as shown in [38], [74], [81], [83], [87], and [89]. In a study by Alsabbagh and Langendörfer [81], a brute-force attack was employed to extract plaintext passwords securing S7-300 PLCs. The authors managed successfully to eliminate the password protection, denoted by setting the protection level to “0.” Consequently, this enabled the attacker to access the compromised PLC for reading and writing operations without undergoing any authentication procedure. In another instance, Ayub et al. [87] delved into the task of deciphering passwords from various PLC manufacturers. They achieved this by intercepting authentication packets exchanged within the network communication between PLCs and their respective engineering software. Subsequently, these researchers scrutinized the authentication algorithms employed by different PLCs. Employing brute-force techniques, they managed to uncover the actual passwords utilized in the tested PLCs. Furthermore, Ward et al. [89] intercepted authentication packets transmitted between S7-400 PLCs and TIA Portal software. This interception led to the revelation of the encoding mechanism used for password protection. Building upon this knowledge, the authors then executed various replay attacks, enabling them to manipulate the passwords associated with the evaluated PLCs.

V. SECURITY SOLUTIONS

In this section, we investigate existing security methods that are employed to safeguard PLC-based systems. It is important to emphasize that these methods have been sourced from prior research works. We categorize these security methods into six groups as follows: 1) PLC program detection; 2) PLC firmware detection; 3) PLC side-channel detection; 4) intrusion detection; 5) honeypot-based detection; and 6) digital forensics, as shown in Fig. 14. In the following, we provide a more comprehensive exploration of each category of these security approaches.

A. PLC PROGRAM DETECTION

The operational status of active PLCs ultimately impacts the overall state of the entire system. As a result, adversaries focus on compromising PLC control logic programs in order to directly inflict significant damage on the physical processes controlled by the compromised PLCs. Several research works have employed formal verification to ensure the safety and security of PLC programs, as shown in Table 3. In this article, we categorize the existing research efforts according to the following criteria.

1) BEHAVIOR MODELING

The objective of behavior modeling is to create a structured portrayal of the behavior exhibited by a PLC program. This depiction enables a formal verification framework to comprehend and confirm it when provided with a specification. In the following, we group behavior modeling into three tiers: program level, binary code level, and program runtime level.

a) PLC program level: At the level of software design, prior research works, such as [171], [172], [173], and [174], have delved into the structured representation of general program actions. Those studies mentioned transformed programs into automata [164] and PNs [165], as these forms were well supported by already existing formal verification frameworks [171]. These transformations typically viewed each component of the program as an automaton, encompassing the main program, functions, and instances of function blocks. Corresponding variables within the program unit were converted into variables in the automaton. Inputs were assigned in a nondeterministic manner at the outset of each cycle of the PLC. The entire program was depicted as an interconnected set of automata, wherein transitions captured alterations in variable values across different execution cycles, and synchronization pairs indicated coordinated transitions involving function calls. In a similar approach, Newell et al. [174] translated FBD programs into models using the prototype verification system (PVS), as certain nuclear power plants specifically supported this representation. Kottler et al. [113] dedicated their efforts to assessing the reliability of PLC programs written in LD and ST languages, aiming to identify specific vulnerabilities related to security. Hailesellasie and Hasan [114] introduced a methodology based on the attributed graph between both manipulated programs and original ones. Such graphs are basically produced through formal modeling based on the Uppsala Timed Automata Analyzer (UPPAAL) framework.

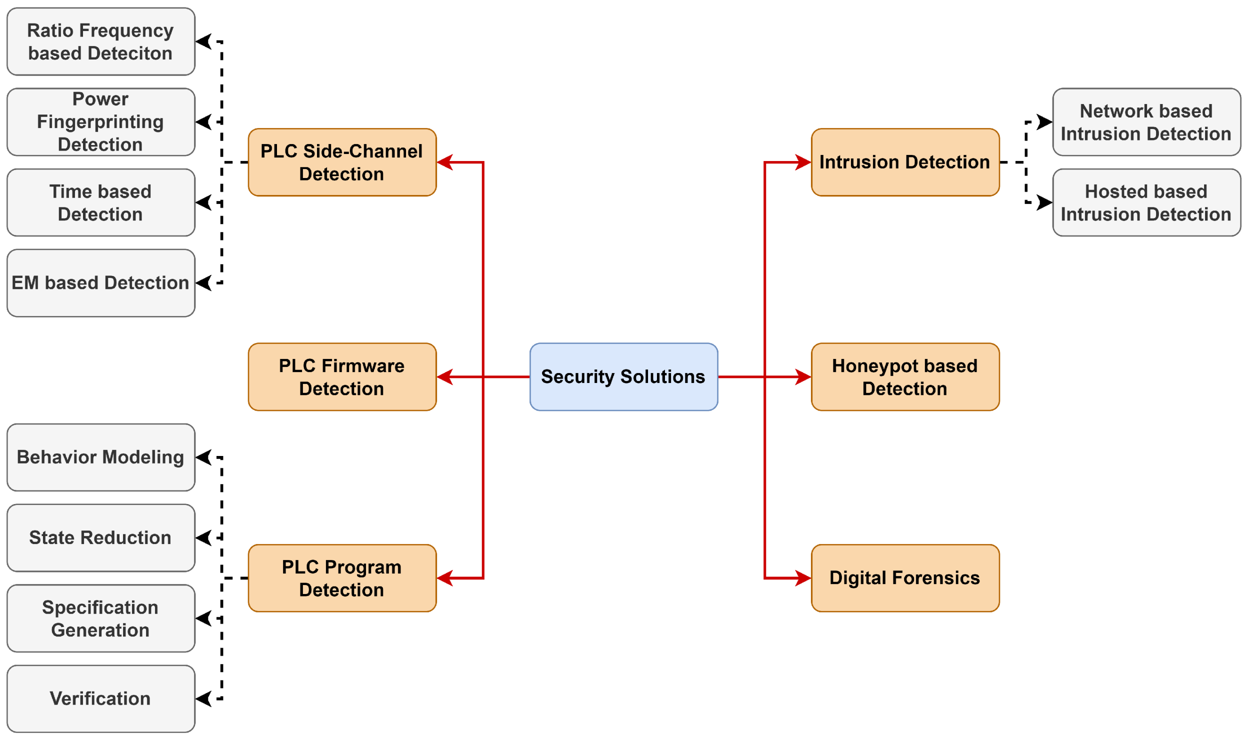

FIGURE 14. **Classification of existing security solutions for PLC-based systems.**

All the previously mentioned studies were successful in formally representing a wide range of PLC behaviors, particularly the internal logic embedded in the PLC code. However, due to the limited availability of only the source code, these behavior models lack integration with the physical PLC hardware and the real-world processes. This limitation increases the possibility of unsafe or malicious behaviors going unnoticed during subsequent formal verification processes.

(b) PLC binary program level: Only few works have discussed behavior modeling at the binary code level, particularly focusing on the challenges of reverse engineering. Previous studies like [110] and [111] highlighted that various features of PLCs are not fully supported by standard instruction sets. PLCs employ a hierarchical system to address input and output buffers, and use function blocks with fixed entry and exit points along with specialized timers that behave differently for bit/logic and arithmetic instructions. Several researchers [110], [111], [112], [175] have delved into the modeling of Siemens binary programs. For instance, McLaughlin et al. [110] utilized an instruction list (IL) derived from the statement list (STL) program to enhance instruction modeling comprehensiveness. They employed a through-silicon via (TSV) tool to extract information flow from PLC registers and memory. By executing multiple scan cycles, they constructed a temporal execution graph to represent controller code states. Zonouz et al. [111] adopted a similar modeling approach, while Chang et al. [112] developed control flow graphs (CFGs) highlighting executable paths. These authors inferred timer output states based on existing output state transition relationships. Xie et al. [175] took a constraint-based approach for program modeling. Lv et al. [115] introduced a decompiler suited for control logic programs, utilizing instruction or operand templates. Similarly, Keliris and Maniatakos [116] introduced a reverse engineering framework designed to deal with PLC machine codes that are compiled with CODESYS. Another work [117] decompiled programs from machine codes into STL code and subsequently constructed a CFG to establish input–output mappings. Abbasi et al. [118] performed a so-called embedded control flow integrity approach to address binary program concerns.

C) PLC PROGRAM RUNTIME LEVEL

Utilizing real-time data, prior research works [107], [176], [177], [178], [179] focused on the conceptualization of software programs in conjunction with their interactions within physical processes, supervisory systems, and operator tasks. This approach facilitated the creation of more lifelike models capable of accommodating time-sensitive instructions and domain-specific behavioral characteristics. Several automated frameworks [107], [180] were introduced to capture PLC behaviors encompassing interrupt scheduling, function invocations, and I/O traces. In a study by Zhou et al. [179], an environment module was integrated to handle inputs and outputs, an interruption module was incorporated for time-dependent instructions, and a coordinator module was devised to orchestrate these two modules alongside the main program logic.

TABLE 3. Existing Works Related to PLC Program Detection

Paper	Defense Focus	Security Level	Verification Techniques	Property	PLC Language	Tools
[173]	BM, SG	PLC Program	MC	CTL, LTL	ST, SFC	nuXmv, PLCVerif, Xtext, UNICONS
[199]	FV	PLC Program	MC	CTL	SFC	Cadence SMV, UPPAAL
[194]	SG, FV	PLC Program	MC	seLTL	LD	Tina Toolkit
[197]	SG, FV	PLC Program	MC	CTL, ptLTL	generic	PLCopen, Arcade.PLC, CEGAR
[189]	SG	PLC Program	TP	N/A	IL	SSReflect in Coq, CompCert
[195]	SG	PLC Program	MC	N/A	SFC	SPIN/Promela, UPPAAL
[185]	SR	PLC Program	MC	CTL, LTL	ST	COI reduction, NuSMV
[190]	SG	PLC Program	EC	N/A	ST	PLCspecif
[191]	SG, FV	PLC Program	N/A	Temporal Logic	ST	PLCverif, nuXmv, UPPAAL
[186]	SR	PLC Program	MC, EC	Temporal Logic	LD, FBD	PLCverif, NuSMV, nuXmv
[174]	BM	PLC Program	N/A	Temporal Logic	IL	PLCverif, Xtext parser
[175]	BM, SG	PLC Program	EC	N/A	ST	GROOVE, ISABELLE, FUJABA
[187]	SR	PLC Program	MC	N/A	ST, LD, IL	NuSMV
[205]	FV	PLC Program	MC	N/A	SFC, ST, IL	BIP, nuXmv, UPPAAL, UBIS
[192]	SG	PLC Program	N/A	N/A	ST	KST
[206]	FV	PLC Program	MC, EC	CTL	FBD, LD	NUDE 2.0, NuSCR
[196]	SG	PLC Program	MC	CTL	LD	N/A
[176]	BM, SR	PLC Program	TP	N/A	FBD	PVS theorem
[202]	FV	PLC Program	MC	N/A	generic	UPPAAL, Program translators
[188]	SR	PLC Program	MC	CTL	FBD	NuSMV
[193]	SG, FV	PLC Program	MC	CTL, ACTL	ST	st2smv, SynthSMV
[210]	BM	PLC Program	N/A	N/A	generic	N/A
[211]	BM, FV	PLC Program	MC	N/A	generic	N/A
[212]	SR	PLC Program	MC	N/A	ST	Z3, PLCopen, Arcade.PLC
[213]	BM	PLC Program	N/A	LTL	ST	Cadence SMV
[214]	BM	PLC Program	N/A	CTL	generic	SMV, CASE tools
[201]	BM, SG	PLC Program	TP	FOL	LD	Swansea
[204]	ALL	PLC Program	MC, CE	CTL	FBD	NuSCR, Cadence SMV, VIS, CASE
[115]	ALL	PLC Program	N/A	CTL	LD	NuSMV
[200]	ALL	PLC Program	MC	LTL	IL	Cadence SMV
[114]	ALL	Binary Program	MC	LTL, CTL	IL	DotNetSiemensPLCToolBoxLibrary
[112]	ALL	Binary Program	MC	LTL	IL	TSV, Z3, NuSMV
[177]	BM, SG, FV	Binary Program	MC	LTL	IL	SMT, NuXMV
[113]	BM, SG, FV	Binary Program	MC	LTL	IL	Z3, NuSMV
[207]	FV	Program Runtime	MC	CTL, LTL	N/A	NuSMV
[178]	BM	Program Runtime	MC	CTL	FBD	Supremica
[198]	SG	Program Runtime	MC	CTL	FBD	ViVe/SWSA
[209]	FV	Program Runtime	MC	DFA	LD, ST	N/A
[179]	BM, SR	Program Runtime	MC	ITL	LD	Tempura
[180]	BM, SR, SG	Program Runtime	TP	CLIMB	N/A	SCIFF checker
[184]	BM, FV, SG	Program Runtime	MC	TCTL	LD, FBD	UPPAAL
[144]	BM, SR, SG	Program Runtime	MC	LTL, MTL	IL	BIP
[164]	ALL	Program Runtime	MC	TPTL	ST	BULDTSEQS algorithm
[182]	BM, SR	Program Runtime	MC	TCTL	TCTL	UPPAAL
[215]	BM, FV	Program Runtime	TP	Gallina	LD	Coq, Vernacular
[208]	BM	Program Runtime	TP	differential dl	ST	KeYmaera X
[216]	BM	Program Runtime	MC	TCTL	LD	UPPAAL
[217]	BM	Program Runtime	N/A	eFSA	N/A	LLVM DG

(BM) Behavior Modeling; (SR) State Reduction; (SG) Specification Generation; (FV) Formal Verification; (MC) Model Checking, (EC) Equivalence Checking; (TP) Theorem Proving.

Wang et al. [180] automated a framework known as behavior interaction priority (BIP), which formalized the scanning mode, interrupt scheduling, and function calls within the PLC. Another work [180] presented a component-based approach to model the entire control command sequence, with each component being described as a timed automaton. For the automation of domain-specific event behaviors, VetPLC [107] generated timed event causality graphs (TECGs) derived from the program itself and the dynamic runtime data traces. The TECG maintained temporal dependencies constrained by machine operations. These investigations effectively eliminated obstacles associated with modeling event-driven and domain-specific behaviors. Furthermore, these methodologies proved capable of mitigating security and safety breaches by identifying and countering anomalous logic sequences.

2) STATE REDUCTION

The aim of state reduction is to enhance the scalability and intricacy of formalizing PLC programs. This process comprises two primary phases. Initially, it is crucial to identify significant states associated with safety and security attributes. Subsequently, we categorize behavioral modeling into three tiers: program level, binary code level, and program runtime level.

A) PLC PROGRAM LEVEL

At the level of source code, prior works like [182], [183], [184], and [185] focused on state reduction techniques. Gourcuff et al. [184] emphasized meaningful states as those linked to input and output variables, which directly govern the behavior of physical processes. They employed static code analysis to establish dependency relations among these variables within an ST program. This analysis revealed numerous irrelevant states. However, this method, while considerably reducing the search space for states, omitted portions of the original code necessary for subsequent verification. To enhance the formalization's code coverage, Pavlovic and Ehrich [185] proposed a comprehensive solution tailored for FBD programs. Their approach involved transforming graphical programs into textual statements in textFBD and then substituting circuit variables with tFBD. This procedure eliminated redundant assignments connecting continuous statements and consolidated them. Building upon this approach, Darvas et al. [182], [183] fine-tuned reduction heuristics by providing a more comprehensive representation. In addition to eliminating unnecessary variables and logic, these heuristics integrated cone of influence (COI)-based reduction and rule-based reduction. The COI-based reduction initially removed unconditional states that all possible executions passed through. Subsequently, it eliminated variables with no impact on specification evaluation. The rule-based reduction was changeable based on the safety requirements of the application domain. Moreover, mathematical models were employed to abstract distinct components. Newell et al. [174] introduced supplementary structures, such as attribute maps, graphs, and block groups, to diminish the state space of their PVS code.

