



Review

A Flashback on Control Logic Injection Attacks against Programmable Logic Controllers

Wael Alsabbagh ^{1,2,*}  and Peter Langendörfer ^{1,2} 

¹ IHP—Leibniz-Institut für Innovative Mikroelektronik, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany

² Computer Science Department, MINT Faculty, Brandenburg University of Technology Cottbus-Senftenberg, 03046 Cottbus, Germany

* Correspondence: alsabbagh@ihp-microelectronics.com; Tel.: +49-(335)-5625-274

Abstract: Programmable logic controllers (PLCs) make up a substantial part of critical infrastructures (CIs) and industrial control systems (ICSs). They are programmed with a control logic that defines how to drive and operate critical processes such as nuclear power plants, petrochemical factories, water treatment systems, and other facilities. Unfortunately, these devices are not fully secure and are prone to malicious threats, especially those exploiting vulnerabilities in the control logic of PLCs. Such threats are known as control logic injection attacks. They mainly aim at sabotaging physical processes controlled by exposed PLCs, causing catastrophic damage to target systems as shown by Stuxnet. Looking back over the last decade, many research endeavors exploring and discussing these threats have been published. In this article, we present a flashback on the recent works related to control logic injection attacks against PLCs. To this end, we provide the security research community with a new systematization based on the attacker techniques under three main attack scenarios. For each study presented in this work, we overview the attack strategies, tools, security goals, infected devices, and underlying vulnerabilities. Based on our analysis, we highlight the current security challenges in protecting PLCs from such severe attacks and suggest security recommendations for future research directions.

Keywords: industrial control system; programmable logic controller; control logic injection attack; program injection; program modification



Citation: Alsabbagh, W.; Langendörfer, P. A Flashback on Control Logic Injection Attacks against Programmable Logic Controllers. *Automation* **2022**, *3*, 596–621. <https://doi.org/10.3390/automation3040030>

Academic Editor: Quanyan Zhu

Received: 22 September 2022

Accepted: 8 November 2022

Published: 11 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Programmable logic controllers (PLCs) are industrial embedded devices that are programmed with control logic defining how to control and monitor critical processes. They are considered as the core component of any automation pyramid. Thus, their safety, durability, and predictable response times are the primary design concerns [1]. Unfortunately, such devices have been an attractive subject for malicious attackers who aim to disturb physical processes controlled by exposed PLCs. Among the diverse kind of threats targeting PLCs, control logic injection attacks are believed to be the most severe. They compromise the flaws in PLC software or hardware to cause undesired behaviors which eventually might lead to catastrophic consequences for critical infrastructure, as Stuxnet [2], TRITON [3], and Black Energy [4] showed.

In this kind of attack, adversaries engage their engineering software to change either the original control logic running on the target PLC, or to manipulate the program inputs [5]. The malicious injection can be applied either to the high-level source code program (i.e., in one of the five programming languages defined in IEC-61131 [6]), or to the low-level bytecode program (i.e., to the machine code that PLCs read and execute). Another control logic attack scenario can be conducted by crafting specific data values involved in a PLC's program which eventually exploit design flaws in the code. However, these threats have

been a research hot spot and have received increasing attention from security researchers since 2010, and more precisely since Stuxnet [2], which is believed to be the first attack designed with the sole aim of causing physical damage. Stuxnet specifically targeted Iran's uranium facility at the Natanz enrichment plant, and aimed at infecting the control logic of Siemens S7-300 PLCs controlling variable frequency drives of centrifuges. The infected logic control disrupted the normal operation of the drives by changing their motor speed periodically from 1410 Hz to 2 Hz to 1064 Hz and then over again. The Iranian plant was completely isolated from the outside world (air gaped). Therefore, the infection probably took place via a USB stick or a portable programming device. Subsequently, some of the uranium centrifuges (approximately 1000) were destroyed, delaying or preventing uranium enrichment.

Most of the information regarding control logic exploits and vulnerabilities come from the following four sources: the common vulnerabilities and exposures (CVE) (<https://cve.mitre.org/cve/>, (accessed on 18 June 2022)), the Industrial Control Systems—Cyber Emergency Response Team (ICS-CERT) (<https://www.cisa.gov/uscert/ics>, (accessed on 14 June 2022)), the National Vulnerability Database (NVD) (<https://www.nist.gov/programs-projects/national-vulnerability-database-nvd>, (accessed on 11 June 2022)), and the Exploit Database (<https://www.exploit-db.com/>, (accessed on 7 June 2022)). All of these projects are designed to provide lists of publicly disclosed information security vulnerabilities and exposures related to ICSs and information technology (IT) systems. By using the research engine for each, we manually filtered the vulnerabilities as control logic related by matching specific keywords, e.g., PLC, control logic, program injection, program modification, remote code execution, and then going through the description of the resulting vulnerabilities and checking online the documentation of each affected PLC. Afterward, we extracted the number of the vulnerabilities reported per year, and finally generated a statistical report from 2010 (i.e., the first control logic related vulnerability was reported) to the end of 2021, as presented in Figure 1. The rapid increase over the years that our report shows is reflected, above all, by the steadily growing number of publications and the popularity of the PLC security in academic research. Precisely, this occurred after 2011, when Beresford [7]—the most cited work in this area—presented a roadmap for threats against PLCs. Since then, there have been extensive academic studies investigating control logic injection attacks.

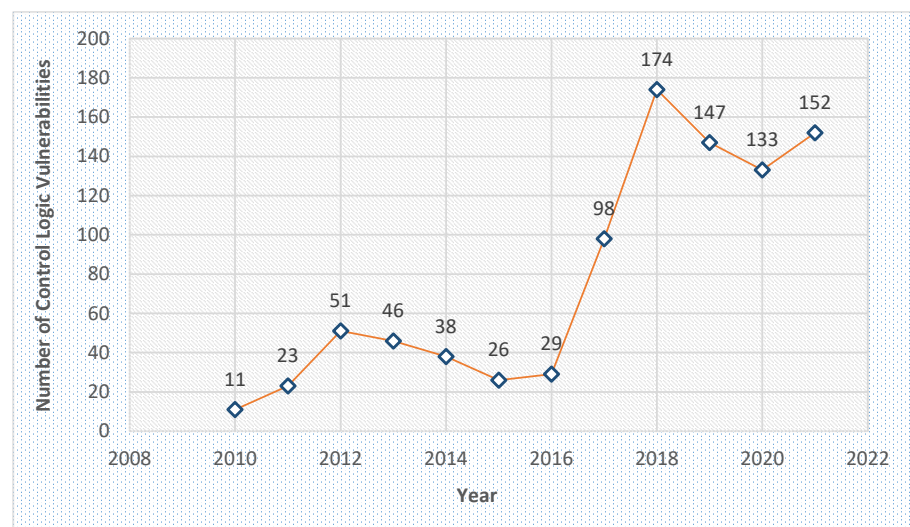


Figure 1. The number of related control logic vulnerabilities reported per year.

1.1. Motivation

Although our report clearly proves that studies on control logic injection attacks are certainly extensive, we admit that systematizing all the control logic injection attacks discussed in the previous works in one article is not an easy task. Our investigations showed that an attacker performs a control logic injection on a target PLC through three main scenarios.

- An attacker aims at modifying PLC codes, i.e., the control logic programs that PLCs run. This scenario represents a typical control logic injection attack.
- An attacker manipulates specific input data to exploit the flaws of a PLC program. Such a scenario is known as a control logic manipulation attack.
- An attacker corrupts the PLC's memory in order to sabotage the execution process of a control logic. This scenario is called a control logic corruption attack.

These three attack scenarios differ from each other in the security goals that are broken, underlying vulnerabilities, attacking tools that are used, etc. For all that, we are convinced that the research community needs a new systematization that assists in understanding the current research progress, raising awareness for such sophisticated attacks and eventually identifying security challenges for future research directions.

1.2. Scope of the Article

In past years, precisely between 2010 and 2021, thousands of academic works discussing control logic injection attacks against ICSs and their PLCs have been published. Most of them are publicly available and accessible on different academic research databases and digital libraries, e.g., IEEE Xplore, ACM, DBLP, Science Direct, SpringerLink, Microsoft Academic, etc. By using the research engine of each, we manually extracted the publications as control logic injection related by using certain keywords, e.g., PLC, control injection, control modification, control manipulation, and control corruption. Then, we sorted them based on the year of publication as shown in Figure 2.

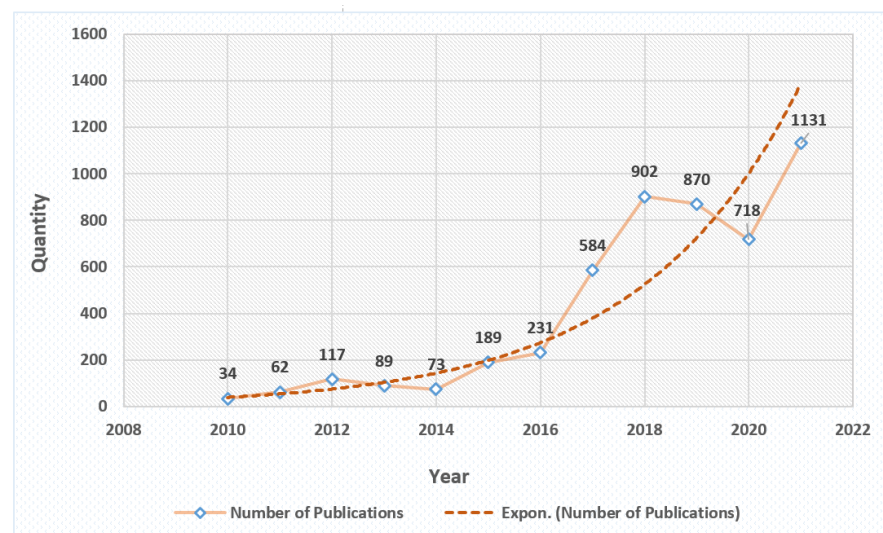


Figure 2. The number of related control logic injection attack publications per year.

Please note that the number of publications includes different kinds of academic works, e.g., reports, white papers, conference papers, and scientific magazine and journal articles. As can be seen in Figure 2, the number of publications related to control logic injection attacks is quite large (over 5000 contributions). However, in this article, we selected only a short list of 36 publications. This is for a better overview and consistent presentation. In the following, we illustrate the criteria and terms we considered in our selection for the literature of this article.

- The study investigates control logic injection attacks. Our focus is on attacks that targeted the PLC's program by modifying the program payload, injecting malicious data to trigger program design flaws, or corrupting the PLC's memory to confuse the execution of the control logic. To achieve a realistic survey, we exclude in this article all the works that are not tested on real hardware and industrial settings.
- The study plays an impactful role in the research community. We discuss top cited papers in this area, showing the very new vulnerabilities that the authors of these papers exploited in order to modify the control logic of different types of PLCs. To be more accurate, we exclude all the works that have less than 20 citations unless they present novel attack approaches, or reveal new vulnerabilities.
- The study discovers a new direction for future research. We take into consideration the papers that suggest new paths for further research or potential security directions to be covered.