These previous efforts effectively minimized program state sizes. However, they were confined to rudimentary Boolean representation reduction. For programs featuring intricate time-related variables, function blocks, or multitasking elements, these studies fell short. Furthermore, it remained unclear whether these reduction techniques could compromise program security.

B) PLC BINARY PROGRAM LEVEL

Research focusing on the binary level has primarily utilized a combination of symbolic execution and flow-based representation. This approach has illustrated that significant states generate distinct symbolic output vectors. The TSV [110] method consolidated input states that could potentially result in identical output values. In addition, it abstracted temporal execution graphs by eliminating symbolic variables in relation to their alignment with the valuations of linear temporal logic (LTL) properties. To enhance the elimination of irrelevant states, Chang et al. [117] minimized the overlap among output states within the same scan cycle. They also discarded output states that had undergone analysis in prior cycles. In order to streamline timer modeling overhead, the authors employed a deduction technique for timer output states. This technique involved scrutinizing existing relationships governing output state transitions. Importantly, these reduction strategies did not compromise the primary objective of detecting malicious behaviors spanning multiple cycles.

C) PLC PROGRAM RUNTIME LEVEL

By utilizing runtime information, a more comprehensive grasp of genuinely significant conditions can be attained. These encompass insights derived from event arrangement concerning subroutines and interrupts, along with the authentic inputs and outputs originating from processes specific to the given domain. Prior papers such as [107], [177], [178], and [179] have showcased diverse strategies for condensing states. To minimize the model's scope, Zhou et al. [179] integrated timers directly into the primary program instead of employing a distinct automaton. This choice was guided by their incorporation of genuine environmental traces, interruptions, and the coordination between them within their model. In a parallel vein, Wang et al. [142] merged segments devoid of jump and call instructions into singular transitions. Furthermore, alongside consolidating extraneous states, incorporating actual inputs and domain-focused insights can narrow down the scope when modeling numerical and floating-point variables. In Zhang et al.'s methodology [107], continuous timing behaviors were discretized into multiple time slices, each with a consistent interval. Given the accessibility of application-specific I/O traces, the time intervals were refined within a range that strikes a balance between efficiency and precision.

3) SPECIFICATION GENERATION

The objective of this research here is to formulate precise safety and security specifications using formal semantics. Prior investigations have concentrated on two key areas: 1) process-independent attributes that outline the fundamental prerequisites for a control system based on PLCs and 2) domain-specific attributes that necessitate expertise within a particular field.

A) PLC PROGRAM LEVEL

In the PLC programming theme, several research works, such as [173], [186], [187], [188], and [189], have delved into the realm of specification generation through the lens of process-independent attributes. These attributes encompass elements like the avoidance of variable locks, prevention of reaching unreachable operating modes, establishment of mutually exclusive operating modes, and the elimination of inconsequential logic [190]. Prior investigations, like [171], [190], [191], [192], and [193], adopted the utilization of formulas based on computation tree logic (CTL) or LTL to articulate these attributes. LTL pertains to the future progression of pathways, signifying conditions that will eventually hold true or conditions that will endure until another fact becomes valid. In contrast, CTL is concerned with variance and reachability, encompassing concepts such as the perpetual confinement within a set of states or the capability to attain a specific set of states. Variations in this approach include the application of the universal fragment of CTL (ACTL), as demonstrated by Rawlings et al. [190], and the adoption of past time linear temporal logic (ptLTL), as explored by Biallas et al. [194]. Apart from employing CTL- and LTL-based formulas, an investigation to facilitate the formal development of proofs was taken. For the meticulous delineation of syntax and semantics, Mesli-Kesraoui et al. [181] harnessed a proof assistant rooted in type theory—Coq. This was instrumental in defining safety properties for IL programs. The focus of semantics was centered around the rigorous formalization of on-delay timers, utilizing discrete time with fixed intervals. Alongside Coq, the KST framework [189] was embraced, offering a formal operational semantics platform for ST programs. KST operates on a rewriting-based semantic framework, previously employed to define semantics for programming languages such as C and Java. In comparison to Coq, KST offers a less stringent formality, resulting in a more accessible and comprehensible framework. However, this is balanced by the necessity for manual oversight to maintain the formal integrity of definitions. It is noteworthy that prior studies confined the process of specification generation to specific program models. Addressing this limitation, Darvas et al. [187] introduced PLCspecif, a solution that enables formal semantics for a variety of program models, encompassing state-based, data-flow-oriented, and time-dependent paradigms. These various works have opened up avenues for engineers without a deep-seated formalism background to generate precise and formal requirements. The inclusion of proof assistant frameworks even facilitated the generation of directly executable programs, such as those in the C programming language. Nevertheless, the automation of these processes was primarily confined to process-independent attributes. The subsequent discourse delves into the exploration of specification generation incorporating a more comprehensive array of information.

B) PLC BINARY PROGRAM LEVEL

As previously noted, the utilization of symbolic execution in the studies facilitated the incorporation of numeric and floating-point variables into program modeling. This inclusion of variables expanded the capacity for defining properties within the specifications. For instance, in the work by TSV [110], properties were established to limit the ranges of numerical device parameters, encompassing factors like maximum drive velocity and acceleration. Similarly, other researchers [111], [112], [175] formulated properties geared toward identifying instances of malicious code injection and parameter tampering attacks. Notably, Xie et al. [175] extended these properties even further to encompass the detection of subtle attacks such as stealthy incursions and DoS attacks.

C) PLC PROGRAM RUNTIME LEVEL

Utilizing real-time operational data, the focus of specification generation shifted toward domain-specific attributes. Within the context of a wastewater treatment plant, Luccarini et al. [178] employed artificial neural networks to distill qualitative patterns from continuous signals in the water, including metrics like pH and dissolved oxygen levels. These qualitative patterns were subsequently correlated with control events within the physical processes. This mapping was recorded using XML and then translated into formal rules to define specifications. This approach leveraged the captured input and output traces as reliable indicators of security and safety properties, diminishing the reliance on domain expertise. However, it is worth noting that actual runtime traces could potentially be contaminated or lack complete properties suitable for verification. To ensure the accuracy and comprehensiveness of domain-specific rules, previous research works [162], [195] also explored semiautomated methodologies that combined automated data mining with manual domain expertise. VetPLC [162] established safety properties through a combination of automatic data mining and event extraction, complemented by domain knowledge in formulating safety specifications. VetPLC adopted timed propositional temporal logic (TPTL), a more suitable framework for quantitatively expressing safety specifications. Apart from semiautomated approaches to specification generation, Mesli-Kesraoui et al. [181] manually established a set of rules governing interactions among each component along the control chain. The requirements were formulated using CTL temporal logic. To aid domain experts in devising formal rules, Wang et al. [142] formalized the semantics of a BIP model encompassing various types of PLC programs. This automated the generation of process-independent rules for interrupts, such as adhering to the first-come first-served principle.

Overall, these studies facilitated the creation of specifications by integrating domain-specific expertise. As a result, they extended the realm of security research, with a heightened emphasis on safety prerequisites.

4) VERIFICATION

Various studies such as [190], [191], [192], [193], [196], [197], [198], [199], [200], and [201] employed model checking and theorem proving to establish the safety and security of software programs. Each of the aforementioned studies has engaged multiple formal verification frameworks. Notably, a predominant choice among these frameworks has been the utilization of *UPPAAL* and Cadence *SMV*. *UPPAAL* has been primarily employed for real-time verification, adeptly representing intricate networks of timed automata, further extended to encompass integer variables, structured data types, and channel synchronization. On the other hand, Cadence *SMV* has been favored for nontimed verification tasks. In the subsequent sections, we delve into the specifics of each of these studies, categorized according to our established criteria.

A) PLC PROGRAM LEVEL

At the program level, formal verification investigations are conducted to detect vulnerabilities and enhance defenses against broad safety issues. This approach has been employed across various industries. For instance, Bender et al. [191] utilized model checking to assess LD programs represented as timed PNs. Using the Tina toolkit's model checkers, they verified LTL properties. Bauer et al. [196] opted for Cadence SMV and UPPAAL to, respectively, validate non-timed and timed models of SFC programs, successfully identifying errors across three reactors. Similarly, Niang et al. [199] employed UPPAAL to verify an SFC-based circuit breaker program. Hailesellasie and Hasan [114] employed UPPAAL, comparing two formally generated attribute graphs: 1) the Golden Model with properties and 2) a random model formalized from a PLC program. Their verification process centered around node and edge comparisons within the graphs, revealing instances of covert code injections.

In addition to utilizing existing tools, certain studies designed their own frameworks for verification. For example, Arcade.PLC [194] introduced support for model checking with CTL- and LTL-based properties applicable to various types of PLC programs. PLCverif [188] accommodated programs from all five Siemens PLC languages. NuDE 2.0 [202] provided a formal method-based approach for software development, verification, and safety analysis within nuclear industries. Rawlings et al. [190] employed symbolic model checking tools, specifically st2smv and SynthSMV, to validate and invalidate an ST program controlling batch reactor systems. This process automatically confirmed process-independent properties. Beyond model checking, certain studies [174] also incorporated PVS theorem proving to verify safety properties as described in tabular expressions for a railway interlocking system. However, it is important to note that these investigations primarily focus on verifying general safety requirements.

B) PLC BINARY PROGRAM LEVEL

Many research works [110], [111], [112], [175], [180] have contributed to the identification of binary injection attacks. TSV [110] adopted a hybrid approach, combining symbolic execution and model checking. It supplied the model checker with an abstracted temporal execution graph, along with a manually formulated safety property using LTL. However, TSV encountered limitations due to its compatibility with time-related operations within a single cycle, leading to challenges related to code verification and state explosion. To address these limitations, Xie et al. [175] employed constraints to validate random input signals and mitigate the aforementioned issues. They leveraged the nuXmv model checker for this purpose. Alternatively, Chang et al. [117] pursued a less formal verification methodology based on state count. These investigations effectively uncovered instances of malicious parameter tampering attacks. These instances were demonstrated through sample programs controlling diverse systems such as traffic lights, elevators, water tanks, stirrers, and sewage injectors.

C) PLC PROGRAM RUNTIME LEVEL

Using real-time data, prior research has been able to confirm safety and security concerns specific to certain domains. In the work by Carlsson et al. [203], NuSMV was employed to validate the interaction between the Open Platform Communications (OPC) interface and the program. This was achieved by defining properties as server/client states. Various issues such as synchronization problems—including delays, race conditions, and slow sampling arising from the OPC interface—were identified. In a similar vein, Mesli-Kesraoui et al. [181] utilized UPPAAL to analyze multilayer timed automata. They established a collection of safety and usability properties written in CTL and pinpointed synchronization errors between control programs and supervision interfaces. Incorporating insights from physical processes, VetPLC [162] amalgamated runtime traces and employed BUILDTSEQS to validate security properties expressed in TPTL. Contrasting VetPLC's approach, HyPLC [204] adopted the KeYmaera theorem to validate properties defined in differential dynamic logic. However, HyPLC pursued a two-way validation between physical processes and the PLC program, aiming to detect safety breaches.

The previously mentioned studies were inclined toward either offline verification or provided vague references to employing a supervisory component for online verification. To introduce an online verification framework, Garcia et al. [205] introduced an on-device real-time solution designed to detect control logic corruption. They capitalized on an embedded hypervisor within the PLC, endowed with enhanced computational capabilities and direct library function call integration. The hypervisor effectively addressed challenges related to stringent timing requirements and limited resources, enabling enforcement of verification within each scan cycle.

B. PLC FIRMWARE DETECTION

The firmware of a PLC serves as a bridge connecting the software to the hardware components. Many studies revealed that PLCs lack the ability to undergo firmware audits, as discussed in Section III-A2. In cases where the firmware is exploited by adversaries, they can eventually gain control over other physical components of the system through the compromised PLCs. Therefore, the detection of firmware modifications holds paramount significance in any control system based on PLCs. McMinn and Butts [119] introduced a tool for verifying firmware authenticity during an ongoing serial data upload process. This tool can be deployed across diverse platforms without necessitating alterations in the configurations of PLCs or their related systems. Furthermore, Basnight et al. [51] proposed an approach to deducing the validation of firmware updates. Through the utilization of reverse engineering techniques, they could uncover inconspicuous modifications to firmware, such as the HARVEY attack. Furthermore, Garcia et al. [54] offered insights into detecting firmware attacks. They put forth a method enabling the assessment of PLC firmware integrity and the monitoring of data exchanges in both directions: from input devices to PLCs and from PLCs to output devices.

C. PLC SIDE-CHANNEL DETECTION

In prior research, it has been demonstrated that unintended information can be exposed from PLCs. This exposure includes various forms like radio frequency (RF) emissions, power consumption patterns, electromagnetic (EM) emanations, and the duration of operations. These unintentional disclosures are usually gathered through physical measurements obtained from what is known as a side channel. Consequently, the analysis of side channels has emerged as a widespread approach for uncovering malicious attacks or unintended activities affecting PLCs. This article studies the current methods employed for detecting and understanding the effects stemming from side channels.

1) RATIO-FREQUENCY-BASED DETECTION

In 2012, Stone and Temple [120] introduced an RF-based approach to identify anomalous operations within PLCs. Subsequently, the authors enhanced the anomaly detection capabilities by employing sequences of unintentional time-domain emissions from PLCs, transformed using the Hilbert transform [121]. Notably, this RF-based analysis scheme operates independently of network-based cyberattacks, relying solely on physical layer information from the isolated PLCs.

2) POWER FINGERPRINTING DETECTION

Gonzalez and Hinton [122] devised a method for monitoring PLCs and detecting malicious software execution using power fingerprinting. However, real-time monitoring posed challenges due to the need for sensors closely interacting with PLCs. To address this, Xiao et al. [123] proposed a noninvasive real-time detection method that utilizes a resistor to collect power consumption traces, alleviating the burden on PLCs caused by sensors and high-frequency data acquisition.

3) TIME-BASED DETECTION

The aforementioned techniques, such as RF-based detection and power fingerprinting detection, necessitate additional hardware integration into PLC-based systems, which can make the proposed methods intricate. Alternatively, some researchers have focused on simpler detection mechanisms, like utilizing timing-based side channels. For instance, Dunlap et al. [124] introduced a method for detecting unauthorized PLC manipulation based on measuring the execution time. McDonald and Mueller [170] presented a monitoring technique for intrusion detection that involves inserting time checks along code paths. Their approach offers comprehensive protection throughout the entire execution path.

4) EM EMANATION-BASED DETECTION

A research group [125] investigated the possibility of using EM techniques for detecting code execution in PLC systems. The authors employed a signal cliff detection method, which allowed them to monitor regular and irregular activities independently. Likewise, another group led by Van Aubel et al. [126] leveraged EM side-channel measurements to spot alterations in behavior during the execution of industrial software. Their proposed approach involved a two-tier verification process: the initial tier evaluated the runtime of the user program, while the subsequent tier involved a comparison of its EM trace with a baseline version.

D. INTRUSION DETECTION

Given the presence of vulnerable industrial components and unsecured communication protocols, PLC-based systems face a broad spectrum of potential cyber threats. However, intrusion detection stands out as a prevalent remedy for identifying and thwarting cyberattacks. In terms of data origins, the field of intrusion detection can be categorized into two types: 1) network-centric detection; and 2) host-centric detection. This section delves into various intrusion techniques that have undergone testing and integration within ICSs. For a concise overview of our primary discoveries, see Table 4.