1.3. Contributions

Our main contributions in this work are highlighted as follows:

- To the best of our knowledge, this is the first survey paper to provide a comprehensive study of different control logic attack methods against PLCs. We hope that our article can trigger more research activities in this domain.
- We survey cyberattacks and their impacts on different PLCs from different vendors.
- We provide a new systematization of the recent studies based on three attack scenario models.
- We identify the current security challenges in protecting PLCs from such attacks.
- We suggest future research paths to protect PLC-based environments from control logic injection threats.

The rest of the paper is organized as follows. Section 2 introduces the basic background of a standard PLC. We present our systematization methodology in Section 3, and we discuss and analyze the current studies in Section 4. In Section 5, we present the mostly common mitigation solutions that PLC vendors recommend their consumers implement. The current security challenges to defend against this threat is introduced in Section 6, and Section 7 concludes this paper.

2. Background

In this section, we give an overview of the architecture of a standard PLC, its runtime environment, and control logic.

2.1. PLC Architecture

A PLC is a device programmed on a digital basis that is used to control machines, industries, and plants. Figure 3 shows the architecture of a standard PLC. In the simplest case, it has a power supply, input and output modules, an operating system (OS), memory such as RAM or EEPROM, and an interface to upload/download the user program to/from the engineering station. The OS, as well as the user-specific program, is stored in the EEPROM. Input devices (e.g., sensors, switches, etc.) provide the current state of the physical process to the PLC, which the PLC processes through its control logic, and drives the physical process accordingly via output devices (e.g., motors, valves, etc.).

2.2. PLC Runtime Environment

PLCs run a real-time OS that is designed to process specific tasks by cyclic repetition-defined command sequences in very short periods of time. Each execution cycle, namely as a scan cycle, is comprised of four main steps, as seen in Figure 3. First, the CPU reads the inputs from the connected sensors and saves the reading values to a data table or an input image. Then, the logic execution updates the inputs of the running program with the new values. Afterward, the control logic is executed, and the output statuses are updated accordingly. The fourth step is dedicated to the communication tasks. During this period,

the realization of any data exchange with devices connected to the PLC is implemented. After the communication scan, the OS brings the PLC to the maintenance phase. This phase encompasses the update of the internal clocks and registers, performing memory management and other tasks related to a system maintenance. The user is not informed about this maintaining sequence, but it regularly runs in the background as an essential part of the scan cycle.

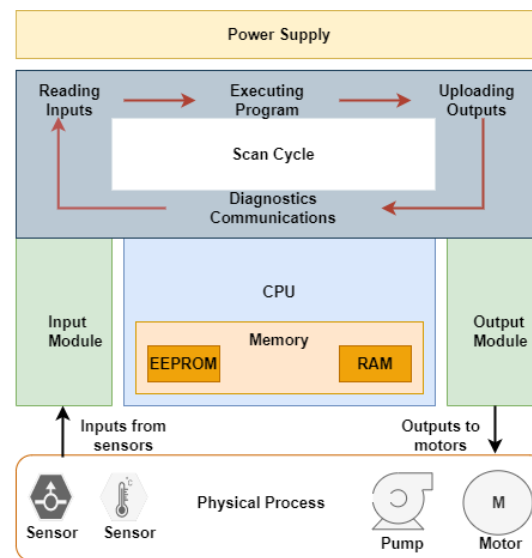


Figure 3. A typical PLC architecture.

2.3. PLC Control Logic

A PLC control logic, known also as a user program, is a code developed by an ICS operator to define how a PLC should operate. It is structured by programming units (blocks) that contain the required instructions to control the PLC and thus maintain the industrial process at a desired state. An ICS operator writes a PLC control logic in a high-level format (source code) by using a vendor-specific engineering software that compiles the source code into its lower representation (binary/bytecode), which the PLC reads and processes. In the following, we give an overview of both PLC program formats.

2.3.1. Source Code

PLCs are offered by several vendors such as Siemens, Allen–Bradley, Mitsubishi, Schneider, and many others. All controllers are programmed with one of the five programming languages defined in IEC-61131 [6] as follows: ladder diagram (LD), structured text (ST), sequential function chart (SFC), instruction list (IL), and function block diagram (FBD). Engineers develop PLC programs by using vendor-specific software that contain standard compliant or integrated development environments (IDEs) and compilers. Some PLCs support, in addition to the five mentioned languages, computer-compatible languages (e.g., BASIC, C, and assembly), proprietary-developed languages (e.g., Siemens GRAPH5 (Simatic S5 PLC. <https://en.wikipedia.org/wiki/SimaticS5PLC> (accessed on 23 March 2022)), and specific language (e.g., boolean logic (<https://www.plcmanual.com/plc-programming> (accessed on 19 March 2022))).

2.3.2. Machine Code

An engineering software compiles a high-level format of a user program, i.e., it compiles a source code to a bytecode or binary depending on the PLC's vendor. For instance, Siemens S7 PLCs compile their source codes to MC7 bytecodes, whereas CODESYS devices compile their source codes to binaries. The format of the PLC bytecode or binary is not documented and often proprietary for each PLC vendor. Consequently, any deeper exploration or analysis always demands reverse-engineering mechanisms.

3. Systematization Methodology

In this section, we first present an example application that helps in understanding the three attack scenarios that our systematization is based upon. Then, we describe the most common control logic vulnerabilities followed by the security goals that attackers aim to break.

3.1. PLC Example Application

Figure 4 shows a simple example application in which a PLC controls the movement of objects on a conveyor.

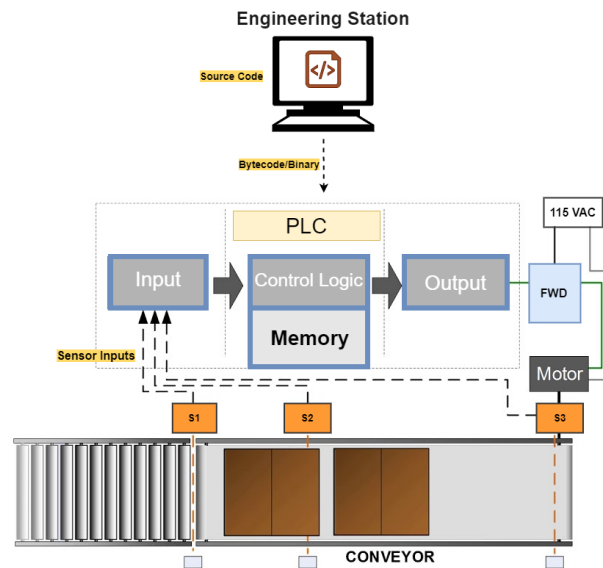


Figure 4. Example application.

For the given example, there are three sensors placed along the conveyor to define the exact locations of the cartons, and how many cartons are currently on the conveyor. In a normal operation case, if an object arrives to the conveyor, the first sensor (S1) reports to the PLC by sending an input value to the input module. The PLC, in the next execution scan cycle, reads this value and processes the control logic that is stored in the PLC's memory. Afterward, the PLC sends an appropriate control command to the output modules, i.e., it switches on the electrical switch (FWD) that is connected to the motor. The motor's rotation results in synchronized movement of the belt, and the looped movement of the belt helps in transporting the object forward. For an abnormal case, when a new object arrives and the conveyor is full, i.e., sensors S1, S2, and S3 are activated, the PLC switches the motor off, and the belt stops running. This process must have the right timing or else the movement sequence of the objects will be disrupted.

3.2. Control Logic Injection Scenarios

Most of the previous works discussing control logic injection threats are achieved through one of three attack scenarios. In the following, we describe these scenarios with the help of our given application in Figure 4.

3.2.1. Scenario 1—Attackers Have Access to the Engineering Station

In this scenario, studies assumed that attackers have access to the engineering station as shown in Figure 5. For such a scenario, attackers are able to access the source code program that is developed in a high-level programming language. Therefore, all injection attacks are performed by directly altering the source code stored on the engineering software. Here, attackers could be either internal, i.e., who already have access to the engineering station,

or external, i.e., who can bypass specific vulnerabilities [8,9] in the engineering station to gain access.

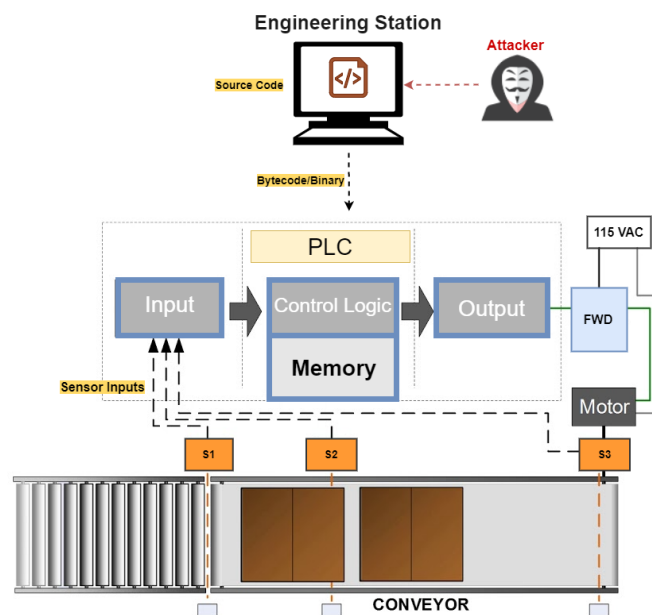


Figure 5. Attack scenario 1.

3.2.2. Scenario 2—Attackers Have Access to the Control Network

In this scenario, the engineering station is not reachable for attackers, but they can access the control network as shown in Figure 6. For such a scenario, adversaries aim at intercepting, modifying the transmission, and leveraging vulnerabilities in the network communication [10–12]. This means that they use packet-sniffing tools such as Wireshark (<https://www.wireshark.org/> (accessed on 3 May 2022)) to record the network traffic exchanged between the engineering station and the victim PLC(s). Then they extract the program bytecode/binary from specific packets and maliciously alter the control logic in one of the following techniques.

- Modifying the control logic code in its decompiled format. To this end, attackers first reverse-engineer the program bytecode/binary, and then modify the decompiled code in its high-level source code. Afterward, they recompile the modified code to its low-level binary code and push it to the victim PLC through a network packet.
- Modifying the control logic code in its compiled format. Here, attackers intercept the packets that contain the program bytecode, and replace the exact original bytes representing the code with malicious ones. The modified bytes are either identified by using a reverse-engineering method beforehand or prerecorded from another session prior to the attack. After the modification, attackers push the crafted packet to the PLC over the network.

3.2.3. Scenario 3—Attackers Have Access to the PLC’s Runtime Environment

Figure 7 depicts this scenario. Attackers here have neither access to the engineering software nor to the payload transmitted between the stations. However, they can still speculate about the logic of the control program by reaching the PLC’s runtime environment including the PLC firmware, memory, and I/O traces. For this scenario, attackers aim at either modifying the real-time sensor inputs to the program or corrupting the PLC’s memory where the program is stored, which eventually results in executing the control logic inappropriately.