1) NETWORK-BASED INTRUSION DETECTION

In the realm of cybersecurity, the deployment of network-based IDSs has become imperative due to the concealment of cyberattack vectors within the stream of network commands. Nonetheless, a distinct type of attack exists, characterized by the presence of multiple control commands. These commands appear valid when scrutinized on a per-packet basis; however, when observed collectively, they hold the potential to compromise the proper functioning of PLCs. To effectively identify such attacks, a novel approach involving critical state analysis has been introduced. This approach entails the monitoring of a sequence of packets that induce changes in the states of PLCs. Furthermore, the application of deterministic finite automata (DFA) has emerged as a prevalent technique for constructing a comprehensive traffic model conducive to intrusion detection. Investigating scenarios that involve periodic Modbus/TCP traffic between HMIs and PLCs, Goldenberg, and Woo [127] devised individual DFA models for each communication channel. These models were adept at sensitively detecting anomalies. Building upon this foundation, another work [129] introduced a strategy that amalgamates DFA with configuration-level specifications to monitor communication sessions. This amalgamation notably addresses the challenge of retraining the model following configuration alterations, circumventing the need for extensive retraining data. Markman et al. [130] identified a notable characteristic within the HMI-PLC communication channel: bursts of packets with semantic significance. This realization led to the proposition of a novel burst DFA model, which significantly improved anomaly detection accuracy compared to prior methodologies. Concurrently, other researchers [128] inspected the intricacies of PLC applications through the identification of semantic attacks. Such threats were categorized into three subtypes:

TABLE 4. Existing Works Already Discussed Intrusion Detection Mechanisms

Year	Data Source	Method	Security Focus	Reference
2013	Traffic	DFA	Traffic abnormality	[129]
2014	Traffic	Autoregressive modeling	variable abnormality	[130]
2016	Traffic	DFA	Sequence Attacks	[131]
2017	Traffic	DFA	Sequence Attacks	[132]
2017	Log	SVM	Memory address values abnormality	[133]
2017	Log	OCSVM	Abnormal operation	[134]
2017	Field device status	Petri	Abnormal PLC behavior	[135]
2018	Field Device Status	Petri	Abnormal sensor and actuator behaviors	[136]
2019	HPC	SVM	HPC abnormal readings	[137]
2019	PLC parameters	FSM	PLC disruption	[138]

DFA Deterministic Finite Automation; SVM Support Vector Machine; OCSVM One Class Support Vector Machine; FSM Finite State Machine.

reconnaissance, direct control, and indirect control. This categorization paved the way for the development of a semantic network intrusion detection approach that encompasses various behavioral models, encompassing constants, attributes, and continuous series. To establish baseline expectations for constant and attribute data, a set of anticipated values was formulated. Techniques such as autoregression and control limits were harnessed to model continuous data. This multifaceted approach collectively contributes to enhancing the capabilities of network intrusion detection in PLC-based systems.

2) HOST-BASED INTRUSION DETECTION

To develop an effective IDS, cybersecurity researchers employ various techniques, including modeling state values such as relevant PLC memory addresses and control system state transitions. Previous studies extensively utilized supervised and semisupervised machine learning methods to detect anomalies or abnormal behaviors [131], [132]. The authors of these research works capitalized on relevant memory address values alongside time stamps to create a model capable of distinguishing abnormal operations within PLCs. In pursuit of this objective, they introduced two anomaly detection strategies based on PN, focusing on experimental validation. Initially, they manually constructed a white list that mirrored field device characteristics using PN modeling. This white list was subsequently converted into an LD format, incorporating PN-based constraint conditions. This conversion empowered the PLC to identify and respond to abnormal behaviors [133]. However, this white-listing approach had limitations, particularly in cases where the system displayed complexity. To address this issue, researchers enhanced the white-listing method by introducing an automatic generation technique [134]. Here, the authors employed the SFC programming language instead of LD. Moreover, self-parameters were integrated to detect various malicious attacks. Krishnamurthy et al. [135] focused on modeling baseline behaviors within PLC-based systems and subsequently identifying anomalies. Their approach is appropriate for both multithreaded as well as interrupt-driven processes across different PLCs. Building upon this foundation, Chatterjee et al. [136] developed a (k, l) threshold signature scheme using a finite-state machine. This approach had the capability to detect both patched devices and manipulated states, demonstrating efficacy particularly for legacy PLCs.

E. HONEYPOT-BASED DETECTION

In contrast to the passive techniques mentioned earlier (see Sections V-A–V-D), the method of honeypot detection involves actively monitoring network conditions, gathering data, and analyzing potential threats. When it comes to safeguarding PLC-based systems, there are several features associated with honeypots, including obfuscation, high-fidelity emulation, secure storage of malware samples, and the redirection of network traffic. The utilization of honeypot-based detection systems has demonstrated their effectiveness in identifying malicious intrusions and potential vulnerabilities before they can result in severe attacks. As a result, certain researchers have turned their attention to examining honeypots as a promising approach for enhancing the security of systems.

When categorizing existing honeypots designed for PLCs based on their level of interaction, we can identify two main categories. The first category consists of low-interaction honeypots, with Conpot [137] as an example. The second category encompasses high-interaction honeypots like CryPLH [138], [139], XPOT [140], and S7COMMTrace [141]. When applying honeypots to ICSs, researchers take into account important attributes such as performance, authenticity, scalability, cost, and risk. Conpot, falling into the low-interaction class, supports multiple protocols and offers limited function codes. However, it has a drawback in that attackers can often detect its presence due to its distinctive fingerprint. To enhance authenticity, CryPLH has developed improved interaction capabilities and introduced more original PLC implementations. This has proven highly effective in real-world control network deployments, allowing CryPLH to gather valuable data. In contrast, S7COMMTrace boasts a more extensive range of subfunction codes within the protocol, leading to a more accurate simulation of PLC behavior.

TABLE 5. Digital Forensics for PLC-Based Systems

Year	Level	PLC	Data	Tools	Reference
2014	Network	S7 SIMATIC PLC	Network Traffic	DFA model	[148]
2015	Device	S7 SIMATIC 1200 PLC	Values of relevant memory addresses	PLC Logger, *CFTT*	[149]
2015	Device	S7 SIMATIC 1200 PLC	Values of relevant memory addresses	N/A	[150]
2016	Device	S7 SIMATIC PLC	TIA Portal project file	TIA Portal software	[152]
2017	Network	GE Fanuc Series 90-30	Values of relevant memory address	N/A	[154]
2017	Network	Allen-Bradley Micrologix 1400	Network Traffic	Wireshark	[66]
2017	Device	S7 SIMATIC 1200 PLC	Values of relevant memory address	OCSVM algorithm	[133]
2017	Device	S7 SIMATIC 1200 PLC	Values of relevant memory address	SVM algorithm	[134]
2017	Device	S7 SIMATIC 1200 PLC	Critical value, data block address and time-stamp	N/A	[153]
2018	Network	S7 SIMATIC 1200 PLC	Network traffic	Wireshark	[155]
2018	Network	Internet-facing PLC	System information via a web interface	N/A	[151]

DFA: deterministic finite automation; SVM: support vector machine; OCSVM: one class support vector machine; FSM: finite-state machine.

As a result, the risk of discovery by cyber search engines like Shodan is minimized. Moreover, a high-quality PLC honeypot should facilitate program compilation and interpretation to enhance its interactive capabilities. XPOT has achieved this by enabling the honeypot to be programmed using standard integrated development environments (IDEs).

F. DIGITAL FORENSICS

Security strategies are meticulously crafted not solely to shield PLC applications from malicious threats, but also to possess the capability of preemptive measures and the ability to trace back to the origin of any issues. Consequently, the role of digital forensics becomes paramount in addressing the latter concern. While a multitude of frameworks, methodologies, and tools have been devised for IT system forensics, only a fraction of these are directly applicable to PLC environments due to the pronounced distinctions in the specific requirements of ICSs, as elaborated in Section II-E.

The continuous availability of PLC-based systems around the clock has granted researchers profound insights into real-time data acquisition from operational instances. This encompasses the extraction of volatile memory data and data stored on hard disks. The concept of live forensics is progressively emerging as a practical solution for instantaneous digital investigations. Moreover, an effective strategy to bolster digital evidence analysis involves the enhancement of logging capabilities. Concurrently, akin to digital forensics, the realm of incident response research is dedicated to addressing and recovering from incidents in ICSs, aiming to reinstate normal operations. This convergence of methodologies, along with the development of tools and corresponding case studies, has captivated researchers, leading them to adapt digital forensics techniques for legacy PLC-based systems. See Table 5 for a more comprehensive breakdown of details.

Table 5 presents the utilization of cutting-edge methodologies at both the device and network levels. Concerning network strategies, the prevalent approach involves the parsing of proprietary industrial protocols and data extraction. This practice has gained prominence in analyzing interdevice communication, as seen in [65], [146], [152], and [153]. At the device level, ongoing research is focused on monitoring the data within multiple memory addresses of PLCs, as indicated in [147] and [148]. It is important to note, however, that these efforts have predominantly catered to vendor-specific PLCs, such as those produced by Siemens, General Electric, Rockwell Allen-Bradley, among others. The primary objective remains the expansion of the applicability of these proposed methods to a wide range of devices and protocols. For instance, Choi et al. [149] illustrates this goal. They achieved that by utilizing a web interface provided by a PLC to develop a monitoring system that is not tied to any particular vendor. A significant aspect of this system is its collection of security logs, which play a vital role in the context of digital forensics investigations.

VI. SECURITY SOLUTIONS CHALLENGES

As PLC-based systems continue to evolve, so do the threats posed by malicious attackers. Security measures designed for PLCs encounter significant hurdles when applied. The enduring prevalence of legacy systems, characterized by outdated hardware and software, hampers seamless integration of contemporary security protocols. Interoperability issues among diverse devices and communication protocols further complicate the establishment of consistent security measures. Moreover, the limited computing resources inherent to PLCs necessitate a delicate balance between robust security implementations and optimal system performance [229], [230]. Real-time monitoring difficulties, human errors, and stringent regulatory compliance demands further amplify the complexity of securing PLCs. However, in the following, we list the most common challenges for the currently detection approaches, as shown in Fig. 15.

A. PROGRAM DETECTION CHALLENGES

1) FALSE POSITIVES/NEGATIVES

Detection systems for programs commonly utilize pattern matching and heuristics to spot malicious code. However, this approach can result in false positives, where legitimate code is mistakenly flagged as malicious, and false negatives, allowing malicious code to slip through undetected. Simply refining the

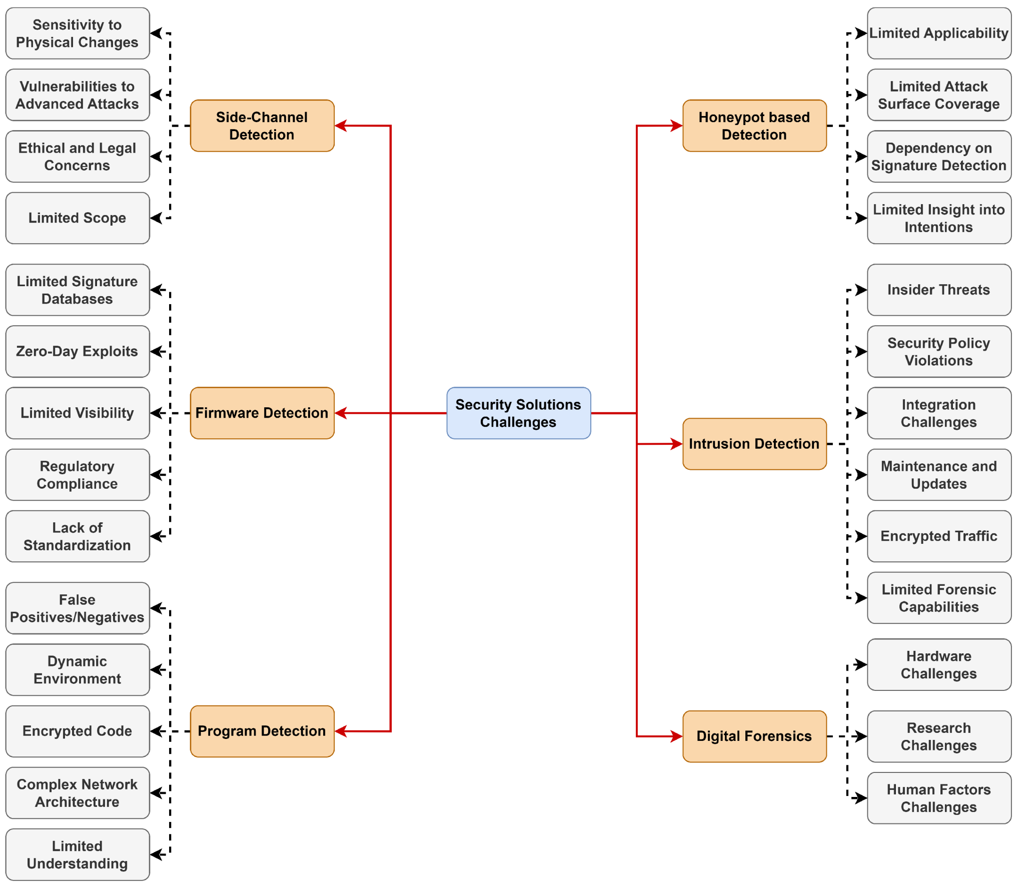

FIGURE 15. **Classification the security solutions challenges for PLC-based systems.**

system to minimize these errors does not make the security solution completely effective.

2) DYNAMIC ENVIRONMENT

Program detection methods encounter significant challenges due to the constantly changing nature of industrial environments. In these settings, replacing devices, removing/adding hardware or software, maintenance processes, etc., by ICS operators are standard practice. However, these changes can lead to false alarms or make existing detection mechanisms ineffective if they are not promptly adapted and updated.

3) ENCRYPTED CODE

Cybercriminals employ encryption techniques to evade detection, making it challenging for conventional program detection methods to identify malicious payloads concealed within encrypted code.

4) COMPLEX NETWORK ARCHITECTURE

Complex network architectures are common in ICSs, where numerous devices interact with each other, as illustrated in Fig. 4. Implementing program detection methods in these intricate setups poses significant challenges. Modern systems demand uninterrupted communication, making it difficult to seamlessly integrate detection methods without disruptions.

5) LIMITED UNDERSTANDING OF PROCESS CONTEXT

Conventional methods for detecting programs are limited in their ability to grasp the specific context of industrial processes. What might appear as a regular occurrence in one context could signify a potential security threat in another. It is crucial to comprehend the unique context of the process to accurately assess and identify potential threats.

B. FIRMWARE DETECTION CHALLENGES

1) LIMITED SIGNATURE DATABASES

Detecting firmware manipulations relies on recognizing specific patterns to identify malicious codes. Yet, maintaining an extensive and current database of these patterns is challenging. The constant emergence of new malicious code variants makes it hard to ensure the database remains both comprehensive and efficient.

2) ZERO-DAY EXPLOITS

Detecting zero-day exploits proves challenging for firmware detection systems as these exploits target vulnerabilities unknown to both the vendor and the security community. Since no signatures exist for zero-day exploits, they can effortlessly evade detection methods designed to identify firmware vulnerabilities.

3) LIMITED VISIBILITY

Methods for detecting firmware often concentrate exclusively on the firmware layer, disregarding other potential attack vectors like network-based assaults. A robust security strategy should encompass various layers of defense, as depending solely on firmware detection neglects these additional threats.

4) REGULATORY COMPLIANCE

Ensuring regulatory compliance in industrial sectors necessitates a strong security foundation. Relying solely on firmware detection may not be adequate to fulfill these compliance standards.

5) LACK OF STANDARDIZATION

In the realm of firmware detection, a striking absence of standardized methods is evident, spanning various vendors and systems. This absence of uniformity poses a significant challenge, hindering the implementation of a consistent and cohesive security strategy within diverse industrial settings.

C. INTRUSION DETECTION CHALLENGES

1) INSIDER THREATS

Conventional intrusion detection techniques may face challenges in identifying insider threats or attacks emanating from within an organization. This is because the patterns of such attacks often diverge significantly from those of external threats, making them harder to detect.

2) SECURITY POLICY VIOLATIONS

Recent intrusion detection methods face a significant hurdle: identifying subtle policy violations or deviations from security protocols is exceptionally complex. It is essential to recognize that not all policy breaches are necessarily malicious; some might inadvertently expose PLCs to threats.

3) INTEGRATION CHALLENGES

Incorporating IDSs smoothly into current PLC-based systems without causing any disruptions to their physical operations poses a notable challenge in this security strategy.