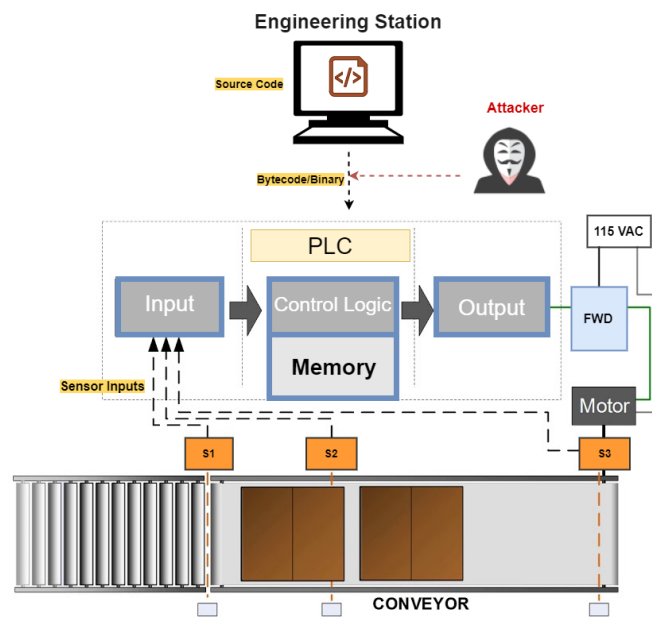


Figure 6. Attack scenario 2.

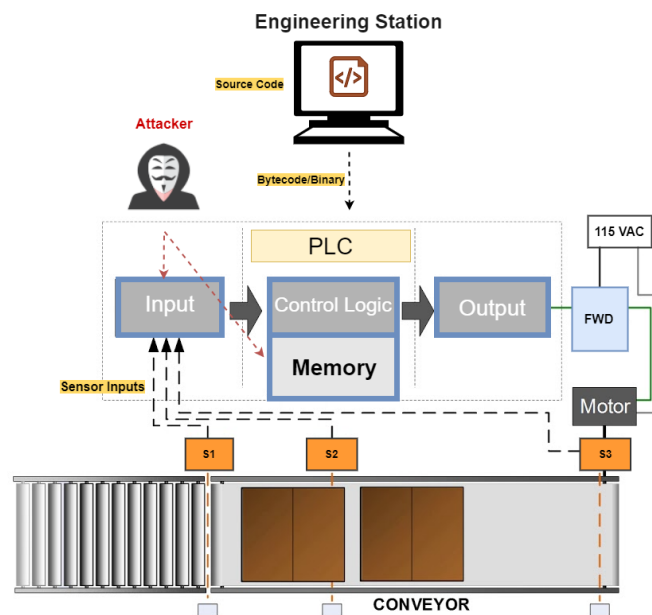


Figure 7. Attack scenario 3.

3.3. Control Logic Vulnerabilities

Attackers usually seek entry points in the target system and then leverage existing vulnerabilities and weaknesses in the PLC(s) for achieving successful attacks. In the following, we illustrate the most common related control logic vulnerabilities that have been disclosed in previous studies.

3.3.1. V1—Race Condition

A race condition scenario occurs when two or more threads try to access a shared resource at the same time. It happens mostly when an output variable depends on multiple sensor readings. For our given example in Figure 4, exploiting a race condition vulnerability occurs when one sensor reports to the PLC that objects arrive to their zones faster or slower

than it must be, at a configured moment, so the PLC will send a false control command to the motor, causing an undesired state.

3.3.2. V2—Variables without Use

This vulnerability is exploited when a variable's value is assigned but never used in the PLC's program. This means that if an input variable is uninitialized, attackers can provide an illegal value for it in a running program. Similarly, attackers can leverage unused output variables to produce a certain behavior. For our example application, this vulnerability happens if the user assigns new input variables for the sensors, i.e., S1, S2, and S3 without removing the previous variables. Thus, an attacker can initialize one input variable to a malicious value so the PLC will read and process during the running program.

3.3.3. V3—Hidden Jumpers

A hidden jumper could involve either bypassing a portion of the program forcing a jump to an empty branch, or jumping to another branch on a certain condition the attacker configures. Attackers can exploit this vulnerability by embedding a malware in the bypassed portion of the program. For our example application, an attacker can create a force by activating a mechanism within the PLC which makes the PLC override certain elements, e.g., the portion where a PLC updates the sensor readings.

3.3.4. V4—Improper Input Validation

In this vulnerability, attackers can craft illegal input values based on the types of the input variables to cause unsafe behavior. For example, attackers can send a specially crafted packet to the PLC, providing an input index that is out of the bound of an input array. Exploiting this vulnerability may cause the target PLC to become inaccessible from the network, i.e., denial of service (DoS) condition.

3.3.5. V5—Predefined Hierarchical Memory Layout

PLCs usually address the specific programming format of a storage class, e.g., *I* for input, *Q* for output, a data size (e.g., *X* for BOOL, and *W* for WORD), and a hierarchical address designating the location of the corresponding port. Adversaries can exploit this format to predict the variables during the running program.

3.3.6. V6—Real-Time Constraints

Each execution cycle should accomplish its task within a preconfigured maximum cycle time to ensure the real-time execution. However, for programs in which tasks are executed without being interrupted, i.e., in non-preemptive multitask programs, one task has to wait for the completion of another task before starting the next scan cycle. To launch synchronization attacks, adversaries can generate loops or flood the PLC with a large number of I/O operations to enlarge the execution time slice.

It is worth mentioning that the vulnerabilities V1, V2, and V3 are usually compromised in attack scenario 1. This is due to the fact that attackers in scenario 1 can involve these vulnerabilities as bad coding practices, without negatively impacting the core control logic. The other vulnerabilities are mostly exploited in scenarios 2 and 3 as shown later in the following section.

3.4. Security Goals

The security goals, which previous research aimed at breaking by conducting control logic injection attacks, are related to the security properties of the CIA model [13], i.e., confidentiality, integrity, and availability. In the following, we describe each security property in more detail.

3.4.1. Confidentiality

Confidentiality is a security property that ensures that critical information is only disclosed to authorized entities. There is a vast amount of sensitive information in PLCs, e.g., the PLC vendor, media access control (MAC) address, I/O data from sensors and actuators, etc. The attacks violate confidentiality of PLCs by stealthily monitoring the execution of PLC programs leveraging existing vulnerabilities (e.g., V2 and V3).

3.4.2. Integrity

It is a security property that prevents unauthorized modification or introduction of information by unauthorized users or systems. The attacks violate integrity by forcing PLCs to cause unsafe behaviors for the physical process by leveraging mostly the vulnerabilities V4 and V5.

3.4.3. Availability

It is a security property that ensures that the data, system, or device is accessible when required. Loss of availability could undermine the industrial process as timeliness of real-time information is often crucial for the control process. The attacks violate availability by exhausting PLC resources (e.g., memory or processing power) and causing a DoS condition.

4. Control Logic Injection Attacks

This section describes the control logic injection attacks that existing studies discussed. We summarized all the studies in Table 1.

Table 1. The studies investigating control logic injection attacks.

Contribution' Year	Attack Scenario	Vulnerability	Security Goal	PLC Vendor/ Language	Tool
Serhane'18 [14]	1	V1,2,3	C,I,A	Rockwell/LD	N/A
Valentine'13 [15]	1	V1,2,3,6	C,I	Rockwell/LD	PLC-SF
McLaughlin'14 [16]	1,3	V4	I	N/A/generic	CaFDI
McLaughlin'11 [17]	2	V4	I	Siemens/LD	N/A
McLaughlin'12 [18]	2	V4	I	Siemens/LD	NuSMV
Senthivel'18 [19]	2	V4	I	Rockwell/LD	Laddis
Keliris'19 [20]	2	V4	I	N/A/generic	ICSREF
Kella '19 [21]	2,3	V4	I	Schneider/IL	CLIK
Qasim'20 [22]	2	V4	I	Schneider/IL	Reditus
Alsabbagh'21 [1]	2	V4	I,A	Siemens/ST	N/A
Alsabbagh'21 [23]	2	V4	I	Siemens/LD	N/A
Beresford'11 [7]	2,3	V4,5	I	Siemens/LD	Metasploit
Klick'15 [24]	2,3	V5	I	Siemens/IL	PLCinject
Spennenberg'16 [25]	2,3	V5	C,A	Siemens/IL	PLC-Blaster
Lei'17 [26]	2	V5	I	Siemens/N/A	Spear
Biham'19 [27]	2	V5	I	Siemens/N/A	Rogue7
Hui'18 [28]	2	V5	C, I	Siemens/N/A	N/A
Hui'21 [29]	2	V5	C, I	Siemens/N/A	N/A
Alsabbagh'21 [30]	2	V3,5,6	I	Siemens/ST	PLCinject
Alsabbagh'22 [31,32]	2	V3,5,6	I	Siemens/ST	PLCinject
Basnight'13 [33]	3	V4	I	Allen-Bradley/N/A	N/A
Basnight'13 [34]	3	V4	I	Allen-Bradley/N/A	N/A
Peck'09 [35]	3	V4	I	Koyo, Rockwell/N/A	N/A
Schuettt'14 [36]	3	V4	I	Allen-Bradley/N/A	N/A
Rais'21 [37]	3	V4	I	Allen-Bradley/N/A	Kyros
Garcia'17 [38]	3	V4,5	I,C	Allen-Bradley/N/A	HARVEY
Lim'17 [39]	3	V4,5	I,A	Schneider/N/A	N/A
Wang'18 [40]	3	V4,5	I,A	Siemens/N/A	N/A
Yoo'19 [41,42]	3	V5	I,A	Schneider, Allen-Bradley/N/A	DPI
Govil'17 [43]	3	V4,6	I	Allen-Bradley/LD	LLB
Xiao'16 [44]	3	V4	I	N/A/generic	N/A
Alsabbagh'21 [45]	3	V1,4,6	I	Siemens/N/A	N/A
Noorizadeh'21 [46]	3	V1,4,6	I	Siemens/N/A	N/A
Abbasi'16 [47]	3	V4	I	Generic/Codesys-based	Codesys platform

Confidentiality (C), Integrity (I), Availability (A), Not Available (N/A).

4.1. Attack Scenario 1—Source Code Injection

For this type of attack, adversaries have access to the engineering station. Therefore, attackers at the source code level aim at performing a stealthy injection in a way that no remarkable changes would be applied to the fundamental functionality of a PLC's program or masked as novice programmer mistakes. This means that attacks could be concealed as unintentional bad coding practices, as shown in [14,15]. The authors of those works focused on attacks against graphical languages, e.g., LD, because tiny modifications of such languages could not be easily observed.

Serhane et al. [14] explored LD code vulnerabilities and bad code practices that may become the root cause of bugs and subsequently be exploited by attackers. The authors showed that attackers could uncertainly generate tumbling output variables, e.g., forcing two timers to control a certain output value could cause a race condition (V1). Such a scenario might result in serious damage to the devices infected, similar to Stuxnet [2]. Another scenario that the authors pointed out is that adversaries can compromise some functions and set specific operands at desired values, or introduce empty branches/jumps (V3). In order to achieve a stealthy modification, attackers could use instruction sets or determined commands to insert false parameters and values (V2). They also discussed the possibility of applying infinite loops via jumps and used timers to activate the branch containing the malicious instructions only at a time determined by an attacker. This scenario could slow down or even halt operating the device under attack in a severe manner.

Valentine [15] introduced an attack scenario in which an adversary installed a jump to a subroutine function, and changes the intercommunication between two or more ladders in an LD code (V3). The author showed that an attacker with access to an engineering station could use a JMP command to insert his own malicious code at the mislabeled location and cause multiple errors before the RTN command returns the code to the intended location. This leads the PLC to run the original control logic program up to the point of the JMP, read the inserted code, and then continue along the original path, with the possibility of introducing a new set of parameters.

In addition to these works, McLaughlin et al. [16] introduced a control logic attack to confuse the behavior of a PLC. The authors analyzed an I/O trace of the exposed PLC to produce a set of inputs to achieve the desired PLC outputs. Their attack is comprised of two steps. First, they speculated on what the control logic looks like by constructing a model of the PLC's internal logic from the captured I/O traces. Then, they searched for a set of inputs that caused the model to calculate the desired malicious behavior. Their attack was evaluated against a set of representative control systems, proving that it is a feasible threat against insecure sensor configurations. Please note that [16] can also be classified among those attacks grouped in scenario 3. This is because the manipulation happens at the PLC's input level. However, because the authors assumed in their study that an attacker has access to the engineering software and can read the source code, we have elaborated upon this work in scenario 1.

4.2. Attack Scenario 2—Machine Code Injection

Studies that are grouped under the second scenario mostly discussed control logic reverse engineering and injection attacks. Instead of concealing the infection as a bad coding practice like those in scenario 1, the injection at the machine code level intends to sneakily avoid behavior-based detection methods. However, in the following we subgroup the studies of this scenario into two main groups based on the format of the control logic program attacked.

4.2.1. Decompiled Code Injection Attack

Here, adversaries generate control logic attacks by first reverse-engineering the program in its bytecode/binary format, then modifying the decompiled code, recompiling the modified code, and finally pushing the malicious code to the PLC over the network. Such attacks happen during a payload transmission from engineering stations to PLCs or the

other way around. Studies in this group investigated program reverse-engineering which is the prime challenge of such an attack. This is due to the fact that several PLC features are not supported in normal instruction sets [48,49]. However, existing studies like [1,17–23] discussed modifying the decompiled code by different reverse-engineering methods.

McLaughlin [17] conducted an injection attack on a train interlocking program. The malicious program he introduced was reverse-engineered by utilizing a decompiled-configuration code. With the help of this code, he extracted the field-bus ID that is dedicated to the PLC model and vendor. Afterward, he retrieved hints about the structure of the physical process. Based on the disclosed clues, he designed his own program, which generates unsafe behaviors for the train, e.g., resulting in conflict states for the train signals. As a real attack scenario, he targeted timing-sensitive signals and switches. Furthermore, the author managed successfully to overcome the security solution presented in [50], and made changes in the program without alarming the detector. In a follow-up work, McLaughlin et al. [18] implemented a so-called SABOT attacking tool. It required a high-level illustration of the control logic. For example, the target system includes two ingredient valves, as well as one drain valve. This kind of description can be obtained from public channels, and is quite similar to the one used by other systems running in the same industrial sector. With the help of this knowledge, SABOT created a specific behavior for the control processes and employed a gradational checking model to find a mapping between a variable/parameter involved in the program and a determined physical process. Based on the pairs of mappings, SABOT compiled a control logic code customized for the industrial process. McLaughlin in both of his studies leveraged the vulnerability V4 of the control logic targeting the security goal I. However, his attacks were limited to Siemens PLCs, and the author did not provide a sufficient illustration on the reverse-engineering method he used.

The authors of [19] introduced a manual reverse-engineering approach to develop a decompiler called Laddis, which can decompile the low-level binary of a ladder logic to a higher-level presentation. Laddis was used to perform three control logic injection attacks, in which an attacker interferes with the engineering operations of downloading and uploading PLC control logic. In the first attack scenario, an attacker, placed in a man-in-the-middle position between a target PLC and its engineering software, injects malicious control logic to the PLC and replaces it with a normal (original) control logic to deceive the engineering software when the uploading operation is requested. The second scenario that their paper presented is very similar to the first scenario but differs in that an attacker uploads malformed control logic instead of the original control logic to crash the engineering software. The last scenario does not require a man-in-the-middle position, as the attack just injects crafted, malformed control logic to the target PLC. The Laddis decompiler can decompile the ladder logic program from the network traffic in both directions, i.e., upload and download, but it was limited to the programmable controller communication commands (PCCC) protocol and Allen–Bradely RSLogix 500 engineering software. The authors targeted the security goal I to stay stealthy and leveraged the vulnerability V4 of the control logic to achieve a successful attack.

Keliris et al. [20] presented a new open-source decompiler, namely ICSREF, that reverse-engineers a CodeSys-based control logic, and creates malicious payloads. The authors showed that skilled attackers can enable dynamic process-aware payload generation, lowering the requirements for malicious actors, and allowing sophisticated attacks against even air-gapped systems without requiring prior knowledge. ICSREF leveraged the vulnerability V4 and aimed at manipulating the PID controller functions and parameters, e.g., set points, proportional/integral/derivative gains, initial values, etc. ICSREF deduced the physical features and characteristics of the physical process, so that substituted binaries could prevail in more severe attacks.

Kalle et al. [21] demonstrated a remote attack, namely CLIK, on the control logic of Schneider PLCs, precisely Modicon M221 and its vendor supplied engineering software (SoMachine-Basic). The goal of CLIK is to introduce malicious logic in a target PLC automatically. CLIK involves compromising the PLC security measures and stealing the

control logic from a PLC, decompiling the stolen binary of the control logic to inject the malicious logic, and then transferring the infected binary back to the PLC. As a part of this work, the authors developed a decompiler called Eupheus that transforms a low-level control logic in RX630 microcontroller instructions [51] to a high-level IL program. CLIK leveraged the vulnerability V4 of the control logic and targeted the security goal I, but it was limited to Modicon M221 PLCs.

A group of researchers introduced a control logic forensics framework for control logic injection attacks called Reditus [22]. It extracts and decompiles the control logic from a network traffic dump automatically without any manual reverse-engineering or prior knowledge of ICS protocols. Reditus is based on the observation that an engineering software can read a control logic from a PLC remotely referred to as an upload function, and has a built-in decompiler that can further transform the control logic into its source code. The authors used Reditus to leverage the vulnerability V4 and perform a control logic injection attack scenario on only Modicon M221 PLC and its SoMachine-Basic engineering software.

Alsabbagh et al. [1] showed that it is possible to retrieve the bytecode from a target S7 PLC, and to decompile the bytecode to its high-level source code, e.g., ST source code. The authors created a mapping database of pairs: bytecode to the corresponding ST instructions in order to transform the bytecode to its source code format. Afterward, they conducted a typical control logic attack showing that even a very tiny modification in the code is sufficient to harm the target system. Similar to Kalle's work [21], the authors generated a fake PLC impersonating a real S7 PLC to perform a fully stealth injection attack. They tested both attack scenarios on real industrial settings by using S7-300 PLCs. In a follow-up work, Alsabbagh et al. [23] discussed an approach that allows an attacker to modify a PLC program in its source code format. The authors first retrieved the bytecode by recording the transmitted packets between the victim PLC and the engineering software. Then, they employed a reverse-engineering-based decompiler to convert the stolen bytecode to its decompiled version, e.g., LD program. Afterward, the authors exposed the structure and semantic of the control logic, and modified the program to cause abnormal behavior to the system they targeted. Finally, they recompiled it to its low-level format, i.e., bytecode before pushing it back to the victim PLC. The authors in both studies targeted the security goals I and A, and leveraged the vulnerability V4 to conduct a successful control logic injection. However, their attacks were only tested on Siemens PLCs and S7Communication injection protocol.

4.2.2. Compiled Code Injection Attack

This kind of attack involves modifying the original control logic running on a target PLC by engaging the attacker engineering software, typically employing a man-in-the-middle approach. Several studies discussed hijacking certain network packets between the engineering station and PLCs. Beresford [7] exploited packets (e.g., ISO-TSAP) between PLCs (Siemens S7-300 and S7-1200 PLCs) and their engineering software, i.e., totally integrated automation (TIA) portal. These packets contain critical data, e.g., variable names, data block names, and also the PLC's model and vendor. The author mapped the values extracted from the memory dumps to their corresponding variables in the PLC, and could successfully manipulate these variables to cause undesired behavior. Beresford tested his attacks only on S7 PLCs and its S7Communication protocol, revealing new vulnerabilities V4 and V5 in S7 PLCs, and targeting the security goal I. However, Beresford made the entry point through the network traffic, ignoring the fact that security measures could have enabled deep packet inspection (DPI) between the PLC and the engineering station.

In 2015, Klick et al. [24] presented an injection of a malware into the control logic of an Siemens S7 PLC, precisely S7 314C-2 PN/DP, without disrupting the service. The authors showed that a knowledgeable adversary with access to a PLC can download and upload code to it, as long as the code consists of an MC7 bytecode. They introduced a so-called PLCinject tool which crafts a payload with a simple network management protocol (SNMP) scanner and proxy. Their investigations disclosed a vulnerability existing in the predefined memory layout of SIMATIC PLCs (V5); this allows attackers to place a malicious payload at

the beginning of the main organization block (OB). This security gap changes the execution sequence of the control logic and turns the PLC into a gateway mode, i.e., the security goal I was broken. Because the authors assumed that attackers are already familiar with the PLC's memory layout, this study can be also grouped among the attacks in scenario 3.

By using PLCinject, Spenneberg [25] described a worm which can spread out from one PLC to another by copying itself, and adapting the next target PLC to execute the worm along with the running control logic. The worm was designed by using a state machine in which the current state is stored in a global variable. At the start of each cycle, the appropriate code in the worm is called. Thus, the maximum cycle time is never violated. PLC-Blaster utilized a variety of antidetection mechanisms, e.g., the worm evaded the antireplay byte method, was stored in the less-used block of the running program on the target PLC, and could meet the maximum scan cycle limit. This severe attack implemented an endless loop triggering an error condition within the PLC with the impact of denial of service condition. Please note that this work can be also presented with the attacks in scenario 3 as the author assumed that an attack has access to the PLC's run-time environment, leveraging the vulnerability V5 and targeting the security goals C and A.

Lei et al. [26] demonstrated a worm that can break the security wall of the S7CommPlus protocol that Siemens SIMATIC S7-1200 PLCs utilize. The authors first used Wireshark software to analyze the communications between the TIA Portal software and S7 PLCs. Then, they applied reverse debugging software (WinDbg (<http://windbg.org/>, accessed on 14 July 2022)) to break the encryption mechanism of the S7CommPlus protocol. Afterward, they demonstrated two attacks. First, a replay attack was performed to start and stop the PLC remotely. In the second attack scenario, the authors manipulated the input and output values of the victim, causing serious damage for the physical process controlled by the target PLC. The presented worm in this work was designed to attack only specific PLCs, i.e., S7-1200, and its S7CommPlus protocol exploiting the vulnerability V5, and targeting the security goal I.