4) MAINTENANCE AND UPDATES

Ensuring the effectiveness of IDSs requires consistent maintenance, updates, and tuning. This is particularly crucial in environments utilizing PLCs, where operational downtime can incur significant costs. Implementing updates in PLC-based setups without interrupting ongoing operations poses a complex challenge.

5) ENCRYPTED TRAFFIC

Identifying malicious activities within encrypted traffic without decryption poses a significant challenge for IDSs due to the growing prevalence of encryption technologies.

6) LIMITED FORENSIC CAPABILITIES

Industrial IDSs may lack sophisticated forensic features, posing challenges in conducting comprehensive incident investigations and grasping the full scope of a security breach.

D. SIDE-CHANNEL DETECTION CHALLENGES

1) SENSITIVITY TO PHYSICAL CHANGES

Side-channel attacks exploit physical parameters like power consumption, EM emissions, and timing variations. These measurements are susceptible to environmental influences and can fluctuate due to system aging or physical changes. This sensitivity undermines the long-term reliability of existing side-channel detection methods.

2) VULNERABILITIES TO ADVANCED ATTACKS

As methods for detecting side-channel attacks become more advanced, attackers are also evolving their techniques. Skilled attackers can now use intricate strategies to bypass or manipulate these detection methods, making them ineffective against increasingly complex and novel attack approaches.

3) ETHICAL AND LEGAL CONCERNS

Side-channel attacks frequently entail the monitoring of physical signals, giving rise to ethical and legal dilemmas, particularly in sensitive contexts. Concerns related to privacy and legal constraints could restrict the implementation of specific side-channel detection techniques.

4) LIMITED SCOPE

Detection techniques for side-channel attacks often target particular attack types, offering effectiveness against specific classes of threats. However, these methods might not offer complete protection against all potential side-channel attack vectors. This leaves PLCs susceptible to new or unforeseen attacks, highlighting the need for more comprehensive security measures.

E. HONEYPOT-BASED DETECTION CHALLENGES

1) LIMITED APPLICABILITY

Honeypots prove to be highly efficient in conventional IT settings; however, their effectiveness diminishes in industrial systems such as PLCs. Industrial networks employ specialized architectures and communication protocols, making it challenging for standard honeypots to accurately mimic these environments. Consequently, their applicability and utility are limited in such contexts.

2) LIMITED ATTACK SURFACE COVERAGE

Honeypots are limited to detecting only certain vulnerabilities and attack methods within their predefined scope. If an attacker exploits a vulnerability not within the honeypot's coverage, the intrusion will remain unnoticed.

3) DEPENDENCY ON SIGNATURE-BASED DETECTION

Numerous honeypot strategies hinge on signature-based detection techniques, which are susceptible to evasion by attackers employing innovative methods or exploiting zero-day vulnerabilities.

4) LIMITED INSIGHT INTO INTENTIONS

Methods relying on honeypots offer insights into attack techniques but might not uncover attackers' true motives. Grasping the motive behind an attack is essential for a proactive incident response strategy.

F. DIGITAL FORENSICS CHALLENGES

1) HARDWARE CHALLENGES

a) Constrained resources: Applications that rely on PLCs have inherent limitations in their ability to handle data due to their restricted CPU, memory, and I/O resources. This constraint is particularly noticeable in the case of older or legacy PLCs. When it comes to forensic investigation, researchers might face challenges in obtaining and analyzing data from these systems.

b) Local access: Due to the remote placement of field devices, accessing compromised devices poses a challenge for forensic tools that depend on close physical proximity.

c) Proprietary Devices, protocols, and software: Devices produced by individual manufacturers utilize proprietary protocols, unique OSs, and at times specialized hardware. These factors create challenges for all-encompassing forensic tools to be effective in industrial control environments. In most cases, specific manufacturers do provide a restricted range of compatible interfaces to aid in enabling digital forensic functionalities.

d) Insufficient logging: Because the main objectives of utilizing PLCs involve overseeing and managing physical processes, the logging mechanisms associated with this function do not adequately facilitate comprehensive security inquiries. Furthermore, the storage of logging data also imposes an additional load on the memory of PLCs, which is constrained by its limited size.

e) Huge data to be processed: Several sensors or actuators produce a substantial volume of data related to lower level control processes. This abundance of data adds complexity to the tasks of filtering and analyzing the pertinent information.

2) RESEARCH CHALLENGES

a) Constrained resources: Using simulators, the replication of intricate industrial situations remains challenging for the purpose of conducting digital forensic research experiments. Regrettably, simulations conducted without meticulous deliberation can occasionally lead investigators astray, ultimately leading to erroneous conclusions.

b) Small-scale testbeds: Creating testbeds comprising actual physical equipment is a prudent decision on the part of researchers. Regrettably, this approach faces scalability challenges due to the substantial costs associated with real industrial hardware.

c) Research for specific control processes: Unlike digital forensics in IT systems, the particular disparity lies in the focused control procedure. However, insufficient exploration in this domain results in the inability to retroactively trace occurrences stemming from semantic attacks.

3) HUMAN FACTORS CHALLENGES

a) Lack of background knowledge: Frequently, ICS operators find themselves without the essential foundational comprehension. This includes a comprehensive grasp of intricate control protocols, detailed knowledge regarding vulnerable devices, an awareness of how forensic tools affect system performance, and various other related aspects.

b) Industry collaboration: Due to concerns about data leakage, most industrial enterprises are hesitant to collaborate with the research community. This reluctance could potentially impede the progress of practical advancements in digital forensic tools and methodologies.

VII. FUTURE SECURITY DIRECTIONS

Nevertheless, all security solutions presented so far couldn't sufficiently address the evolving threat landscape. Upon scrutinizing the available body of work, it becomes obvious that industrial vendors and engineers must undertake secure upgrades or updates of control systems and their components, precisely PLCs. To fulfill this goal, it is essential to employ secure embedded systems and communication protocols. Moreover, acknowledging the vital need for thorough testing and analysis, PLC applications require validation techniques known for their exceptional precision. This level of precision can be attained by incorporating virtualization and utilizing open-source industrial control units. Furthermore, we contend that the integration of existing systems with cloud-based, fog-based, and dynamic network strategies opens up new avenues for ICS operators to bolster their defenses against a multitude of attacks and surmount prevailing security challenges. In the ensuing discourse, we outline six pivotal directions that underscore our approach to cybersecurity in the pursuit of conceiving considerably more fortified PLC-based systems.

A. SECURE EMBEDDED SYSTEMS

In the upcoming years, there will be a growing requirement to incorporate advanced security features into embedded systems, commonly known as "secure embedded systems." The vulnerabilities identified in current PLCs mainly originate from weaknesses within their firmware and control logics, as detailed in Section IV. To enhance the security level of PLCs, it is crucial to consider supplementary security measures such as secure boot processes, reliable program updates, and efficient embedded management. As a solution, we propose that manufacturers and industrial engineers explore inventive methods to develop resilient controllers. These methods could encompass integrating embedded hypervisors [214], employing CPUs with security processors, adopting frameworks for secure firmware updates, and incorporating secure boot mechanisms, either through software or hardware, in the next iteration of PLC applications. Moreover, we suggest embracing the principles outlined in the "Design Life-cycle of Secure Embedded Devices" methodology [215]. It is important to acknowledge that these solutions might necessitate greater computational resources. Consequently, a key challenge for future research lies in discovering approaches to ensure security without compromising system functionality.

B. SECURE COMMUNICATION PROTOCOLS

The vulnerabilities present in outdated communication protocols compromise the security of the existing PLC-based systems. These protocols exhibit weaknesses like nonencrypted transmission, lack of user authentication, and absence of integrity verification mechanisms. As explained in Section IV, these systems are open to various types of attacks, including DoS, injection, replay, and MitM attacks. In response, manufacturers of industrial equipment have taken measures to bolster the security of their communication protocols. They have incorporated features such as encryption, antireplay protections, authorization mechanisms, and integrity checks. Nonetheless, ensuring the compatibility of these upgraded protocol versions with the existing array of PLCs poses a significant challenge. An example is the latest version of the S7CommPlus protocol, which cannot be used with older S7 PLC models like S7-300 and S7-400. However, despite the advancements in protocol versions, our investigations revealed that these versions still contain vulnerabilities exploitable by sophisticated attackers. For instance, the most recent iteration of the S7CommPlus protocol employs complex encryption algorithms to secure communication between S7 PLCs and their associated TIA portal against replay attacks. Despite this effort, several studies [49], [82], [84], [85], [86], [88] highlighted that malicious attackers can orchestrate attacks targeting these protocols. As a consequence, it remains crucial to persist in research aimed at enhancing industrial communication protocols. This involves both refining existing protocols and developing novel protocols incorporating advanced security measures.

C. VIRTUALIZATING PLC-BASED SYSTEMS

Given the financial implications and prevailing norms in the industry, the task of scrutinizing experiments or affirming the effectiveness of security solutions in real industrial settings is immensely challenging. Consequently, there exists a pressing need for controlled testing environments, commonly referred to as testbeds. These testbeds can be categorized into four distinct groups: those grounded in practical implementations [221], [222], simulations integrating tangible devices [220], solitary simulations, and collaborative simulations [223]. With the advent of virtualization technologies, virtual federated simulations offer distinct advantages in terms of cost efficiency and scalability, particularly in the realm of ICSs. The central focus of ongoing research revolves around the virtualization of industrial components, such as virtual PLCs, and the creation of decoy systems using virtual hosts within the control network.

It is imperative to address concerns surrounding the accuracy of virtual components. For instance, if attackers were to identify disparities between the characteristics of virtual devices in the decoy system and their real-world counterparts, they might refrain from advancing their nefarious activities in the virtual environment. In essence, the value of the decoy system could be compromised due to the inability to faithfully replicate the characteristics of real devices. In a broader context, the future direction of PLC-based system virtualization goes beyond the mere establishment of an efficient performance testing platform, e.g., testing platforms against cyberattacks similar to what Verma et al. introduced in [224]. It also aims to develop an exceptional and manageable alternative to the existing industrial devices, one that is both reliable and capable of meeting industrial demands.

D. OPEN-SOURCE UNITS

Simulating PLCs in virtual environments presents a notable challenge. This arises due to potential gaps in comprehending their internal behaviors, which can be further compounded when attempting to directly port PLC source codes onto general-purpose OSs. In the context of simulated attack experiments aimed at uncovering system vulnerabilities, a significant hurdle emerges: researchers must utilize PLCs as part of a "hardware-in-the-loop" setup. However, the progress of such endeavors is impeded by the constraints imposed by vendors' proprietary software and hardware. These limitations restrict the exploration of the embedded system's various facets, including operational logic and internal mechanisms. To surmount this predicament, a solution comes in the form of the OpenPLC project [154]. This initiative was conceptualized with the objective of creating a standardized open-source framework that not only offers the complete source code but also encompasses an IDE and a range of available hardware configurations. Notably, this project caters to platforms such as Raspberry Pi, Arduino, and ESP8266. At the heart of the OpenPLC project lies a modular framework, granting researchers the freedom to construct tailored testbeds. This adaptability extends to incorporating virtualization techniques, allowing for the implementation of specific features as required. For instance, the integration of an encryption layer directly into PLCs, thereby bolstering the security of communication channels, can be seamlessly realized. As we look ahead, the emergence of open-source PLCs signals a promising trajectory as Mellado and Núñez introduced in their IoT-PLC project [225]. Their new project provides not only regulatory control capabilities but also fog-computing functionalities as filtering and field data storage. Nevertheless, the evolution should not be limited to these entities alone. It is imperative that a broader array of open-source industrial control units be cultivated. This expansion is vital for the research community to undertake meticulous cybersecurity analyses targeting each pivotal component within PLC applications.

E. COMPUTING BASED ON CLOUD

In the realm of the Industrial Internet of Things (IIoT), the integration of cloud-based computing has facilitated the transition from conventional PLC-based control systems to cloud environments [213]. This transition showcases a dynamic and flexible approach to treating PLCs as a service. However, this shift has also led to an expansion of potential vulnerabilities due to the intricate nature of large-scale frameworks, resulting in increased points of entry from diverse sources [226], [227]. Addressing the future security concerns associated with safeguarding these control systems involves the utilization of security platforms hosted in the cloud [228]. To illustrate, consider innovative platforms that exemplify this approach: Cloud PLC Prototype: A cutting-edge concept known as the "cloud PLC"[3] employs the substantial analytical capabilities of cloud computing resources. This prototype executes thorough security assessments, enabling a final validation of commands before they are executed on actual controlled devices. Security Cloud Platform for Service-Oriented Architectures is another approach that involves the creation of a security-focused cloud platform tailored for service-oriented architecture systems. This platform serves as a comprehensive "tool-box" equipped with essential functionalities such as service planning, end-to-end security, monitoring, and enforcement. By providing these tools, the platform ensures data integrity and effectively mitigates potential attacks. Furthermore, this adaptable "tool-box" approach extends its utility to bolstering security and privacy at the fog layer. This augmentation is particularly important in countering cyber threats like compromised fog node attacks, thereby enhancing the overall resilience of the system. In conclusion, the emergence of cloud-based computing within the IIoT landscape has led to a paradigm shift in control systems. While presenting novel opportunities, it has also necessitated the implementation of robust security measures. The proposed innovative platforms, such as the cloud PLC prototype and the security cloud platform for service-oriented architectures, exemplify the strides being taken to fortify these systems against evolving cybersecurity challenges.

F. MOVING TARGET DEFENSE (MTD)

Regarding the prerequisites for PLC-based systems outlined in Section II, the task of frequently updating or patching these systems proves to be a significant challenge, particularly in the case of systems engaged in continuous processing operations. An approach that holds promise as an active defense technique is MTD. This approach involves deliberately allowing a portion of vulnerabilities that have not been patched to persist within the foundational systems. The intention is to introduce supplementary mechanisms that complicate the efforts of adversaries attempting to launch attacks. Up to this point, a handful of notable MTD research efforts have been conducted for control systems, as shown in [216], [217], and [218]. These efforts encompass the implementation of dynamic IP and the utilization of randomly configured parameters. The employment of dynamic IP techniques serves to obscure the identification of peer hosts during the reconnaissance phase and hinders seamless communication between them. On the other hand, the incorporation of random configuration parameters imparts a time-varying and stochastic nature to the systems, limiting attackers' foreknowledge of the control process.

It is important to acknowledge that MTD is not an independent and comprehensive solution by itself; rather, it operates as a redundant and harmonized security augmentation. Consequently, MTD stands as a promising avenue to enhance the security and resilience of PLC-based systems, thereby contributing to a more secure future for such systems.

VIII. CONCLUSION

In this article, we conducted an in-depth review of the literature focusing on the security aspects of PLCs and related systems. This comprehensive survey encompassed various dimensions, including vulnerabilities, attack scenarios, security detection methodologies, digital forensic investigations, and prospects for future developments. The review examined these aspects from two perspectives: the fundamental component level, i.e., PLC level, and the overarching system level. Our investigation not only delved into the existing security landscape of control systems but also proffered proactive recommendations for fortifying forthcoming control systems. Diverging from prior surveys, our work introduced precise classifications for vulnerabilities, attack types, and security detection strategies. Regarding vulnerabilities in PLCs, our analysis scrutinized facets such as programming, memory management, and firmware integrity. Furthermore, we analyzed application software, communication protocols, and interconnected industrial devices. Categorically classifying attacks, we identified three major domains: attacks aimed at compromising system availability, tampering with data integrity, and breaching data confidentiality. Subsequently, we presented a range of security detection methods, each offering distinctive detection capabilities. These methods were broadly categorized as program detection, firmware analysis, device fingerprint-based identification, intrusion detection, and honeypot-based monitoring. Overall, our work concludes with actionable recommendations directed toward the research and industry community, aimed at enhancing the security posture of future control systems built upon PLC foundations.