The researchers behind Rogue7 [27] created a rogue engineering station that can impersonate the TIA portal to the latest Siemens S7 PLCs line, running various firmware versions and using the latest S7CommPlus protocol. Their so-called Rogue7 tool can perform several attacks against S7-1500 PLCs. They first performed a typical start/stop attack, and then downloaded a replayed control logic program to the target PLC. In their most advanced attack—the stealth program injection attack—they managed successfully to maintain the source program as the engineer expects to see it, while programming the victim PLC to run a malicious bytecode that the engineer will never see. They achieved their attacks through a detailed reverse-engineering analysis, as well as the key generation and cryptographic primitives. As a part of their research, they analyzed versions of the S7 protocol running on different generations of PLCs, e.g., 1200 and 1500 family. The authors targeted the security goal I exploiting the vulnerability V5.

Hui et al. [28] investigated different potential exploits against Siemens S7-1211C PLCs and their TIA portal software. The authors used the WinDbg and Scapy (<https://scapy.net/>, (accessed on 8 May 2022)) tools in their investigations and showed that the antireplay mechanism of S7CommPlus protocol is vulnerable. Based on their findings, they managed to perform several attacks against the tested PLC, e.g., session stealing, phantom PLC, cross-connecting controllers, and a control logic injection attack. They proved that if adversaries could obtain information about the antireplay mechanism and 20-byte integrity check that S7 PLCs use, the possibility of executing a malicious payload is very high even without being observed by ICS operators. In a follow-up work, Hui et al. [29] analyzed and identified specific necessary bytes to craft valid network packets, and demonstrated a successful replay attack on S7-1200 PLCs. The presented attacks in this work opened the door for attackers to manipulate the control logic that the victim PLC runs. However, both works were limited to S7 PLCs and did not take any security means that might be implemented into account. The authors exploited the vulnerability V5 in S7 PLCs, and targeted the security goals C and I.

The researchers behind [30] presented a novel control logic injection attack strategy that allows adversaries to patch their malicious codes once they access exposed PLCs, keeping their attack in idle mode inside the infected device, and then activate the attack at a later time without even being connected to the target PLC on the attack date. The authors developed the PLCinject tool [24] to inject an S7-300 PLC with an interrupt block, i.e., a time-of-day (ToD) organization block OB10. The attacker's block is located at the very beginning of the main code (OB1) and the CPU checks whether the condition of the interrupt is met in each single execution cycle. This means that the attacker's code will always be checked but only executed when the date and time of the CPU's clock matches the date and time set by the attacker. However, their approach has limitations. The attack was only tested on S7-300 PLCs and its S7Comm protocol. Furthermore, the ICS operator could reveal the infection in the control logic by uploading and comparing both programs, the one running on the PLC with the one stored on the engineering software. In follow-up works, Alsabbagh et al. [31,32] took their attack approach one step further and extended their experiments to involve the newest S7-1500 PLCs that use the S7CommPlus protocol. The authors managed to overcome the limitations of their previous work [30] and could hide the malicious interrupt code in the PLC's memory until the very moment determined by the attacker. They also revealed two vulnerabilities in the integrity protection method that S7-1500 PLCs and their S7CommPlus protocol use. However, their attacks were only tested on Siemens devices, and exploited the vulnerabilities V3, V5, and V6 targeting the security goal I.

4.3. Attack Scenario 3—Runtime Environment Injection

At this level, existing works explored two types of threats. First, a firmware modification, known also as a control logic corruption. Here, adversaries aim at maliciously infecting either the firmware or the memory of a target PLC. Secondly, we have a control logic manipulation. In such an attack, adversaries alter the input values that are fed to the control logic. Knowing this, the input manipulation could come from two main sources: the communication between the PLC and its engineering software or from the sensor readings that are scanned during execution of the control logic (see Figure 3). In the following, we present the existing studies of each threat.

4.3.1. Control Logic Corruption Attacks

In recent years, a number of attacks aiming at manipulating the firmware of PLCs have been published. Basnight [33] explored vulnerabilities in a well-known PLC (Allen-Bradley ControlLogix) to perform an intentional firmware modification attack. The author analyzed the firmware update validation to discover weaknesses that facilitate firmware counterfeiting. After that, he created a counterfeit firmware sample that was uploaded and executed on an ControlLogix L61 PLC. The firmware sample used to compromise the PLC was obtained directly from the vendor website as a firmware update package. In a follow-up work, Basnight [34] presented two methods of control logic corruption attacks on Allen-Bradley PLCs. The first method used immediate values in instructions to infer a reasonable image base. The second method used a hardware debugger to halt a PLC in order to obtain a memory dump. The image base could be found by manually analyzing common instruction patterns in the memory dump. This vulnerability allowed the authors to execute arbitrary code in a PLC by exploiting the firmware update feature. Both studies were tested on Allen-Bradley PLCs, targeting the security goal I, and exploited the vulnerability V4 that existing in ControlLogix PLCs.

Peck et al. [35] demonstrated how, by using commonly available tools, an adversary can learn how firmware is loaded into different field device Ethernet cards and write his own malicious firmware before loading that malicious firmware into the field device Ethernet cards. In their experiments, the authors found a lack of source and data authentication on firmware uploads in both Koyo and Rockwell Automation PLCs. As a proof of concept, they uploaded modified web pages that are available from a similar PLC module. This

study compromised the vulnerability V4 in the control logic of PLCs, and targeted the security goal I.

Schuetz [36] performed control logic corruption attacks on PLCs by using the Joint Test Action Group (JTAG) interface. The author first extracted the firmware image and performed static and dynamic analysis to identify execution paths and generate memory dumps. The firmware is repackaged with a malicious attack that triggers a DoS attack with a combination of control commands by writing a sentinel value to an unused flash memory area. Rais et al. [37] proposed a JTAG-based framework, namely Kyros, for a reliable PLC memory acquisition. Kyros systematically creates a JTAG profile of a PLC through hardware assessment, JTAG pins identification, memory map creation, and optimizing acquisition parameters. As a case study, the authors implemented Kyros on Allen-Bradley PLCs and revealed the tested PLC's memory dumps that are basically used in typical firmware modification attacks. Both studies targeted the security goal I, and exploited the vulnerability V4.

The work from Garcia [38] presented HARVEY, a rootkit that once installed in the device's firmware has the capability to inspect the control logic and then modify its instructions. The rootkit can evade operators viewing the human-machine interface (HMI) by faking sensor input to the control logic to generate adversarial commands that an operator would expect to see. HARVEY could enlarge the harm to the control process and result in extremely huge failures without operators being alarmed about the ongoing attack. However, the authors did their investigation on an assumption that attackers already have access to the PLC firmware, which was less observed than the user program. HARVEY was also aware of the control process that the PLC handles and could exploit the vulnerabilities V4 and V5 to intercept the measurement inputs that are used by this process, i.e., it targeted the security goals C and I.

4.3.2. Control Logic Manipulation Attacks

Many academic efforts discussed manipulating I/O data exchanged between engineering stations and PLCs over specific network packets. Among many attacks that Beresford introduced in [7], he showed that an adversary could reveal the mapping technique between the names and input variables that the control logic runs in the victim PLC. This would allow him to modify the program based on needs, causing disastrous damage to the physical processes. However, the chance of achieving a successful mapping of the variables through a memory probing is small.

Alsabbagh et al. [45] manipulated the I/O bytes that represents actual sensor readings exchanged between the connected devices. The authors introduced a new attack approach based on integrating an I/O database created prior to the attack. They first collected network captures that contain actual sensor and output values from the target device, and then employed a man-in-the-middle attack in order to interrupt and replace the correct I/O data with a false one from the I/O database. Their attack technique did not require adversaries to map the I/O data bytes to its readable version as [46] did. However, both studies [45,46] exploited the vulnerability V1, V4, and V6 targeting the security goal I.

Lim et al. [39] interrupted and altered the command-37 packets transmitted between a PLC (Schneider Tricon PLC) and its engineering software by performing a hijacking attack. The crafted packets contain inputs to industrial PLCs that are commonly used in nuclear power plant settings. The authors used reverse-engineering techniques by which the general structure of the Tricon communication protocol was identified. Afterward, they conducted a successful control logic injection causing common-mode failures for all modules which required a reset of the Tricon PLC. This work leverages the vulnerabilities V4 and V5, and targets the security goals I and A. However, the authors did not take into account that if a DPI security method had been enabled, it would have detected the attacker payload or the input data before reaching the device.

To evade the DPI detection method, Yoo et al. [41,42] introduced a stealthiness injection by splitting the malware transmission into small fragments and transferring one byte

per packet with a substantial padding of noises. This is due to the fact that the DPI mechanism relies on merging packets to detect any abnormal traffic. Thus, it was not able to disclose a very small size payload. The authors exploited the vulnerability V5 to control the victim PLC by injecting malicious code. The studies also discussed the possibility of conducting a stealthy program modification and input manipulation at the network level. The authors applied their attacks on two industry-scale PLCs, precisely Modicon M221 and Allen–Bradley MicroLogix 1400. The attack scenarios presented in these works managed successfully to evade well-known intrusion-detection methods, e.g., signature-based intrusion detection and payload-based anomaly detection [52].

A research group in [40] demonstrated a control logic injection attack that aimed at manipulating the data stored in the PLC's intermediate register. To this end, the authors read data values representing inputs, inverted these values and wrote them back to the victim. They repeated this operation in a loop until the PLC stops running. The authors of this work exploited the vulnerabilities V4 and V5, targeting the security goals C and I. However, their attack was limited to Siemens PLCs, and did not take any security means implemented in mind.

The authors of [43] introduced ladder logic bombs (LLB). It is a combination of a typical control logic injection attack and a control logic manipulation attack. The adversary's malware was sneakily inserted in an already running LD program as a regular subroutine that is only activated by a certain condition. Once the condition is met, the malware impersonates a legitimate sensor reading with false values. LLB was customized to bypass any inspection method applied by giving legitimate instructions/names similar to the previously used names in the same industrial setup. It is worth mentioning that this study targeted the security goal I and managed successfully to exploit the vulnerabilities V4 and V6, causing unsafe behavior of the attacked PLCs.

McLaughlin et al. [16] conducted a controller-aware false data injection (CaFDI) attack. It allows adversaries to get only partial information about the target subsystem and produces predictable malicious results. CaFDI proceeds in two main steps. First, a model representing the PLC control logic is constructed based on the I/O traces. Secondly, it seeks for a set of inputs that lead the model to calculate the desired malicious behavior. In other words, CaFDI seeks for an I/O path that could be used as the sensor readings for the control logic. CaFDI was evaluated on a set of representative control systems and showed that exploiting insecure sensor configurations is feasible.

In addition, Xiao [44] showed that an attacker could collect fault-free I/O traces, and formalizes a representative model by using a non-deterministic autonomous automation with output (NDAAO). He first built a word set of NDAAO sequences and sought unobserved false sequences from the word set to infect the exposed sensors. His attack exploited the vulnerability V4 resulting in operating the physical process controlled by the infected PLC improperly.

In a similar setting, Abbasi et al. [47] combined the control flow of the program to exploit certain pin control operations, leveraging the absence of hardware interrupts associated to the PLC's pins. Their attack allows an attacker to reliably take control of the control process, while remaining stealthy to both the PLC runtime and the ICS supervisor who observes the process through an HMI. This attack did not demand modifications of the PLC logic, traditional kernel tampering, or hooking techniques that are normally observed by anti-rootkit tools. The authors introduced two variations of their attack implementations. The first one allows an extremely reliable manipulation of the process at the cost of requiring root access, and the second implementation allows manipulation to be achieved without root access.

5. PLC Vendors Responses to Control Logic Injection Attacks

In the former section, we gave an overview of the current studies discussing control logic injection attacks. Our investigations showed that several PLC vendors, e.g., Siemens, Schneider Electronic, Rockwell Automation, Koyo, Mitsubishi, etc. were all impacted by

control logic injection attacks. The security researchers report to vendors the vulnerabilities existing in their devices. On the other hand, vendors assess the reports and respond by releasing new security patches, if necessary, and recommending the consumers to follow specific protection measures in order to reduce the risk of potential threats. This collaboration between PLC vendors and researchers helps in advancing the industrial products' security, and supporting development of industry best practices. More importantly, it assists millions of consumers to manage security risks and make ICSs more resilient against attacks. In this section, we highlight the mostly common mitigation solutions that the top three PLC vendors (Siemens, Schneider Electric, and Rockwell Automation) recommend their consumers implement in response to control logic injection attacks.

5.1. Siemens

The leading automation vendor dedicated a team of seasoned security experts, named ProductCERT (<https://new.siemens.com/global/en/products/services/cert.html#SecurityPublications>, accessed on 24 October 2022), to assess and investigate the public reporting of security issues related to Siemens products. This team acts as the central contact point for researchers to report potential vulnerabilities. The ProductCERT team suggested several security measures and mitigation solutions that users can implement to reduce the risk of control logic injection attacks as follows [53–56]:

- Users are recommended to apply protection-level 3 (read/write protection), i.e., protecting the control logic with a password. Therefore, whenever an ICS supervisor attempts to access the control logic running in a PLC, the device first requires an authentication to allow him to read/write the code.
- Siemens strongly recommends one protect network access to devices with appropriate mechanisms, e.g., cell protection concept, network segmentation, etc.
- Consumers are encouraged to use virtual private networks (VPNs) for protecting network communication between cells, operate the device only within trusted networks, and apply defense-in-depth. These measures aid the prevention of external attackers from accessing the PLC and its running program via web vulnerabilities.
- Users of Siemens S7 PLCs should configure an intrusion detection system (IDS) to monitor traffic for unusual or unauthorized activity.
- Using firewalls to manage communication to and within the automation network. All control system networks and remote devices should be located behind firewalls and be isolated from the business network.
- Allowing only known and verified MAC addresses to communicate with appropriate resources on the automation network. For instance, users should not permit a policy allowing any engineering workstation to communicate with all PLCs on the automation network.

5.2. Schneider Electric

Schneider Electric recommends users apply the following mitigations to help reduce the risk of control logic injection attacks against its controllers [57,58]:

- Users are highly recommended to set up network segmentation and implement a firewall to block all unauthorized access to Port 502/TCP.
- Users should disable all unused protocols, especially programming protocol. This action will prevent unintended remote programming access.
- Schneider recommends its consumers to set a password to protect the project. Users are also advised to not use the same password for both read and write access on the controller.
- Users should locate control and safety system networks and remote devices behind firewalls and isolate them from the business network.
- Users are recommended to install physical controls, so no unauthorized personnel can access industrial control and safety systems, components, and networks.
- All PLCs should be placed in locked cabinets and never left in the "Program" mode.

- It is encouraged that one minimize network exposure for all industrial devices and systems and ensure that they are not accessible from the Internet.
- When remote access is required, ICS operator should use secure methods such as VPNs. However, VPNs may also have vulnerabilities and should be updated to the most current version available.

5.3. Rockwell Automation

Rockwell Automation listed many mitigation steps that its consumers are encouraged to follow to prevent control logic injection attacks as follows [59,60]:

- Users are advised to monitor PLC change log for any unexpected modifications or anomalous activity.
- Users should utilize the controller log feature, as well as use change detection in the Logix designer application.
- If it is possible, users should utilize the functionality in FactoryTalkAssetCenter software to detect control logic changes.
- Users are strongly advised to implement CIP Security (https://literature.rockwellautomation.com/idc/groups/literature/documents/at/secure-at001_-en-p.pdf (accessed on 25 October 2022)) to assist in preventing unauthorized connections when properly deployed.
- It is recommended that one minimize network exposure for all control devices and/or systems, and ensure they are not accessible from the Internet.
- Users should locate control system networks and remote devices behind firewalls and isolate them from the business network.

The mitigation solutions recommended by vendors lowered the chances for adversaries to access and compromise PLCs, but unfortunately could not completely prevent control logic injection attacks. For instance, protecting the user program with a password is not fully successful. Academic efforts, e.g., [61] proved that attackers can bypass the authentication and access the control logic in different password-protected PLCs for different vendors. Furthermore, implementing only firewalls without any additional security measures is not sufficient to secure ICSs. If any network server behind the firewall is authorized to access the target PLC through the firewall, there is a vulnerability. In the next section, we illustrate in detail the challenges that both PLC vendors and researchers face in terms of securing PLCs against such severe threats.

6. Security Challenges and Future Direction Recommendations

In this section, we highlight the security challenges related to control logic injection threats, and offer recommendations that might help in detecting those attacks and mitigating their impacts. Our recommendations suggest future research directions based on a thorough analysis of the current security challenges.

6.1. Security Challenges

Our investigations showed that the research community is still far away from providing a complete and generalized defense solution against control logic injection attacks. However, in the following we summarize the current security challenges that the existing research encounters.

6.1.1. Expanded Attack Surfaces

The attack surfaces for control logic have been expanded. The aforementioned studies have shown several potential weaknesses that an attacker can compromise and then inject his malicious payload. For instance, the communication between the engineering station and the PLC could be intercepted and hijacked. PLCs provided with a web server functionality, i.e., Internet-faced PLCs are insecure and vulnerable in such a way that an attacker can turn them as gateways to inject other PLCs in the same subnet [24]. Furthermore, attackers can compromise sensors and firmware to cause unsafe behaviors. For all that, it becomes

challenging for defense solutions to generate a convenient attacker model because any industrial device running in different automation layers could be exploited.

6.1.2. Predefined Hierarchical Memory Layout

Multiple research studies have discussed this vulnerability and conducted successful control logic corruption attacks based on compromising the PLC's memory. However, recent defense solutions like [62,63] have faced some challenges. For instance, the address space layout randomization (ASLR) solution consumes more scan cycles. As a consequence, the maximum allowed scan cycle time for PLCs is pretty likely not met. Another challenge is that regular control flow integrity based solutions request a considerable space in the memory which makes detecting the infection in real time is quite difficult. Furthermore, the PLC memory structure is vendor-specific. Therefore, most of the attacks aiming at corrupting them are product-driven. For all that, designing a lightweight and generalized defense solution is a difficult task.

6.1.3. Confidentiality and Integrity of the Program I/O

Our analyses of the current works showed that plenty of the studies relied on exploiting I/O traces to perform control logic manipulation attacks. This is mostly done by extracting information of the physical process driven by target PLC(s) and manipulating I/O data to cause unsafe behaviors. Securing the control logic I/O is challenging with respect to the input surfaces of the control logic that have already expanded as sensors and physical processes that day by day become public infrastructure. Furthermore, the I/O needs to be updated recursively to meet the maximum scan cycle time allowed.

6.1.4. Stealthy Attacks

Many stealthiness attack strategies have been discussed and presented in different attack scenarios, e.g., concealing a malicious code as a bad coding practice, payload evasiveness with fragmentation and noise padding to avoid being detected by DPI security solutions, crafting input data values to evade verification mechanisms, hiding the attacker injection in a certain memory block or in a configuration block (e.g., the less used one), and misleading the engineering software with a faked legitimate behavior. For all that, we can conclude that detecting stealthy attacks by using several techniques is a huge challenge for the security community.

6.1.5. Different Schemes for Compiling Codes

PLC vendors use different mechanisms and schemes to address inputs and outputs of PLCs. They might also choose different methods by which to organize memory layouts, programming blocks, and instruction sets. Therefore, it is difficult for a control logic to be compatible across many PLCs from different vendors or even models from the same manufacturer. As a consequence, it is quite challenging for any security solution to be applied on different PLC models provided from different vendors or not sharing the same model version.

6.2. Future Research Recommendations

We discussed the security challenges that the security research community is facing to prevent/mitigate control logic injection attacks. In this section, we recommend more research efforts and suggest future paths that might help in detecting and mitigating such severe attacks.

6.2.1. Source Code Injection

In Section 4.1, we discussed malicious attacks at the source code level, which could disguise themselves as bad coding practices, and are difficult to notice. Our investigations showed that attackers could perform stealthy attacks by setting certain user-defined operands at desired values or using hidden jumps with a stealthy logger to leak critical

program information. Therefore, we recommend further research in terms of comparing the original program behavior with unsafe behaviors and performing automatic program cleaning in case unsafe behavior is detected. This solution might detect stealthy injections at the very beginning and avoid disastrous consequences that such attacks might cause.

6.2.2. Bytecode/Binary Injection

This attack strategy is widely used to maliciously alter the PLC's program and disrupt the physical process. In fact, only a few studies [48,49,64,65] suggested detection-solution methods at the bytecode level. This is due to the fact that several PLC features are not supported in the normal instruction sets, and a reverse-engineering is keenly required in any possible detection solution. The research community still has a lack of open source libraries, similar to [66], aiming at disassembling PLC's programs from different vendors into high-level source code programs and vice versa. Such libraries would help the security researchers in understanding better how attackers can manipulate the PLC programs in low-level formats, and thus formalize a detection plant model that checks the compiled code sent from the engineering software with the one received on the PLC side. We also recommend further studies to reverse-engineer the function blocks, parameters list, and timers and counters that each type of PLC supports. We believe that such open-source libraries and research studies can provide a more resistant PLC-based environment against malicious attacks at the bytecode level.

6.2.3. Control Logic Manipulation Attacks

Our investigations showed that manipulation attacks are widely adopted by the attackers. This means that they are still capable of conducting control logic attacks by manipulating sensor readings. Thus, we recommend more future research in the direction of checking the program behaviors consistently. A possible solution might be formalizing a plant model to detect manipulation attacks by considering instruments on the input and output variables of the programs and comparing the values with these from the plant model. We believe such a solution would mitigate these attacks and monitor the program behaviors accurately.