ACKNOWLEDGMENT

The authors would like to thank Lisa Coiffard for her constructive feedback in the preparation of this paper.

[3][Online]. Available: http://cloudplc.org/

REFERENCES

[1] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Secur. Privacy*, vol. 9, no. 3, pp. 49–51, May/Jun. 2011, doi: 10.1109/MSP.2011.67.

[2] N. Falliere, L. O. Murchu, and E. Chien, *W32. Stuxnet Dossier*. Mountain View, CA, USA: Symantec Corp., 2011.

[3] D. Hentunen and T. Antti, *Havex Hunts for ICS/SCADA Systems*, Helsinki, Finland: F-Secure, 2014.

[4] A. Di Pinto, Y. Dragoni, and A. Carcano, "TRITON: The first ICS cyber attack on safety instrument systems," in *Proc. Black Hat USA*, 2018, pp. 1–26.

[5] M. J. Assante, "Confirmation of a coordinated attack on the Ukrainian power grid," *SANS Ind. Control Syst. Secur. Blog*, vol. 207, 2016. [Online]. Available: https://www.sans.org/blog/confirmation-of-a-coordinated-attack-on-the-ukrainian-power-grid/

[6] R. Lee, M. J. Assante, and T. Conway, "German steel mill cyber attack," *Ind. Control Syst.*, vol. 30, no. 62, 2014, pp. 1–15.

[7] M. Tiegelkamp and K. John, *IEC 61131-3: Programming Industrial Automation Systems*. Berlin, Germany: Springer, 1995.

[8] F. Fronchetti et al., "Language impact on productivity for industrial end users: A case study from programmable logic controllers," *J. Comput. Lang.*, vol. 69, 2022, Art. no. 101087, doi: 10.1016/j.cola.2021.101087.

[9] S. McLaughlin et al., "The cybersecurity landscape in industrial control systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1039–1057, May 2016, doi: 10.1109/JPROC.2015.2512235.

[10] S. A. Milinković and L. R. Lazić, "Industrial PLC security issues," in *Proc. 20th Telecommun. Forum*, Belgrade, Serbia, 2012, pp. 1536–1539, doi: 10.1109/TELFOR.2012.6419513.

[11] F. Khorrami, P. Krishnamurthy, and R. Karri, "Cybersecurity for control systems: A process-aware perspective," *IEEE Des. Test*, vol. 33, no. 5, pp. 75–83, Oct. 2016, doi: 10.1109/MDAT.2016.2594178.

[12] A. Amrein et al., "Security intelligence for industrial control systems," *IBM J. Res. Develop.*, vol. 60, no. 4, pp. 13:1–13:12, Jul./Aug. 2016, doi: 10.1147/JRD.2016.2575698.

[13] J. E. Rubio, C. Alcaraz, R. Roman, and J. Lopez, "Analysis of intrusion detection systems in industrial ecosystems," in *Proc. 14th Int. Joint Conf. e-Bus. Telecommun.*, 2017, pp. 116–128, doi: 10.5220/0006426301160128.

[14] S. Nazir, S. Patel, and D. Patel, "Assessing and augmenting SCADA cyber security: A survey of techniques," *Comput. Secur.*, vol. 70, 2017, pp. 436–454, doi: 10.1016/j.cose.2017.06.010.

[15] C. Davidson, T. Andel, M. Yampolskiy, T. McDonald, B. Glisson, and T. Thomas, "On SCADA PLC and fieldbus cyber-security," in *Proc. 13th Int. Conf. Cyber Warfare Secur.*, 2018, pp. 140–148.

[16] Y. Hu, A. Yang, H. Li, Y. Sun, and L. Sun, "A survey of intrusion detection on industrial control systems," *Int. J. Distrib. Sens. Netw.*, vol. 14, no. 8, 2018, doi: 10.1177/155014771879461.

[17] I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev, "Programmable logic controller forensics," *IEEE Secur. Privacy*, vol. 15, no. 6, pp. 18–24, Nov./Dec. 2017, doi: 10.1109/MSP.2017.4251102.

[18] R. N. Rodofile, K. Radke, and E. Foo, "Extending the cyber-attack landscape for SCADA-based critical infrastructure," *Int. J. Crit. Infrastruct. Protection*, vol. 25, pp. 14–35, 2019, doi: 10.1016/j.ijcip.2019.01.002.

[19] A. Volkova, M. Niedermeier, R. Basmadjian, and H. de Meer, "Security challenges in control network protocols: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 619–639, First Quarter 2019, doi: 10.1109/COMST.2018.2872114.

[20] X. Pan, Z. Wang, and Y. Sun, "Review of PLC security issues in industrial control system," *J. Cyber Secur.*, vol. 2, no. 2, pp. 69–83, 2020, doi: 10.32604/jcs.2020.010045.

[21] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Cyber–physical systems forensics: Today and tomorrow," *J. Sens. Actuator Netw.*, vol. 9, no. 3, Art. no. 37, doi: 10.3390/jsan9030037.

[22] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan, and N. Meskin, "Cybersecurity for industrial control systems: A survey," *Comput. Secur.*, vol. 89, 2020, Art. no. 101677, doi: 10.1016/j.cose.2019.101677.

[23] R. A. Awad, S. Beztchi, J. M. Smith, B. Lyles, and S. Prowell, "Tools, techniques, and methodologies: A survey of digital forensics for SCADA systems," in *Proc. 4th Annu. Ind. Control Syst. Secur. Workshop*, New York, NY, USA, 2018, pp. 1–8, doi: 10.1145/3295453.3295454.

[24] H. R. Ghaeini and N. O. Tippenhauer, "HAMIDS: Hierarchical monitoring intrusion detection system for industrial control systems," in *Proc. 2nd ACM Workshop Cyber-Phys. Syst. Secur. Privacy*, 2016, pp. 103–111, doi: 10.1145/2994487.2994492.

[25] H. R. Ghaeini, N. O. Tippenhauer, and J. Zhou, "Zero residual attacks on industrial control systems and stateful countermeasures," in *Proc. 14th Int. Conf. Availability, Rel. Secur.*, 2019, pp. 1–10, doi: 10.1145/3339252.3340331.

[26] R. Sun, A. Mera, L. Lu, and D. Choffnes, "SoK: Attacks on industrial control logic and formal verification-based defenses," in *Proc. IEEE Eur. Symp. Secur. Privacy*, Vienna, Austria, 2021, pp. 385–402, doi: 10.1109/EuroSP51992.2021.00034.

[27] H. P. D. Nguyen, L. Ruiz, and Z. Rajnai, "Industrial control system (ICS): The general overview of the security issues and countermeasures," in *Informatics and Cybernetics in Intelligent Systems* (Lecture Notes in Networks and Systems Series), vol. 228, R. Silhavy Ed. Cham, Switzerland: Springer, 2021.

[28] V. R. Malik, K. Gobinath, S. Khadsare, A. Lakra, and S. V. Akulwar, "Security challenges in industry 4.0 SCADA systems—A digital forensic prospective," in *Proc. Int. Conf. Artif. Intell. Comput. Sci. Technol.*, Yogyakarta, Indonesia, 2021, pp. 229–233, doi: 10.1109/ICAICST53116.2021.9497829.

[29] J. Hajda, R. Jakuszewski, and S. Ogonowski, "Security challenges in industry 4.0 PLC systems," *Appl. Sci.*, vol. 11, no. 21, Art. no. 9785, doi: 10.3390/app11219785.

[30] M. Alanazi, A. Mahmood, and M. J. M. Chowdhury, "SCADA vulnerabilities and attacks: A review of the state-of-the-art and open issues," *Comput. Secur.*, vol. 125, 2023, Art. no. 103028, doi: 10.1016/j.cose.2022.103028.

[31] F. Casino et al., "Research trends, challenges, and emerging topics in digital forensics: A review of reviews," *IEEE Access*, vol. 10, pp. 25464–25493, 2022, doi: 10.1109/ACCESS.2022.3154059.

[32] W. Alsabbagh and P. Langendörfer, "A flashback on control logic injection attacks against programmable logic controllers," *Automation*, vol. 3, pp. 596–621, 2022.

[33] R. Ma et al., "Towards comprehensively understanding the run-time security of programmable logic controllers: A 3-year empirical study," 2022, *arXiv:2212.14296*.

[34] Y. Li, S. Wu, and Q. Pan, "Network security in the industrial control system: A survey," 2023, *arXiv:2308.03478*.

[35] M. Cook, A. Marnerides, C. Johnson, and D. Pezaros, "A survey on industrial control system digital forensics: Challenges, advances and future directions," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 3, pp. 1705–1747, Third Quarter 2023, doi: 10.1109/COMST.2023.3264680.

[36] A. M. Y. Koay, R. K. L Ko, H. Hettema, and K. Radke, "Machine learning in industrial control system (ICS) security: Current landscape, opportunities and challenges," *J. Intell. Inf. Syst.*, vol. 60, no. 2, pp. 377–405, 2022, doi: 10.1007/s10844-022-00753-1.

[37] H. Kayan, M. Nunes, O. Rana, P. Burnap, and C. Perera, "Cybersecurity of industrial cyber-physical systems: A review," *ACM Comput. Surv.*, vol. 54, no. 11s, pp. 1–35, Sep. 2022, doi: 10.1145/3510410.

[38] W. Alsabbagh and P. Langendörfer, "A stealth program injection attack against S7-300 PLCs," in *Proc. IEEE 22nd Int. Conf. Ind. Technol.*, Valencia, Spain, 2021, pp. 986–993, doi: 10.1109/ICIT46573.2021.9453483.

[39] W. Alsabbagh and P. Langendörfer, "A control injection attack against S7 PLCs—Manipulating the decompiled code," in *Proc. IEEE 47th Annu. Conf. Ind. Electron. Soc.*, Toronto, ON, Canada, 2021, pp. 1–8, doi: 10.1109/IECON48115.2021.9589721.

[40] A. Serhane, M. Raad, R. Raad, and W. Susilo, "PLC code-level vulnerabilities," in *Proc. Int. Conf. Comput. Appl.*, Beirut, Lebanon, 2018, pp. 348–352, doi: 10.1109/COMAPP.2018.8460287.

[41] A. Serhane, M. Raad, R. Raad, and W. Susilo, "Programmable logic controllers based systems (PLC-BS): Vulnerabilities and threats," *SN Appl. Sci.*, vol. 1, pp. 1–12, 2019, doi: 10.1007/s42452-019-0860-2.

[42] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security* (Lecture Notes in Computer Science), vol. 10683. Cham, Switzerland: Springer, 2018, pp. 110–126.

[43] S. E. Valentine, "PLC code vulnerabilities through SCADA systems," Ph.D. dissertation, Dept. Comput. Sci., Univ. South Carolina, Columbia, SC, USA, Jan. 2013.

[44] G. Bonney, H. Höfken, B. Paffen, and M. Schuba, "ICS/SCADA security analysis of a beckhoff CX5020 PLC," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy*, Angers, France, 2015, pp. 1–6.

[45] G. P. H. Sandaruwan, P. S. Ranaweera, and V. A. Oleshchuk, "PLC security and critical infrastructure protection," in *Proc. IEEE 8th Int. Conf. Ind. Inf. Syst.*, Peradeniya, Sri Lanka, 2013, pp. 81–85, doi: 10.1109/ICIInfS.2013.6731959.

[46] M. H. Rais, R. A. Awad, J. Lopez, and I. Ahmed, "JTAG-based PLC memory acquisition framework for industrial control systems," *Forensic Sci. Int.: Digit. Investigation*, vol. 37, 2021, Art. no. 301196, doi: 10.1016/j.fsidi.2021.301196.

[47] E. Leverett and R. Wightman, "Vulnerability inheritance programmable logic controllers," in *Proc. 2nd Int. Symp. Res. Grey-Hat Hacking*, Grenoble, France, 2013.

[48] W. Alsabbagh and P. Langendörfer, "Patch now and attack later—Exploiting S7 PLCs by time-of-day block," in *Proc. IEEE 4th Int. Conf. Ind. Cyber-Phys. Syst.*, Victoria, BC, Canada, 2021, pp. 144–151, doi: 10.1109/ICPS49255.2021.9468226.

[49] W. Alsabbagh and P. Langendöerfer, "A new injection threat on S7-1500 PLCs—Disrupting the physical process offline," *IEEE Open J. Ind. Electron. Soc.*, vol. 3, pp. 146–162, 2022, doi: 10.1109/OJIES.2022.3151528.

[50] W. Alsabbagh and P. Langendörfer, "No need to be online to attack—Exploiting S7-1500 PLCs by time-of-day block," in *Proc. XXVIII Int. Conf. Inf., Commun. Autom. Technol.*, Sarajevo, Bosnia and Herzegovina, 2022, pp. 1–8, doi: 10.1109/ICAT54566.2022.9811147.

[51] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *Int. J. Crit. Infrastruct. Protection*, vol. 6, no. 2, pp. 76–84, 2013, doi: 10.1016/j.ijcip.2013.04.004.

[52] D. Peck and D. Peterson, "Leveraging ethernet card vulnerabilities in field devices," in *Proc. SCADA Secur. Sci. Symp.*, 2009, 1–19.

[53] C. Schuett, J. Butts, and S. Dunlap, "An evaluation of modification attacks on programmable logic controllers," *Int. J. Crit. Infrastruct. Protection*, 2014, vol. 7, no. 1, pp. 61–68, doi: 10.1016/j.ijcip.2014.01.004.

[54] L. Garcia, F. Brasser, M. H. Cintuglu, A. R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, my Malware knows physics! Attacking PLCs with physical model aware rootkit," in *Proc. Netw. Distribut. Syst. Secur. Symp.*, San Diego, CA, USA, 2017, pp. 1–15.

[55] A. Robles-Durazno, N. Moradpoor, J. McWhinnie, G. Russell, and I. Maneru-Marin, "Implementation and detection of novel attacks to the PLC memory of a clean water supply system," in *Communications in Computer and Information Science*, vol. 895. Cham, Switzerland: Springer, 2018.

[56] A. Robles-Durazno, N. Moradpoor, J. McWhinnie, G. Russell, and I. Maneru-Marin, "PLC memory attack detection and response in a clean water supply system," *Int. J. Crit. Infrastruct. Protection*, vol. 26, 2019, Art. no. 100300, doi: 10.1016/j.ijcip.2019.05.003.

[57] N. Zubair, A. Ayub, H. Yoo, and I. Ahmed, "PEM: Remote forensic acquisition of PLC memory in industrial control systems," *Forensic Sci. Int.: Digit. Investigation*, vol. 40, 2022, Art. no. 301336, doi: 10.1016/j.fsidi.2022.301336.

[58] F. Tacliad, T. D. Nguyen, and M. Gondree, "DoS exploitation of Allen-Bradley's legacy protocol through fuzz testing," in *Proc. 3rd Annu. Ind. Control Syst. Secur. Workshop*, New York, NY, USA, 2017, pp. 24–31, doi: 10.1145/3174776.3174780.

[59] E. N. Ylmaz, B. Ciylan, S. Gönen, E. Sindiren, and G. Karacayılmaz, "Cyber security in industrial control systems: Analysis of DoS attacks against PLCs and the insider effect," in *Proc. 6th Int. Istanbul Smart Grids Cities Congr. Fair*, Istanbul, Turkey, 2018, pp. 81–85, doi: 10.1109/SGCF.2018.8408947.

[60] N. Sayegh, A. Chehab, I. H. Elhajj, and A. Kayssi, "Internal security attacks on SCADA systems," in *Proc. 3rd Int. Conf. Commun. Inf. Technol.*, Beirut, Lebanon, 2013, pp. 22–27, doi: 10.1109/ICCITechnology.2013.6579516.

[61] M. Niedermaier et al., "You snooze, you lose: Measuring PLC cycle times under attacks," in *Proc. 12th USENIX Conf. Offensive Technol.*, p. 12.

[62] H. Yang, L. Cheng, and M. C. Chuah, "Detecting payload attacks on programmable logic controllers (PLCs)," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Beijing, China, 2018, pp. 1–9, doi: 10.1109/CNS.2018.8433146.