6.2.4. Control Logic Corruption Attacks

PLCs are embedded devices, meaning that they have only limited memory to store the control logic blocks and any additional security solution. Furthermore, PLCs are increasingly employed in supervisory control and data acquisition (SCADA) and different implementations which transfer the data over possibly insecure networks. For all that, an appropriate security solution could be introducing a lightweight run-time formal verification that shares the memory with the PLC. The proposed verification should have authorized access to the inputs and outputs of the PLC and reveals any abnormal change. Moreover, the control logic blocks should also be concurrently integrated with the verification model and scanned each time the user attempts to upload or download a new control logic.

7. Conclusions

In this paper, a new systematization on the recent studies discussing control logic injection attacks was presented. Our methodology categorized the existing research based on three attack scenarios. For each study involved in this work, we overviewed the attack techniques, security goals, underlying vulnerabilities, and attack tools. Our systematization showed that the attack surfaces have been expanded over the years, and attackers have been capable of exploiting the control logic running in PLCs in different ways. Sophisticated control logic attacks could be extremely severe and compromise the entire chain of the automation pyramid. Meanwhile, skilled adversaries designed their attacks to be stealthy and even evade advanced security detection methods. We also identified the current security challenges in defending against such serious threats. To overcome these challenges,

we encourage future direction paths that investigate in some security aspects that have a lack of studies and research.

Author Contributions: Conceptualization, W.A. and P.L.; systematization methodology, W.A. and P.L.; writing—original draft preparation, W.A.; writing—review and editing, P.L.; visualization, W.A.; supervision, P.L. The paper was proofread and checked by both authors. All authors have read and agreed to the published version of the manuscript.

Funding: The publication of this article was funded by the Open Access Fund of the Leibniz Association.

Data Availability Statement: Not applicable, the study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

A	Availability
ACM	Association for Computing Machinery
ASLR	Address Space Layout Randomization
C	Confidentiality
CaFDI	Controller aware False Data Injection
CI	Critical Infrastructure
CIA	Confidentiality Integrity Availability
CLIK	Control Logic Injection Attack
CPU	Central Processing Unit
CVE	Common Vulnerabilities and Exposures
DBLP	Digital Bibliographic Library Publication
DoS	Denial of Service
DPI	Deep Packet Inspection
EEPROM	Electrically Erasable Programmable Read-only Memory
FBD	Function Block Diagram
FWD	Forward
HMI	Human Machine Interface
I	Integrity
ICS	Industrial Control System
ICS-CERT	Industrial Control Systems Cyber Emergency Response Team
ICSREF	Industrial Control System Reverse Engineering Framework
IDE	Integrated Development Environment
IDS	Intrusion Detection System
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IL	Instruction List
I/O	Input/Output
ISO-TSAP	International Organization for Standardization-Transport Service Access Point
IT	Information Technology
JMP	Jump
JTAG	Joint Test Action Group
LD	Ladder Diagram
LLB	Ladder Logic Bombs
MAC	Media Access Control
MC	Machine Code
NDAAO	Non-Deterministic Autonomous Automation with Output
NVD	National Vulnerability Database
OB	Organization Block
OS	Operating System
PCCC	Programmable Controller Communication Commands
PLC	Programmable Logic Controller
RAM	Random Access Memory

RTN	Return
S	Sensor
SABOT	Specification-based Attacks against Boolean Operations and Timers
SCADA	Supervisory Control and Data Acquisition
SFC	Sequential Function Chart
SNMP	Simple Network Management Protocol
ST	Structured Text
TIA	Totally Integrated Automation
ToD	Time of Day
V	Vulnerability
VPN	Virtual Private Network

References

- Alsabbagh, W.; Langendörfer, P. A Stealth Program Injection Attack against S7-300 PLCs. In Proceedings of the 22nd IEEE International Conference on Industrial Technology (ICIT), Valencia, Spain, 10–12 March 2021; pp. 986–993. Available online: <https://ieeexplore.ieee.org/document/9453483> (accessed on 4 November 2022).
- Falliere, N.; Murchu, L.O.; Chien, E. W32. Stuxnet Dossier, White Paper Symantec Corp. Security Response. 2011; Volume 5, p. 29. Available online: http://popsci.com.au/files/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf (accessed on 4 November 2022)
- Pinto, A. D.; Dragoni, Y.; Carcano, A. Triton: The first ics cyber attack on safety instrument systems. In Proceedings of the Black Hat USA, Las Vegas, NV, USA, 4–9 August 2018; pp. 1–26. Available online: https://www.nozominetworks.com//downloads/US/Nozomi-Networks-TRITON-The-First-SIS-Cyberattack.pdf?__hstc=46213176.fb847b8c24524308051f92c191e5a1c8.1657985478826.1658174564396.1658186120920.11&__hssc=46213176.3.1658186120920&__hsfp=1542169609 (accessed on 4 November 2022).
- Assante, M.J. Confirmation of a Coordinated Attack on the Ukrainian Power Grid. SANS Industrial Control Systems Security Blog: 2016; p. 207. Available online: <https://www.sans.org/blog/confirmation-of-a-coordinated-attack-on-the-ukrainian-power-grid/> (accessed on 4 November 2022).
- Sun, R.; Mera, A.; Lu, L.; Choffnes, D. SoK: Attacks on Industrial Control Logic and Formal Verification-Based Defenses. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS&P), Vienna, Austria, 6–10 September 2021; pp. 385–402. [CrossRef]
- Tiegelkamp, M.; John, K. *IEC 61131-3: Programming Industrial Automation Systems*; Springer: Berlin/Heidelberg, Germany, 2001; Volume VI, p. 376.
- Beresford, D. Exploiting Siemens Simatic S7 PLCs. In Proceedings of the Black Hat USA, Las Vegas, NV, USA, 3–4 August 2011; pp. 723–733.
- ICS-CERT. CVE-2017-13997. Available online: <https://nvd.nist.gov/vuln/detail/CVE-2017-13997> (accessed on 18 June 2022).
- ICS-CERT. CVE-2018-10619. Available online: <https://nvd.nist.gov/vuln/detail/CVE-2018-10619> (accessed on 21 June 2022).
- ICS-CERT. CVE-2017-12739. Available online: <https://nvd.nist.gov/vuln/detail/CVE-2017-12739> (accessed on 22 June 2022).
- ICS-CERT. CVE-2017-12088. Available online: <https://nvd.nist.gov/vuln/detail/CVE-2017-12088> (accessed on 23 June 2022).
- ICS-CERT. CVE-2019-10922. Available online: <https://nvd.nist.gov/vuln/detail/CVE-2019-10922> (accessed on 22 June 2022).
- Perrin, C. *The CIA Triad*; 2008. Available online: <http://www.techrepublic.com/blog/security/the-cia-triad> (accessed on 3 July 2022).
- Serhane, A.; Raad, M.; Raad, R.; Susilo, W. PLC code-level vulnerabilities. In Proceedings of the International Conference on Computer and Applications (ICCA), Beirut, Lebanon, 25–26 August 2018; pp. 348–352. Available online: <https://ieeexplore.ieee.org/document/8460287/> (accessed on 7 March 2022).
- Valentine, S.E. Plc Code Vulnerabilities through Scada Systems. Ph.D. Thesis, University of South Carolina, Computer Science Dept., Columbia, SC, USA, 1 January 2013. Available online: <https://scholarcommons.sc.edu/cgi/viewcontent.cgi?article=1804&context=etd> (accessed on 3 April 2022).
- McLaughlin, S.; Zonouz, S. Controller-aware false data injection against programmable logic controllers. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 848–853. [CrossRef]
- McLaughlin, S. On Dynamic malware payloads aimed at programmable logic controllers. In Proceedings of the 6th USENIX Conference on Hot Topics in Security, San Francisco, CA, USA, 9 August 2011. Available online: https://www.usenix.org/legacy/events/hotsec11/tech/final_files/McLaughlin.pdf (accessed on 7 April 2022).
- McLaughlin, S.; McDaniel, P. SABOT: Specification-based payload generation for programmable logic controllers. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; pp. 439–449.
- Senthivel, S.; Dhungana, S.; Yoo, H.; Ahmed, I.; Roussev, V. Denial of Engineering Operations Attacks in industrial Control Systems. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, Tempe, AZ, USA, 19–21 March 2018; pp. 319–329. [CrossRef]

20. Keliris, A.; Maniatakos, M. ICSREF: A framework for automated reverse engineering of industrial control systems binaries. In Proceedings of the 26th Annual Network and Distributed System Security Symposium, San Diego, CA, USA, 24–27 February 2019. Available online: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_07A-5_Keliris_paper.pdf (accessed on 3 May 2022).
21. Kalle, S.; Ameen, N.; Yoo, H.; Ahmed, I. CLIK on PLCs! Attacking Control Logic with Decompilation and Virtual PLC. In Proceedings of the Workshop on Binary Analysis Research (BAR), San Diego, CA, USA, 24 February 2019. Available online: https://www.ndss-symposium.org/wp-content/uploads/bar2019_74_Kalle_paper.pdf (accessed on 8 May 2022).
22. Qasim, S.A.; Smith, J.M.; Ahmed, I. Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems. *Forensic Sci. Int. Digit. Investig.* **2020**, *33*, 301013. [CrossRef]
23. Alsabbagh, W.; Langendörfer, P. A Control Injection Attack against S7 PLCs -Manipulating the Decompiled Code. In Proceedings of the IECON 2021—47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, 13–16 October 2021; pp. 1–8. [CrossRef]
24. Klick, J.; Lau, S.; Marzin, D.; Malchow, J.O.; Roth, V. Internet-facing PLCs as a network backdoor. In Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 28–30 September 2015; pp. 524–532. Available online: <https://ieeexplore.ieee.org/document/7346865> (accessed on 17 May 2022).
25. Spenneberg, R.; Bruggemann, M.; Schwartke, H. Plc-Blaster: A Worm Living Solely in the Plc. Black Hat Asia, Marina Bay Sands: Singapore. 2016. Available online: <https://www.blackhat.com/docs/asia-16/materials/asia-16-Spenneberg-PLC-Blaster-A-Worm-Living-Solely-In-The-PLC-wp.pdf> (accessed on 24 May 2022).
26. Lei, C.; Donghong, L.; Liang, M. The spear to break the security wall of S7CommPlus. In Proceedings of the Black Hat Europe, London, UK, 4–7 December 2017. Available online: <https://www.blackhat.com/docs/eu-17/materials/eu-17-Lei-The-Spear-To-Break%20-The-Security-Wall-Of-S7CommPlus-wp.pdf> (accessed on 17 May 2022).
27. Biham, E.; Bitan, S.; Carmel, A.; Dankner, A.; Malin, U.; Wool, A. Rogue7: Rogue Engineering-Station attacks on S7 Simatic PLCs. In Proceedings of the Black Hat USA, Las Vegas, NV, USA, 3–8 August 2019. Available online: <https://i.blackhat.com/USA-19/Thursday/us-19-Bitan-Rogue7-Rogue-Engineering-Station-Attacks-On-S7-Simatic-PLCs-wp.pdf> (accessed on 3 June 2022).
28. Hui, H.; McLaughlin, K. Investigating Current PLC Security Issues Regarding Siemens S7 Communications and TIA Portal. In Proceedings of the 5th International Symposium for ICS & SCADA Cyber Security Research, Hamburg, Germany, 29–30 August 2018; pp. 67–73. [CrossRef]
29. Hui, H.; McLaughlin, K.; Sezer, S. Vulnerability analysis of S7 PLCs: Manipulating the security mechanism. *Int. J. Crit. Infrastruct. Prot.* **2021**, *35*, 100470. [CrossRef]
30. Alsabbagh, W.; Langendörfer, P. Patch Now and Attack Later—Exploiting S7 PLCs by Time-Of-Day Block. In Proceedings of the 2021 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), Victoria, BC, Canada, 10–12 May 2021; pp. 144–151. [CrossRef]
31. Alsabbagh, W.; Langendorfer, P. A New Injection Threat on S7-1500 PLCs—Disrupting the Physical Process Offline. *IEEE Open J. Ind. Electron. Soc.* **2022**, *3*, 146–162. Available online: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9713953> (accessed on 21 July 2022). [CrossRef]
32. Alsabbagh, W.; Langendörfer, P. No Need to be Online to Attack—Exploiting S7-1500 PLCs by Time-Of-Day Block. In Proceedings of the 2022 XXVIII International Conference on Information, Communication and Automation Technologies (ICAT), Sarajevo, Bosnia and Herzegovina, 16–18 June 2022; pp. 1–8. [CrossRef]
33. Basnight, Z. Firmware Counterfeiting and Modification Attacks on Programmable Logic Controllers. Master’s Thesis, Graduate School of Engineering and Management Air Force Institute of Technology Air University, Wright Patterson Air Force Base, OH, USA, 2013.
34. Basnight, Z.; Butts, J.; Lopez, J., Jr.; Dube, T. Firmware modification attacks on programmable logic controllers. *Int. J. Crit. Infrastruct. Prot.* **2013**, *6*, 76–84. [CrossRef]
35. Peck, D.; Peterson, D. Leveraging ethernet card vulnerabilities in field devices. In Proceedings of the SCADA Security Scientific Symposium; 2009; pp. 1–19. Available online: <http://www.icsdefender.ir/files/scadadefender-ir/paygahdanesh/ghyreboom/asibpaziriha/DigitalBond%20-%20Leverage%20Ethernet%20Vulnerabilities%20in%20Field%20Devices.pdf> (accessed on 4 August 2022).
36. Schuett, C.; Butts, J.; Dunlap, S. An evaluation of modification attacks on programmable logic controllers. *Int. J. Crit. Infrastruct. Prot.* **2014**, *7*, 61–68. [CrossRef]
37. Rais, M.H.; Awad, R.A.; Lopez, J.; Ahmed, I. JTAG-based PLC memory acquisition framework for industrial control systems. *Forensic Sci. Int. Digit. Investig.* **2021**, *37*, 301196. [CrossRef]
38. Garcia, L.A.; Brasser, F.; Cintuglu, M.H.; Sadeghi, A.R.; Mohammed, O.; Zonouz, S.A. Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit. In Proceedings of the 2017 Network and Distributed System Security Symposium, San Diego, CA, USA, 26 February–1 March 2017. Available online: <https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/hey-my-malware-knows-physics-attacking-plcs-physical-model-aware-rootkit/> (accessed on 11 July 2022).
39. Lim, B.; Chen, D.; An, Y.; Kalbarczyk, Z.; Iyer, R. Attack Induced Common-Mode Failures on PLC-Based Safety System in a Nuclear Power Plant: Practical Experience Report. In Proceedings of the 2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC), Christchurch, New Zealand, 22–25 January 2017; pp. 205–210. [CrossRef]

40. Wang, Y.; Liu, J.; Yang, C.; Zhou, L.; Shuangfei, L.; Zhaoyan, X. Access Control Attacks on PLC Vulnerabilities. *J. Comput. Commun.* **2018**, *6*, 311–325. [CrossRef]
41. Yoo, H.; Ahmed, I. Control Logic Injection Attacks on Industrial Control Systems. In Proceedings of the IFIP International Conference on ICT Systems Security and Privacy Protection, Lisbon, Portugal, 25–27 June 2019. Available online: https://link.springer.com/content/pdf/10.1007/978-3-030-22312-0_3.pdf (accessed on 17 August 2022).
42. Yoo, H.; Kalle, S.; Smith, J.M.; Ahmed, I. Overshadow plc to detect remote control-logic injection attacks. In Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA) 2019, Gothenburg, Sweden, 19–20 June 2019; Perdisci, R., Maurice, C., Giacinto, G., Almgren, M., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2019; Volume 11543. Available online: http://www.people.vcu.edu/~iahmed3/publications/dimva_2019_shade.pdf (accessed on 26 July 2022).
43. Govil, N.; Agrawal, A.; Tippenhauer, N.O. On ladder logic bombs in industrial control systems. In *Computer Security, Proceedings of the SECPRE, Oslo, Norway, 14–15 September 2017*; Springer: Cham, Switzerland, 2017; Volume 10683. [CrossRef]
44. Xiao, M.; Wu, J.; Long, C.; Li, S. Construction of false sequence attack against PLC based power control system. In Proceedings of the 2016 35th Chinese Control Conference (CCC), Chengdu, China, 27–29 July 2016; pp. 10090–10095. [CrossRef]
45. Alsabbagh, W.; Langendörfer, P. A Fully-Blind False Data Injection on PROFINET I/O Systems. In Proceedings of the 2021 IEEE 30th International Symposium on Industrial Electronics (ISIE), Kyoto, Japan, 20–23 June 2021; pp. 1–8. [CrossRef]
46. Noorzadeh, M.; Shakerpour, M.; Meskin, N.; Ünal, D.; Kho-rasani, K. A Cyber-Security Methodology for a Cyber-Physical Industrial Control System Testbed. *IEEE Access* **2021**, *9*, 16239–16253. [CrossRef]
47. Abbasi, A.; Hashemi, M. Ghost in the PLC designing an undetectable programmable logic controller rootkit via pin control attack. In Proceedings of the Black Hat Europe, London, UK, 3–4 November 2016; pp. 1–35. Available online: <https://www.blackhat.com/docs/eu-16/materials/eu-16-Abbasi-Ghost-In-The-PLC-Designing-An-Undetectable-Programmable-Logic-Controller-Rootkit-wp.pdf> (accessed on 3 July 2022).
48. McLaughlin, S.; Zonouz, S.; Pohly, D.; McDaniel, P. A Trusted Safety Verifier for Process Controller Code. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 23–26 February 2014. Available online: <https://web.eecs.umich.edu/~mahlke/courses/583f18/lectures/Nov26/paper2.pdf> (accessed on 7 July 2022)
49. Zonouz, S.; Rrushi, J.; McLaughlin, S. Detecting Industrial Control Malware Using Automated PLC Code Analytics. *IEEE Secur. Priv.* **2014**, *12*, 40–47. [CrossRef]
50. Ferrari, A.; Magnani, G.; Grasso, D.; Fantechi, A. Model checking interlocking control tables. In *FORMS/FORMAT 2010*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 107–115.
51. RX Family User’s Manual: Software, Renesas Electronics. 2013. Available online: <https://www.renesas.com/us/en/document/mas/rx-family-users-manual-software-rev120?language=en> (accessed on 16 July 2022).
52. Wang, K.; Parekh, J.J.; Stolfo, S.J. Anagram: A content anomaly detector resistant to mimicry attack. In Proceedings of the International Conference on Recent Advances in Intrusion Detection (RAID), Hamburg, Germany, 20–22 September 2006; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4219, pp. 226–248. [CrossRef]
53. SSA-232418: Vulnerabilities in SIMATIC S7-1200 and SIMATIC S7-1500 CPU Families. Available online: <https://cert-portal.siemens.com/productcert/pdf/ssa-232418.pdf> (accessed on 22 October 2022).
54. SSA-603476: Web Vulnerabilities in SIMATIC CP 343-1/CP 443-1 Modules and SIMATIC S7-300/S7-400 CPUs. Available online: <https://cert-portal.siemens.com/productcert/pdf/ssa-603476.pdf> (accessed on 22 October 2022).
55. SSA-818183: Denial-of-Service Vulnerability in SIMATIC S7-300 CPU Family. Available online: <https://cert-portal.siemens.com/productcert/pdf/ssa-818183.pdf> (accessed on 22 October 2022).
56. ICS Advisory (ICSA-11-223-01A). Available online: <https://www.cisa.gov/uscert/ics/advisories/ICSA-11-223-01A> (accessed on 22 October 2022).
57. ICS Advisory (ICSA-20-343-04). Available online: <https://www.cisa.gov/uscert/ics/advisories/icsa-20-343-04> (accessed on 22 October 2022).
58. ICS Advisory (ICSA-21-194-02). Available online: <https://www.cisa.gov/uscert/ics/advisories/icsa-21-194-02> (accessed on 22 October 2022).
59. ICS Advisory (ICSA-22-090-05). Available online: <https://www.cisa.gov/uscert/ics/advisories/icsa-22-090-05> (accessed on 22 October 2022).
60. ICS Advisory (ICSA-21-056-03). Available online: <https://www.cisa.gov/uscert/ics/advisories/icsa-21-056-03> (accessed on 22 October 2022).
61. Ayub, A.; Yoo, H.; Ahmed, I. Empirical Study of PLC Authentication Protocols in Industrial Control Systems. In Proceedings of the 15th IEEE Workshop on Offensive Technologies (WOOT’21), Co-Located with the 42nd IEEE Symposium on Security and Privacy and in Cooperation with UsenixAt, San Francisco, CA, USA, 27 May 2021. [CrossRef]
62. Chekole, E.G.; Chattopadhyay, S.; Ochoa, M.; Guo, H.; Cheramangalath, S. CIMA: Compiler-Enforced Resilience Against Memory Safety Attacks in Cyber-Physical Systems. *Comput. Secur.* **2020**, *94*, 101832. [CrossRef]
63. Chekole, E.G.; Ochoa, M.; Cheramangalath, S. SCOPE: Secure Compiling of PLCs in Cyber-Physical Systems. *Int. J. Crit. Infrastruct. Prot.* **2021**, *33*, 100431. [CrossRef]
64. Chang, T.; Wei, Q.; Liu, W.; Geng, Y. *Detecting plc Program Malicious Behaviors Based on State Verification*; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2018; Volume 11067, pp. 241–255.

-
65. Xie, Y.; Chang, R.; Jiang, L. A malware detection method using satisfactorily modulo theory model checking for the programmable logic controller system. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e5724. [[CrossRef](#)]
 66. DotNet Siemens PLC ToolBox Library. Available online: <https://github.com/dotnetprojects/DotNetSiemensPLCToolBoxLibrary> (accessed on 22 October 2022).