[63] S. McLaughlin, "On dynamic malware payloads aimed at programmable logic controllers," in *Proc. 6th USENIX Conf. Hot Topics Secur.*, 2011, p. 10.

[64] S. McLaughlin and P. McDaniel, "SABOT: Specification-based payload generation for programmable logic controllers," in *Proc. ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2012, pp. 439–449, doi: 10.1145/2382196.2382244.

[65] S. Senthivel, I. Ahmed, and V. Roussev, "SCADA network forensics of the PCCC protocol," *Digit. Investigation*, vol. 22, pp. S57–S65, 2017, doi: 10.1016/j.diin.2017.06.012.

[66] S. A. Qasim, J. Lopez, and I. Ahmed, "Automated reconstruction of control logic for programmable logic controller forensics," in *Information Security* (Lecture Notes in Computer Science), vol. 11723, Z. Lin, C. Papamanthou, and M. Polychronakis, Eds. Cham, Switzerland: Springer, 2019.

[67] H. Yoo et al., *Detection of Intrusions and Malware, and Vulnerability Assessment* (Lecture Notes in Computer Science Series), vol. 11543. Cham, Switzerland: Springer, 2021.

[68] S. McLaughlin and S. Zonouz, "Controller-aware false data injection against programmable logic controllers," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Venice, Italy, 2014, pp. 848–853, doi: 10.1109/SmartGridComm.2014.7007754.

[69] M. Xiao, J. Wu, C. Long, and S. Li, "Construction of false sequence attack against PLC based power control system," in *Proc. 35th Chin. Control Conf.*, Chengdu, China, 2016, pp. 10090–10095, doi: 10.1109/ChiCC.2016.7554953.

[70] R. Fritz, P. Schwarz, and P. Zhang, "Modeling of cyber attacks and a time guard detection for ICS based on discrete event systems," in *Proc. 18th Eur. Control Conf.*, Naples, Italy, 2019, pp. 4368–4373, doi: 10.23919/ECC.2019.8795791.

[71] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *ICT Systems Security and Privacy Protection* (IFIP Advances in Information and Communication Technology), vol. 562, G. Dhillon, F. Karlsson, K. Hedström, and A. Zúquete, Eds. Cham, Switzerland: Springer, 2019.

[72] A. R. Abbasi and M. Hashemi, "Ghost in the PLC: Designing an undetectable programmable logic controller rootkit via pin control attack," in *Proc. Black Hat Eur.*, 2016, pp. 1–35.

[73] A. R. Abbasi, M. Hashemi, E. Zambon, and S. Etalle, "Stealth low-level manipulation of programmable logic controllers I/O by pin control exploitation," in *Critical Information Infrastructures Security* (Lecture Notes in Computer Science), vol. 10242, G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen Eds. Cham, Switzerland: Springer, 2016.

[74] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," in *Proc. Black Hat USA*, 2011, vol. 16, pp. 723–733.

[75] W. Alsabbagh, S. Amogbonjaye, D. Urrego, and P. Langendörfer, "A stealthy false command injection attack on Modbus based SCADA systems," in *Proc. IEEE 20th Consum. Commun. Netw. Conf.*, Las Vegas, NV, USA, 2023, pp. 1–9, doi: 10.1109/CCNC51644.2023.10059804.

[76] O. Eigner, P. Kreimel, and P. Tavolato, "Identifying S7comm protocol data injection attacks in cyber-physical systems," in *Proc. 5th Int. Symp. ICS SCADA Cyber Secur. Res.*, 2018, pp. 51–56, doi: 10.14236/ewic/ICS2018.6.

[77] A. Ghaleb, S. Zhioua, and A. Almulhem, "On PLC network security," *Int. J. Crit. Infrastruct. Protection*, vol. 22, pp. 62–69, 2018, doi: 10.1016/j.ijcip.2018.05.004.

[78] W. Alsabbagh and P. Langendörfer, "A fully-blind false data injection on PROFINET I/O systems," in *Proc. IEEE 30th Int. Symp. Ind. Electron.*, Kyoto, Japan, 2021, pp. 1–8, doi: 10.1109/ISIE45552.2021.9576496.

[79] S. Mehner and H. König, "No need to marry to change your name! Attacking profinet IO automation networks using DCP," in *Detection of Intrusions and Malware, and Vulnerability Assessment* (Lecture Notes in Computer Science Series), vol. 11543, R. Perdisci, C. Maurice, G. Giacinto, and M. Almgren, Eds. Cham, Switzerland: Springer, 2019.

[80] M. Noorizadeh, M. Shakerpour, N. Meskin, D. Unal, and K. Khorasani, "A cyber-security methodology for a cyber-physical industrial control system testbed," *IEEE Access*, vol. 9, pp. 16239–16253, 2021, doi: 10.1109/ACCESS.2021.3053135.

[81] W. Alsabbagh and P. Langendoerfer, "A remote attack tool against Siemens S7-300 controllers: A practical report," in *Kommunikation und Bildverarbeitung in der Automation. Technologien für die intelligente Automation*, vol. 14, J. Jasperneite and V. Lohweg, Eds. Berlin, Germany: Springer, 2022.

[82] W. Alsabbagh and P. Langendoerfer, "You are what you attack: Breaking the cryptographically protected S7 protocol," in *Proc. IEEE 19th Int. Conf. Factory Commun. Syst.*, 2023, pp. 1–8, doi: 10.13140/RG.2.2.25549.10721/1.
[83] K. Sushma, A. Nehal, Y. Hyunguk, and A. Irfan, "CLIK on PLCs! Attacking control logic with decompilation and virtual PLC," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–12.
[84] E. Biham, S. Bitan, A. Carmel, A. Dankner, U. Malin, and A. Wool, "Rogue7: Rogue engineering-station attacks on S7 Simatic PLCs," in *Proc. Black Hat USA*, 2019. [Online]. Available: https://i.blackhat.com/USA-19/Thursday/us-19-Bitan-Rogue7-Rogue-Engineering-Station-Attacks-On-S7-Simatic-PLCs-wp.pdf
[85] C. Lei, L. Donghong, and M. Liang, "The spear to break the security wall of S7CommPlus," in *Proc. Black Hat USA*, 2017. [Online]. Available: https://www.blackhat.com/docs/eu-17/materials/eu-17-Lei-The-Spear-To-Break%20-The-Security-Wall-Of-S7CommPlus-wp.pdf
[86] H. Hui and K. McLaughlin, "Investigating current PLC security issues regarding Siemens S7 communications and TIA Portal," in *Proc. Ind. Control Syst. Cyber Secur. Res.*, 2018, pp. 67–73.
[87] A. Ayub, H. Yoo, and I. Ahmed, "Empirical study of PLC authentication protocols in industrial control systems," in *Proc. IEEE Secur. Privacy Workshops*, 2021, pp. 383–397.
[88] H. Hui, K. McLaughlin, and S. Sezer, "Vulnerability analysis of S7 PLCs: Manipulating the security mechanism," *Int. J. Crit. Infrastruct. Protection*, vol. 35, 2021, Art. no. 100470, doi: 10.1016/j.ijcip.2021.100470.
[89] H. Wardak, S. Zhioua, and A. Almulhem, "PLC access control: A security analysis," in *Proc. World Congr. Ind. Control Syst. Secur.*, London, U.K., 2016, pp. 1–6, doi: 10.1109/WCICSS.2016.7882935.
[90] L. Rosa, T. Cruz, P. Simões, E. Monteiro, and L. Lev, "Attacking SCADA systems: A practical perspective," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage.*, Lisbon, Portugal, 2017, pp. 741–746, doi: 10.23919/INM.2017.7987369.
[91] A. Kleinmann, O. Amichay, A. Wool, D. Tenenbaum, O. Bar, and L. Lev, "Stealthy deception attacks against SCADA systems," in *Computer Security*. Berlin, Germany: Springer, 2017, pp. 93–109.
[92] Y. Hu, Y. Sun, Y. Wang, and Z. Wang, "An enhanced multi-stage semantic attack against industrial control systems," *IEEE Access*, vol. 7, pp. 156871–156882, 2019, doi: 10.1109/ACCESS.2019.2949645.
[93] G. Falco, C. Caldera, and H. Shrobe, "IIoT cybersecurity risk modeling for SCADA systems," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4486–4495, Dec. 2018, doi: 10.1109/JIOT.2018.2822842.
[94] E. Korkmaz, M. Davis, A. Dolgikh, and V. Skormin, "Detection and mitigation of time delay injection attacks on industrial control systems with PLCs," in *Computer Network Security* (Lecture Notes in Computer Science Series), vol. 10446, J. Rak, J. Bay, I. Kotenko, L. Popyack, V. Skormin, and K. Szczypiorski, Eds. Cham, Switzerland: Springer, 2017.
[95] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proc. IEEE*, vol. 100, no. 1, pp. 210–224, Jan. 2012, doi: 10.1109/JPROC.2011.2165269.
[96] J. Larsen, "Controlling without modifying: The stale data problem," in *Proc. S4x16*, 2016.
[97] X. Lou et al., "Learning-based time delay attack characterization for cyber-physical systems," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids*, Beijing, China, 2019, pp. 1–6, doi: 10.1109/SmartGridComm.2019.8909732.
[98] K. Yang, H. Wang, H. Wang, and L. Sun, "An effective intrusion-resilient mechanism for programmable logic controllers against data tampering attacks," *Comput. Ind.*, vol. 138, 2022, Art. no. 103613, doi: 10.1016/j.compind.2022.103613.
[99] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing PLCs as a network backdoor," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Florence, Italy, 2015, pp. 524–532, doi: 10.1109/CNS.2015.7346865.
[100] R. Spenneberg, M. Bruggemann, and H. Schwartke, "PLC-blaster: A worm living solely in the PLC," in *Proc. Black Hat Asia*, 2016. [Online]. Available: http://regmedia.co.uk/2016/04/29/plc_87458745.pdf
[101] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, and V. Roussev, "Denial of engineering operations attacks in industrial control systems," in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy*, New York, NY, USA, 2018, pp. 319–329, doi: 10.1145/3176258.3176319.
[102] R. Grandgenett, W. Mahoney, and R. Gandhi, "Authentication bypass and remote escalated I/O command attacks," in *Proc. 10th Annu. Cyber Inf. Secur. Res. Conf.*, New York, NY, USA, 2015, vol. 2, pp. 1–7, doi: 10.1145/2746266.2746268.
[103] B. Lim, D. Chen, Y. An, Z. Kalbarczyk, and R. Iyer, "Attack induced common-mode failures on PLC-Based safety system in a nuclear power plant: Practical experience report," in *Proc. IEEE 22nd Pacific Rim Int. Symp. Dependable Comput.*, Christchurch, New Zealand, 2017, pp. 205–210, doi: 10.1109/PRDC.2017.34.
[104] Y. Yao, C. Sheng, Q. Fu, H. Liu, and D. Wang, "A propagation model with defensive measures for PLC-PC worms in industrial networks," *Appl. Math. Model.*, vol. 69, pp. 696–713, 2019, doi: 10.1016/j.apm.2019.01.014.
[105] L. Xiao, M. Li, M. Gu, and J. Sun, "A hierarchy framework on compositional verification for PLC software," in *Proc. IEEE 5th Int. Conf. Softw. Eng. Serv. Sci.*, Beijing, China, 2014, pp. 204–207, doi: 10.1109/ICSESS.2014.6933545.
[106] S. Stattelmann, S. Biallas, B. Schlich, and S. Kowalewski, "Applying static code analysis on industrial controller code," in *Proc. IEEE Emerg. Technol. Factory Autom.*, Barcelona, Spain, 2014, pp. 1–4, doi: 10.1109/ETFA.2014.7005254.
[107] M. Zhang et al., "Towards automated safety vetting of PLC code in real-world plants," in *Proc. IEEE Symp. Secur. Privacy*, San Francisco, CA, USA, 2019, pp. 522–538, doi: 10.1109/SP.2019.00034.
[108] T. Ovatman, A. Aral, D. Polat, and A. O. Ünver, "An overview of model checking practices on verification of PLC software," *Softw. Syst. Model.*, vol. 15, pp. 937–960, 2016, doi: 10.1007/s10270-014-0448-7.
[109] L. Xiao, R. Wang, M. Gu, and J. Sun, "Semantic characterization of programmable logic controller programs," *Math. Comput. Model.*, vol. 55, nos. 5/6, pp. 1819–1824, 2012, doi: 10.1016/j.mcm.2011.11.038.
[110] S. McLaughlin, S. Zonouz, D. Pohly, and P. McDaniel, "A trusted safety verifier for process controller code," in *Proc. ISOC Netw. Distrib. Syst. Secur. Symp.*, 2014, doi: 10.14722/ndss.2014.23043.
[111] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated PLC code analytics," *IEEE Secur. Privacy*, vol. 12, no. 6, pp. 40–47, Nov./Dec. 2014, doi: 10.1109/MSP.2014.113.
[112] T. Chang, Q. Wei, W. Liu, and Y. Geng, "Detecting PLC program malicious behaviors based on state verification," in *Cloud Computing and Security* (Lecture Notes in Computer Science), vol. 11067, X. Sun, Z. Pan, and E. Bertino, Eds. Cham, Switzerland: Springer, 2018.
[113] S. Kottler, M. Khayamy, S. R. Hasan, and O. Elkeelany, "Formal verification of ladder logic programs using NuSMV," in *Proc. SoutheastCon*, Concord, NC, USA, 2017, pp. 1–5, doi: 10.1109/SECON.2017.7925390.
[114] M. Hailesellasie and S. R. Hasan, "Intrusion detection in PLC-based industrial control systems using formal verification approach in conjunction with graphs," *J. Hardw. Syst. Secur.*, vol. 2, pp. 1–14, 2018, doi: 10.1007/s41635-017-0017-y.
[115] X. Lv, Y. Xie, X. Zhu, and L. Ren, "A technique for bytecode decompilation of PLC program," in *Proc. IEEE 2nd Adv. Inf. Technol., Electron. Autom. Control Conf.*, Chongqing, China, 2017, pp. 252–257, doi: 10.1109/IAEAC.2017.8054016.
[116] A. Keliris and M. Maniatakos, "ICSREF: A framework for automated reverse engineering of industrial control systems binaries," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, San Diego, CA, USA, 2019, doi: 10.14722/ndss.2019.23271.
[117] T. Chang, Q. Wei, Y. Geng, and H. Zhang, "Constructing PLC binary program model for detection purposes," *J. Phys.: Conf. Ser.*, vol. 1087, no. 2, 2018, Art. no. 022022, doi: 10.1088/1742-6596/1087/2/022022.
[118] A. Abbasi, T. Holz, E. Zambon, and S. Etalle, "ECFI: Asynchronous control flow integrity for programmable logic controllers," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, 2017, pp. 437–448, doi: 10.1145/3134600.3134618.
[119] L. McMinn and J. Butts, "A firmware verification tool for programmable logic controllers," in *Critical Infrastructure Protection VI (IFIP Advances in Information and Communication Technology)*, vol. 390, J. Butts and S. Shenoi Eds. Berlin, Germany: Springer, 2012.
[120] S. Stone and M. Temple, "Radio-frequency-based anomaly detection for programmable logic controllers in the critical infrastructure," *Int. J. Crit. Infrastruct. Protection*, vol. 5, no. 2, pp. 66–73, 2012, doi: 10.1016/j.ijcip.2012.05.001.

[121] S. Stone, M. Temple, and R. Baldwin, "Detecting anomalous programmable logic controller behavior using RF-based hilbert transform features and a correlation-based verification process," *Int. J. Crit. Infrastruct. Protection*, vol. 9, pp. 41–51, 2015, doi: 10.1016/j.ijcip.2015.02.001.

[122] C. A. Gonzalez and A. Hinton, "Detecting malicious software execution in programmable logic controllers using power fingerprinting," in *Critical Infrastructure Protection VIII* (IFIP Advances in Information and Communication Technology), vol. 441, J. Butts and S. Shenoi Eds. Berlin, Germany: Springer, 2014.

[123] Y. Xiao, W. Xu, Z. Jia, Z. Ma, and D. Qi, "NIPAD: A non-invasive power-based anomaly detection scheme for programmable logic controllers," *Front. Inf. Technol. Electron. Eng*, vol. 18, pp. 519–534, 2017, doi: 10.1631/FITEE.1601540.

[124] S. Dunlap, J. Butts, J. Lopez, M. Rice, and B. Mullins, "Using timing-based side channels for anomaly detection in industrial control systems," *Int. J. Crit. Infrastruct. Protection*, vol. 15, pp. 12–26, 2016, doi: 10.1016/j.ijcip.2016.07.003.

[125] N. Boggs, J. C. Chau, and A. Cui, "Utilizing electromagnetic emanations for out-of-band detection of unknown attack code in a programmable logic controller," *Proc. SPIE*, vol. 10630, 2018, Art. no. 106300D, doi: 10.1117/12.2304465.

[126] P. Van Aubel, K. Papagiannopoulos, Å. Chmielewski, and C. Doerr, "Side-channel based intrusion detection for industrial control systems," in *Critical Information Infrastructures Security* (Lecture Notes in Computer Science Series), vol. 10707, G. D'Agostino and A. Scala, Eds. Cham, Switzerland: Springer, 2017.

[127] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *Int. J. Crit. Infrastruct. Protection*, vol. 6, no. 2, pp. 63–75, 2013, doi: 10.1016/j.ijcip.2013.05.001.

[128] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, 2014, pp. 126–135, doi: 10.1145/2664243.2664277.

[129] M. Faisal, A. A. Cardenas, and A. Wool, "Modeling Modbus TCP for intrusion detection," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Philadelphia, PA, USA, 2016, pp. 386–390, doi: 10.1109/CNS.2016.7860524.

[130] C. Markman, A. Wool, and A. A. Cardenas, "A new burst-DFA model for SCADA anomaly detection," in *Proc. Workshop Cyber-Phys. Syst. Secur. Privacy*, New York, NY, USA, 2017, pp. 1–12, doi: 10.1145/3140241.3140245.

[131] K. Yau and K.-P. Chow, "Detecting anomalous programmable logic controller events using machine learning," in *Advances in Digital Forensics XIII* (IFIP Advances in Information and Communication Technology), vol. 511, G. Peterson and S. Shenoi Eds. Cham, Switzerland: Springer, 2017.

[132] K. Yau, K. P. Chow, S. M. Yiu, and C. F. Chan, "Detecting anomalous behavior of PLC using semi-supervised machine learning," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Las Vegas, NV, USA, 2017, pp. 580–585, doi: 10.1109/CNS.2017.8228713.

[133] A. Mochizuki, K. Sawada, S. Shin, and S. Hosokawa, "On experimental verification of model based white list for PLC anomaly detection," in *Proc. 11th Asian Control Conf.*, Gold Coast, QLD, Australia, 2017, pp. 1766–1771, doi: 10.1109/ASCC.2017.8287441.

[134] S. Fujita, K. Rata, A. Mochizuki, K. Sawada, S. Shin, and S. Hosokawa, "On experimental validation of whitelist auto-generation method for secured programmable logic controllers," in *Proc. 44th Annu. Conf. IEEE Ind. Electron. Soc.*, Washington, DC, USA, 2018, pp. 2385–2390, doi: 10.1109/IECON.2018.8591275.

[135] P. Krishnamurthy, R. Karri, and F. Khorrami, "Anomaly detection in real-time multi-threaded processes using hardware performance counters," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 666–680, 2020, doi: 10.1109/TIFS.2019.2923577.

[136] U. Chatterjee, P. Santikellur, R. Sadhukhan, V. Govindan, D. Mukhopadhyay, and R. S. Chakraborty, "United we stand: A threshold signature scheme for identifying outliers in PLCs," in *Proc. ACM/IEEE 56th Des. Autom. Conf.*, Las Vegas, NV, USA, 2019, pp. 1–2.

[137] A. Jicha, M. Patton, and H. Chen, "SCADA honeypots: An in-depth analysis of Conpot," in *Proc. IEEE Conf. Intell. Secur. Inform.*, Tucson, AZ, USA, 2016, pp. 196–198, doi: 10.1109/ISI.2016.7745468.

[138] D. I. Buza, F. Juhász, G. Miru, M. Félegyházi, and T. Holczer, "CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot," in *Smart GridSecurity* (Lecture Notes in Computer Science), vol. 8448, J. Cuellar Ed. Cham, Switzerland: Springer, 2014.

[139] T. Holczer, M. Felegyhazi, and L. Buttyan, "The design and implementation of a PLC honeypot for detecting cyber attacks against industrial control systems," in *Proc. Int. Conf. Comput. Secur. Nucl. World Expert Discuss. Exchange*, 2015. [Online]. Available: http://www.hit.bme.hu/~buttyan/publications/HolczerFB2015CN.pdf

[140] S. Lau, J. Klick, S. Arndt, and V. Roth, "POSTER: Towards highly interactive honeypots for industrial control systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2016, pp. 1823–1825, doi: 10.1145/2976749.2989063.

[141] F. Xiao, E. Chen, and Q. Xu, "S7commTrace: A high interactive honeypot for industrial control system based on S7 protocol," in *Information and Communications Security* (Lecture Notes in Computer Science), vol. 10631, S. Qing, C. Mitchell, L. Chen, and D. Liu Eds. Cham, Switzerland: Springer, 2017.

[142] Y. Wang, J. Liu, C. Yang, L. Zhou, S. Li, and Z. Xu, "Access control attacks on PLC vulnerabilities," *J. Comput. Commun.*, vol. 6, pp. 311–325, 2018, doi: 10.4236/jcc.2018.611028.

[143] J. Son, S. Noh, J. Choi, and H. Yoon, "A practical challenge-response authentication mechanism for a programmable logic controller control system with one-time password in nuclear power plants," *Nucl. Eng. Technol.*, vol. 51, no. 7, pp. 1791–1798, 2019, doi: 10.1016/j.net.2019.05.012.

[144] D. Fauri, B. de Wijs, J. den Hartog, E. Costante, E. Zambon, and S. Etalle, "Encryption in ICS networks: A blessing or a curse?," in *Proc. IEEE Int. Conf. Smart Grid Commun.*, Dresden, Germany, 2017, pp. 289–294, doi: 10.1109/SmartGridComm.2017.8340732.

[145] E. N. Yılmaz and S. Gönen, "Attack detection/prevention system against cyber attack in industrial control systems," *Comput. Secur.*, vol. 77, pp. 94–105, 2018, doi: 10.1016/j.cose.2018.04.004.

[146] A. Kleinmann and A. Wool, "Accurate modeling of the Siemens S7 SCADA protocol for intrusion detection and digital forensics," *J. Digit. Forensics, Secur. Law*, vol. 9, no 2, 2014, Art. no. 4, doi: 10.15394/jdfsl.2014.1169.

[147] T. Wu and J. R. C. Nurse, "Exploring the use of PLC debugging tools for digital forensic investigations on SCADA systems," *J. Digit. Forensics, Secur. Law*, vol. 10, no. 4, 2015, Art. no. 7, doi: 10.15394/jdfsl.2015.1213.

[148] K. Yau and K. Chow, "PLC forensics based on control program logic change detection," *J. Digit. Forensics, Secur. Law*, vol. 10, no. 4, 2015, Art. no. 5, doi: 10.15394/jdfsl.2015.1211.

[149] J. Choi, H. Kim, S. Choi, J. Yun, B. Min, and H. Kim, "Vendor-independent monitoring on programmable logic controller status for ICS security log management," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, New York, NY, USA, 2019, pp. 682–684, doi: 10.1145/3321705.3331007.

[150] R. Chan and K. P. Chow, "Forensic analysis of a Siemens programmable logic controller," in *Critical Infrastructure Protection X* (IFIP Advances in Information and Communication Technology), vol. 485, M. Rice and S. Shenoi Eds. Cham, Switzerland: Springer, 2016.

[151] C. F. Chan, K. P. Chow, SM. Yiu, and K. Yau, "Enhancing the security and forensic capabilities of programmable logic controllers," in *Advances in Digital Forensics XIV* (IFIP Advances in Information and Communication Technology), vol. 532, G. Peterson and S. Shenoi, Eds. Cham, Switzerland: Springer, 2018.

[152] G. Denton, F. Karpisek, F. Breitinger, and I. Baggili, "Leveraging the SRTP protocol for over-the-network memory acquisition of a GE Fanuc Series 90-30," *Digit. Investigation*, vol. 22, pp. S26–S38, 2017, doi: 10.1016/j.diin.2017.06.005.

[153] K. Yau, K. P. Chow, and S. M. Yiu, "A forensic logging system for Siemens programmable logic controllers," in *Advances in Digital Forensics XIV* (IFIP Advances in Information and Communication Technology), vol. 532, G. Peterson and S. Shenoi Eds. Cham, Switzerland: Springer, 2018.

[154] T. R. Alves, M. Buratto, F. M. de Souza, and T. V. Rodrigues, "OpenPLC: An open source alternative to automation," in *Proc. IEEE Glob. Humanitarian Technol. Conf.*, San Jose, CA, USA, 2014, pp. 585–589, doi: 10.1109/GHTC.2014.6970342.

[155] C. Perrin, "The CIA triad." Accessed: Jun. 14, 2023. [Online]. Available: http://www.techrepublic.com/blog/security/the-cia-triad/488

[156] F. Schuster et al., "Evaluating the effectiveness of current anti-ROP defenses," in *Research in Attacks, Intrusions and Defenses* (Lecture Notes in Computer Science Series), vol. 8688, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Cham, Switzerland: Springer, 2014.

[157] L. Davi, D. Lehmann, A. R. Sadeghi, and F. Monrose, "Stitching the gadgets: On the ineffectiveness of coarse-grained control-flow integrity protection," in *Proc. USENIX Secur. Symp.*, 2014, pp. 401–416.

[158] R. M. van der Knijff, "Control systems/SCADA forensics, what's the difference?," *Digit. Investigation*, vol. 11, no. 3, pp. 160–174, 2014, doi: 10.1016/j.diin.2014.06.007.

[159] S. East, J. Butts, M. Papa, and S. Shenoi, "A taxonomy of attacks on the DNP3 protocol," in *Critical Infrastructure Protection III* (IFIP Advances in Information and Communication Technology), vol. 311, C. Palmer and S. Shenoi. Berlin, Germany: Springer, 2009.

[160] J. L. Rrushi, "SCADA protocol vulnerabilities," in *Critical Infrastructure Protection: Information Infrastructure Models, Analysis, and Defense*. Berlin, Germany: Springer, 2012, pp. 150–176, doi: 10.5555/2231096.2231107.

[161] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the Modbus protocols," *Int. J. Crit. Infrastruct. Protection*, vol. 1, pp. 37–44, 2008, doi: 10.1016/j.ijcip.2008.08.003.

[162] Y. Zhang, L. Wang, and W. Sun, "Investigating the impact of cyber attacks on power system reliability," in *Proc. IEEE Int. Conf. Cyber Technol. Autom., Control Intell. Syst.*, Nanjing, China, 2013, pp. 462–467, doi: 10.1109/CYBER.2013.6705490.

[163] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *Proc. 1st Int. Symp. ICS SCADA Cyber Secur. Res.*, 2013, doi: 10.14236/ewic/ICSCSR2013.3.

[164] H. Dierks, "PLC-automata: A new class of implementable real-time automata," *Theor. Comput. Sci.*, vol. 253, no. 1, pp. 61–93, 2001, doi: 10.1016/S0304-3975(00)00089-X.

[165] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 1981.

[166] R. Alur and D. Dill, "A theory of timed automata," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 183–235, 1994.

[167] R.S. Sreenivas and B. H. Krogh, "On condition/event systems with discrete state realizations," *Discrete Event Dyn. Syst.*, vol. 1, no. 2, pp. 209–236, 1991.

[168] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified Modeling Language Reference Manual*. Upper Saddle River, NJ, USA: Pearson Higher Educ., 2004.

[169] K. Havelund et al., "Formal analysis of the remote agent before and after flight," in *Proc. 5th NASA Langley Formal Methods Workshop*, 2000, vol. 134, p. 163.

[170] B. McDonald and F. Mueller, "T-SYS: Timed-based system security for real-time kernels," in *Proc. ACM/IEEE 13th Int. Conf. Cyber-Phys. Syst.*, Milano, Italy, 2022, pp. 247–258, doi: 10.1109/ICCPS54341.2022.00029.

[171] B. F. Adiego et al., "Applying model checking to industrial-sized PLC programs," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1400–1410, Dec. 2015, doi: 10.1109/TII.2015.2489184.

[172] D. Darvas, I. Majzik, and E. B. Viñuela, "PLC program translation for verification purposes," *Periodica Polytechnica Elect. Eng. Comput. Sci.*, vol. 61, no. 2, pp. 151–165, 2017, doi: 10.3311/PPee.9743.

[173] H. Giese, S. Glesner, J. Leitner, W. Schäfer, and R. Wagner, "Towards verified model transformations," in *Proc. 3rd Int. Workshop Model Develop., Validation Verif.*, 2006, pp. 78–93.

[174] J. Newell et al., "Translation of IEC 61131-3 function block diagrams to PVS for formal verification with real-time nuclear application," *J. Autom. Reason.*, vol. 60, pp. 63–84, 2018, doi: 10.1007/s10817-017-9415-7.

[175] Y. Xie, R. Chang, and L. Jiang, "A malware detection method using satisfiability modulo theory model checking for the programmable logic controller system," *Special Issue: Secur. Privacy Social Data Mining. Trust Manage. Internet Things*, vol. 34, no. 16, 2022, Art. no. e5724, doi: 10.1002/cpe.5724.

[176] G. Cengic, O. Ljungkrantz, and K. Akesson, "Formal modeling of function block applications running in IEC 61499 execution runtime," in *Proc. IEEE Conf. Emerg. Technol. Factory Autom.*, Prague, Czech Republic, 2006, pp. 1269–1276, doi: 10.1109/ETFA.2006.355187.

[177] H. Janicke, A. Nicholson, S. Webber, and A. Cau, "Runtime-monitoring for industrial control systems," *Electronics*, vol. 4, pp. 995–1017, 2105, doi: 10.3390/electronics4040995.

[178] L. Luccarini et al., "Formal verification of wastewater treatment processes using events detected from continuous signals by means of artificial neural networks. Case study: SBR plant," *Environ. Model. Softw.*, vol. 25, no. 5, pp. 648–660, 2010, doi: 10.1016/j.envsoft.2009.05.013.

[179] M. Zhou, F. He, M. Gu, and X. Song, "Translation-based model checking for PLC programs," in *Proc. IEEE 33rd Annu. Int. Comput. Softw. Appl. Conf.*, Seattle, WA, USA, 2009, pp. 553–562, doi: 10.1109/COMPSAC.2009.80.

[180] R. Wang, Y. Guan, L. Luo, X. Song, and J. Zhang, "Formal modelling of PLC systems by BIP components," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Kyoto, Japan, 2013, pp. 512–518, doi: 10.1109/COMPSAC.2013.85.

[181] S. Mesli-Kesraoui, A. Toguyeni, A. Bignon, F. Oquendo, D. Kesraoui, and P. Berruet, "Formal and joint verification of control programs and supervision interfaces for socio-technical systems components," *IFAC-PapersOnLine*, vol. 49, no. 19, 2016, pp. 426–431, doi: 10.1016/j.ifacol.2016.10.603.

[182] D. Darvas et al., "Formal verification of complex properties on PLC programs," in *Formal Techniques for Distributed Objects, Components, and Systems* (Lecture Notes in Computer Science Series), vol. 8461, E. Ábrahám and C. Palamidessi Eds. Berlin, Germany: Springer, 2014.

[183] D. Darvas, I. Majzik, and E. B. Viñuela, "Formal verification of safety PLC based control software," in *Integrated Formal Methods* (Lecture Notes in Computer Science Series), vol. 9681, E. Ábrahám and M. Huisman Eds. Cham, Switzerland: Springer, 2016.

[184] V. Gourcuff, O. De Smet, and J. M. Faure, "Efficient representation for formal verification of PLC programs," in *Proc. 8th Int. Workshop Discrete Event Syst.*, Ann Arbor, MI, USA, 2006, pp. 182–187, doi: 10.1109/WODES.2006.1678428.

[185] O. Pavlovic and H. D. Ehrich, "Model checking PLC software written in function block diagram," in *Proc. 3rd Int. Conf. Softw. Testing, Verif. Validation*, Paris, France, 2010, pp. 439–448, doi: 10.1109/ICST.2010.10.

[186] S. O. Biha, "A formal semantics of PLC programs in Coq," in *Proc. IEEE 35th Annu. Comput. Softw. Appl. Conf.*, Munich, Germany, 2011, pp. 118–127, doi: 10.1109/COMPSAC.2011.23.

[187] D. Darvas, E. B. Viñuela, and I. Majzik, "A formal specification method for PLC-based applications," in *Proc. 15th Int. Conf. Accel. Large Exp. Phys. Control Syst.*, 2015, doi: 10.18429/JACoW-ICALEPCS2015-WEPGF091.

[188] D. Darvas, I. Majzik, and E. B. Viñuela, "Generic representation of PLC programming languages for formal verification," in *Proc. 23rd Ph.D. Mini-Symp.*, 2016, pp. 6–9.

[189] Y. Huang, X. Bu, G. Zhu, X. Ye, X. Zhu, and J. Shi, "KST: Executable formal semantics of IEC 61131-3 structured text for verification," *IEEE Access*, vol. 7, pp. 14593–14602, 2019, doi: 10.1109/ACCESS.2019.2894026.

[190] B. C. Rawlings, J. M. Wassick, and B. E. Ydstie, "Application of formal verification and falsification to large-scale chemical plant automation systems," *Comput. Chem. Eng.*, vol. 114, pp. 211–220, 2018, doi: 10.1016/j.compchemeng.2017.11.004.

[191] D. F. Bender, B. Combemale, X. Crégut, J. M. Farines, B. Berthomieu, and F. Vernadat, "Ladder metamodeling and PLC program validation through time petri nets," in *Model Driven Architecture—Foundations and Applications* (Lecture Notes in Computer Science Series), vol. 5095, I. Schieferdecker and A. Hartman Eds. Berlin, Germany: Springer, 2008.

[192] E. Brinksma, A. Mader, and A. Fehnker, "Verification and optimization of a PLC control schedule," *Int. J. Softw. Tools Technol. Transfer*, vol. 4, pp. 21–33, 2002, doi: 10.1007/s10009-002-0079-0.

[193] I. Moon, "Modeling programmable logic controllers for logic verification," *IEEE Control Syst. Mag.*, vol. 14, no. 2, pp. 53–59, Apr. 1994, doi: 10.1109/37.272781.

[194] S. Biallas, J. Brauer, and S. Kowalewski, "Arcade.PLC: A verification platform for programmable logic controllers," in *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, Essen, Germany, 2012, pp. 338–341, doi: 10.1145/2351676.2351741.

[195] J. Galvão, C. Oliveira, H. Lopes, and L. Tiainen, “Formal verification: Focused on the verification using a plant model,” in *Innovation, Engineering and Entrepreneurship* (Lecture Notes in Electrical Engineering Series), vol 505, J. Machado, F. Soares, and G. Veiga Eds. Cham, Switzerland: Springer, 2018.

[196] N. Bauer et al., “Verification of PLC programs given as sequential function charts,” in *Integration of Software Specification Techniques for Applications in Engineering* (Lecture Notes in Computer Science Series), vol. 3147. Berlin, Germany: Springer, 2004.

[197] G. Canet, S. Couffin, J. J. Lesage, A. Petit, and P. Schnoebelen, “Towards the automatic verification of PLC programs written in instruction list,” in *Proc. IEEE Int. Conf. Syst., Man, Cybern. ‘Cybern. Evolving Syst., Human, Org., Complex Interact.,’*Nashville, TN, USA, 2000, pp. 2449–2454, doi: 10.1109/ICSMC.2000.884359.

[198] S. Chadwick, P. James, M. Roggenbach, and T. Wetner, “Formal methods for industrial interlocking verification,” in *Proc. Int. Conf. Intell. Rail Transp.*, Singapore, 2018, pp. 1–5, doi: 10.1109/ICIRT.2018.8641579.

[199] M. Niang, A. Philippot, F. Gellot, R. Coupat, B. Riera, and S. Lefebvre, “Formal verification for validation of PSEEL’s PLC program,” in *Proc. Int. Conf. Inform. Control, Autom. Robot.*, 2017, doi: 10.5220/0006418705670574.

[200] M. Rausch and B. H. Krogh, “Formal verification of PLC programs,” in *Proc. Amer. Control Conf.*, Philadelphia, PA, USA, 1998, pp. 234–238, doi: 10.1109/ACC.1998.694666.

[201] J. Yoo, E. Jee, and S. Cha, “Formal modeling and verification of safety-critical software,” *IEEE Softw.*, vol. 26, no. 3, pp. 42–49, May/Jun. 2009, doi: 10.1109/MS.2009.67.

[202] E.-S. Kim, D.-A. Lee, S. Jung, J. Yoo, J.-G. Choi, and J.-S. Lee, “NuDE 2.0: A formal method-based software development, verification and safety analysis environment for digital I&Cs in NPPs,” *J. Comput. Sci. Eng.*, vol. 11, no. 1, pp. 9–23, 2017.

[203] H. Carlsson, B. Svensson, F. Danielsson, and B. Lennartson, “Methods for reliable simulation-based PLC code verification,” *IEEE Trans. Ind. Inform.*, vol. 8, no. 2, pp. 267–278, May 2012, doi: 10.1109/TII.2011.2182653.

[204] L. Garcia, S. Mitsch, and A. Platzer, “HyPLC: Hybrid programmable logic controller program translation for verification,” in *Proc. ACM/IEEE 10th Int. Conf. Cyber-Phys. Syst.*, pp. 47–56, doi: 10.1145/3302509.3311036.

[205] L. Garcia, S. Zonouz, D. Wei, and L. P. de Aguiar, “Detecting PLC control corruption via on-device runtime verification,” in *Proc. Resilience Week*, Chicago, IL, USA, 2016, pp. 67–72, doi: 10.1109/RWEEK.2016.7573309.

[206] A. Mader, “A classification of PLC models and applications,” in *Discrete Event Systems* (The Springer International Series in Engineering and Computer Science Series), vol. 569, R. Boel and G. Stremersch Eds. Boston, MA, USA: Springer, 2000.

[207] D. Bohlender and S. Kowalewski, “Compositional verification of PLC software using horn clauses and mode abstraction,” *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 428–433, 2018, doi: 10.1016/j.ifacol.2018.06.336.

[208] E. V. Kuzmin, A. A. Shipov, and D. A. Ryabukhin, “Construction and verification of PLC programs by LTL specification,” in *Proc. Tools Methods Prog. Anal.*, Kostroma, Russia, 2013, pp. 15–22, doi: 10.1109/TMPA.2013.7163716.

[209] M. Bonfe and C. Fantuzzi, “Design and verification of mechatronic object-oriented models for industrial control systems,” in *Proc. IEEE Conf. Emerg. Technol. Factory Autom. Proc.*, Lisbon, Portugal, 2003, pp. 253–260, doi: 10.1109/ETFA.2003.1248708.

[210] H. Wan, G. Chen, X. Song, and M. Gu, “Formalization and verification of PLC timers in Coq,” in *Proc. IEEE 33rd Annu. Int. Comput. Softw. Appl. Conf.*, Seattle, WA, USA, 2009, pp. 315–323, doi: 10.1109/COMPSAC.2009.49.

[211] H. B. Mokadem, B. Berard, V. Gourcuff, J. M. Roussel, and O. De Smet, “Verification of a timed multitask system with UPPAAL,” in *Proc. IEEE Conf. Emerg. Technol. Factory Autom.*, Catania, Italy, 2005, pp. 347–354, doi: 10.1109/ETFA.2005.1612699.

[212] L. Cheng, K. Tian, and D. Yao, “Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks,” in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, pp. 315–326, doi: 10.1145/3134600.3134640.

[213] Z. Mandić, S. Stankovski, G. Ostojić, and B. Popović, “Potential of edge computing PLCs in industrial automation,” in *Proc. 21st Int. Symp. Infoteh-Jahorina*, East Sarajevo, Bosnia and Herzegovina, 2022, pp. 1–5, doi: 10.1109/INFOTEH53737.2022.9751324.

[214] T. Cruz, P. Simões, and E. Monteiro, “Virtualizing programmable logic controllers: Toward a convergent approach,” *IEEE Embedded Syst. Lett.*, vol. 8, no. 4, pp. 69–72, Dec. 2016, doi: 10.1109/LES.2016.2608418.

[215] D. Levshun, A. Chechulin, and I. Kotenko, “Design lifecycle for secure cyber-physical systems based on embedded devices,” in *Proc. IEEE 9th Int. Conf. Intell. Data Acquisition Adv. Comput. Syst.: Technol. Appl.*, Bucharest, Romania, 2017, pp. 277–282, doi: 10.1109/IDAACS.2017.8095090.

[216] A. Kanellopoulos and K. G. Vamvoudakis, “A moving target defense control framework for cyber-physical systems,” *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 1029–1043, Mar. 2020, doi: 10.1109/TAC.2019.2915746.

[217] P. T. Duy, H. D. Hoang, N. H. Khoa, D. T. T. Hien, and V. -H. Pham, “Fool your enemies: Enable cyber deception and moving target defense for intrusion detection in SDN,” in *Proc. 21st Int. Symp. Commun. Inf. Technol.*, Xi’an, China, 2022, pp. 27–32, doi: 10.1109/ISCIT55906.2022.9931208.

[218] J. A. Giraldo, M. El Hariri, and M. Parvania, “Moving target defense for cyber–physical systems using IoT-Enabled data replication,” *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13223–13232, Aug. 2022, doi: 10.1109/JIOT.2022.3144937.

[219] R. A. Awad, M. H. Rais, M. Rogers, I. Ahmed, and V. Paquit, “Towards generic memory forensic framework for programmable logic controllers,” *Forensic Sci. Int.: Digit. Investigation*, vol. 44, 2023, Art. no. 301513, doi: 10.1016/j.fsidi.2023.301513.

[220] Y. Zhang, Z. Hao, N. Hu, J. Luo, and C. Wang, “A virtualization-based security architecture for industrial control systems,” in *Proc. IEEE 7th Int. Conf. Data Sci. Cyberspace*, Guilin, China, 2022, pp. 94–101, doi: 10.1109/DSC55868.2022.00020.

[221] M. Fu, Z. Wang, J. Wang, Q. Wang, Z. Ma, and D. Wang, “Multicrane visual sorting system based on deep learning with virtualized programmable logic controllers in industrial Internet,” *IEEE Trans. Ind. Inform.*, early access, doi: 10.1109/TII.2023.3313641.

[222] T. Kampa, A. El-Ankah, and D. Grossmann, “High availability for virtualized programmable logic controllers with hard real-time requirements on cloud infrastructures,” in *Proc. IEEE 21st Int. Conf. Ind. Inform.*, Lemgo, Germany, 2023, pp. 1–8, doi: 10.1109/INDIN51400.2023.10218014.

[223] S. Rosioru, V. Mihai, M. Neghina, D. Craciunean, and G. Stamatescu, “PROSIM in the cloud: Remote automation training platform with virtualized infrastructure,” *Appl. Sci.*, vol. 12, 2022, Art. no. 3038, doi: 10.3390/app12063038.

[224] P. Verma, M. P. De Leon, J. G. Breslin, and D. O’Shea, “FedTIU: Securing virtualized PLCs against DDoS attacks using a federated learning enabled threat intelligence unit,” in *Proc. IEEE Int. Conf. Smart Comput.*, Nashville, TN, USA, 2023, pp. 233–236, doi: 10.1109/SMARTCOMP58114.2023.00058.

[225] J. Mellado and F. Núñez, “Design of an IoT-PLC: A containerized programmable logical controller for the industry 4.0,” *J. Ind. Inf. Integr.*, vol. 25, 2022, Art. no. 100250, doi: 10.1016/j.jii.2021.100250.

[226] S. Ivanova and N. Moradpoor, “Fake PLC in the cloud, we thought the attackers believed that: How ICS honeypot deception gets impacted by cloud deployments?,” in *Proc. IEEE 19th Int. Conf. Factory Commun. Syst.*, Pavia, Italy, 2023, pp. 1–4, doi: 10.1109/WFCS57264.2023.10144119.

[227] S. Rashid, A. Haq, S. Hasan, M. Furhad, M. Ahmed, and A. Ullah, “Faking smart industry: Exploring cyber-threat landscape deploying cloud-based honeypot,” *Wireless Netw.*, pp. 1–15, 2022, doi: 10.1007/s11276-022-03057-y.

[228] A. Yao, G. Li, X. Li, F. Jiang, J. Xu, and X. Liu, “Differential privacy in edge computing-based smart city applications: Security issues, solutions and future directions,” *Array*, vol. 19, 2023, Art. no. 100293, doi: 10.1016/j.array.2023.100293.

[229] R. Ramirez, C.-K. Chang, and S.-H. Liang, "PLC cyber-security challenges in industrial networks," in *Proc. 18th IEEE/ASME Int. Conf. Mechatron. Embedded Syst. Appl.*, Taipei, Taiwan, 2022, pp. 1–6, doi: 10.1109/MESA55290.2022.10004463.

[230] R. Ramirez, C.-K. Chang, and S.-H. Liang, "PLC CyberSecurity test platform establishment and cyberattack practice," *Electronics*, vol. 12, 2023, Art. no. 1195. [Online]. Available: https://doi.org/10.3390/electronics12051195

WAEL ALSABBAGH (Graduate Student Member, IEEE) received the B.S. and M.S. degrees in automatic control and computer engineering from Al-Baath University, Homs, Syria, in 2012 and 2015, respectively. He is currently working toward the Ph.D. degree in computer science with the Brandenburg University of Technology (BTU), Cottbus, Germany.

Since 2018, he has been a Scientist with the IHP—Leibniz-Institut für Innovative Mikroelektronik, Frankfurt (Oder), Germany. His research interests include the cyberattacks and security, mitigation methods of the attacks targeting industrial control systems, and supervisory control and data acquisition.

Mr. Alsabbagh is a Technical Paper Reviewer in many conferences and journals, including IEEE ACCESS, IEEE INTERNET OF THINGS, and *Computers and Security*. He is a Member of the IEEE Industrial Electronics Society.

PETER LANGENDÖRFER received the Diploma degree from the Technical University (TU) of Braunschweig, Braunschweig, Germany, in 1995 and the Ph.D. degree from the Brandenburg University of Technology (BTU), Cottbus, Germany, 2001, both in computer science.

Since 2000, he has been with the IHP—Leibniz-Institut für Innovative Mikroelektronik, Frankfurt (Oder), Germany, where he is currently with the Department of Wireless Systems. From 2012 to 2020, he was with the Chair for Security in Pervasive Systems, Technical University of Cottbus-Senftenberg, Cottbus, Germany, where he has also been with the Chair of Wireless Systems since 2020. He has authored or authored more than 150 refereed technical articles and filed 17 patents of which ten have been granted already. His research interests include security for resource constraint devices, low-power protocols, and efficient implementations of artificial intelligence means and resilience.

Dr. Langendörfer was a Guest Editor for many renowned journals, such as *Wireless Communications and Mobile Computing* and *ACM Transactions on Internet Technology*.