symmetry

MDPI

Article

Aye: A Trusted Forensic Method for Firmware Tampering Attacks

Yipeng Zhang, Ye Li and Zhoujun Li *

Department of Computer Science and Engineering, Beihang University, Beijing 100191, China
* Correspondence: zjli@buaa.edu.cn

Abstract: The Programmable Logic Controller (PLC) is located at the junction of the virtual network and physical reality in the Industrial Control System (ICS), which is vulnerable to attacks due to its weak security. Specifically, firmware tampering attacks take the firmware under the PLC operating system as the primary attack target. The firmware provides the bridge between PLC's hardware and software, which means tampering against the firmware can be more destructive and harmful than other attacks. However, existing defense and forensics methods against firmware tampering attacks are asymmetrical, which directly leads to the proliferation of such attacks and the difficulty of forensic tracing. How to accurately, quickly, and efficiently conduct forensics for such attacks is an urgent problem. In this paper, we designed and implemented a reliable detection method based on Joint Test Action Group (JTAG) and memory comparison—Aye, which can detect mainstream firmware tampering attacks reliably. To determine the effectiveness and reliability of Aye, we selected a widely used PLC to observe Aye's performance in defense and forensics by simulating the two latest PLC firmware tampering attack methods. The experimental results show that Aye can effectively defend against firmware tampering attacks, helping improve the efficiency and accuracy of such attack detection and forensics.

Keywords: industrial control system security; programmable logic controller; firmware tampering attack; digital forensics; joint test action group

Citation: Zhang, Y.; Li, Y.; Li, Z. Aye: A Trusted Forensic Method for Firmware Tampering Attacks. *Symmetry* **2023**, *15*, 145. https://doi.org/10.3390/sym15010145

Academic Editor: Debiao He

Received: 27 November 2022
Revised: 13 December 2022
Accepted: 22 December 2022
Published: 3 January 2023

1. Introduction

The Programmable Logic Controller (PLC) is widely used in Industrial Control Systems (ICSs), such as oil facilities, water supply, steel mills, and nuclear power plants, connecting the ICS's network and physical space [1]. Initially, PLC was designed more concerned with usability than security. Almost all PLCs lack encryption, authorization, and authentication mechanisms, leading to vulnerable and weak security [2]. More and more security events, e.g., the Stuxnet, Duqu, and Black Energy [3] have shown attacks against the PLC may cause substantive damages with economic and even life losses in the real world [4]. Specifically, a tampering attack against the PLC firmware is incredibly harmful and has become one of the most threatening attacks [5].

The PLC firmware can be considered as PLC's operating system, which can interpret code into binary signals that influence input and output signals, registers, and even the communication of network signals [6]. From a certain point of view, the PLC firmware has complete control over a PLC's software and hardware. In the firmware tampering attack, attackers can download a re-signed firmware with malicious code into PLC to set a backdoor, as shown in Figure 1, allowing device Denial of Service (DoS), quiet collection of data, and even causing catastrophic failures with substantive impact. However, existing forensics methods for firmware tampering attacks are mainly based on PLC memory contents, which are always acquired with PLC debugging tools or ICS protocols, such as GE-SRTP protocol [7], Modicon M221 protocol [8], and PCCC protocol of Allen-Bradley [9].

These approaches are less effective since they cannot acquire the entire PLC memory and are limited to the memory contents within a PLC's protocol address space [10].

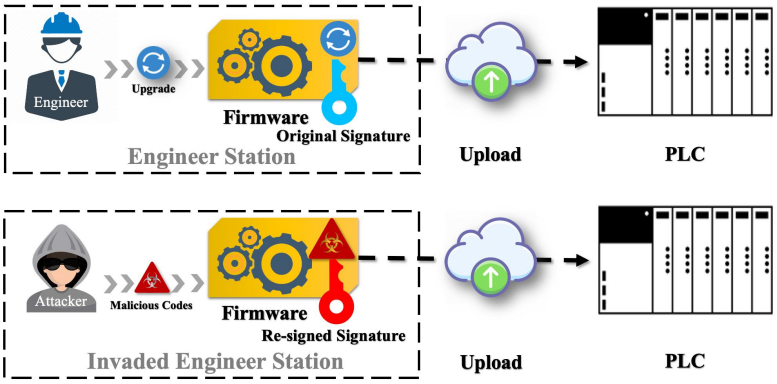


Figure 1. PLC firmware attacks in regular and invaded engineer stations.

Due to a PLC always having limited computational power and finite storage space and memory, it is also impossible to deploy complicated security defense measures [11]. Other advanced forensics methods, such as watchdog timer [12,13], always require manufacturers' support for additional software and hardware modifications of PLCs, which is not applicable to old equipment. How to credibly use forensics PLC and acquire memory without causing crashes or suspensions is a problem that needs urgent solving. The Joint Test Action Group (JTAG) may be a better choice to overcome such asymmetry. JTAG is an industry standard for on-chip instrumentation in Electronic Design Automation (EDA) as a complementary tool to digital simulation [14]. Almost all PLC manufacturers use JTAG at the developing and testing stage, such as updating firmware on the chip and debugging [10]. However, in the literature, JTAG is only utilized to demonstrate some novel PLC firmware tampering attacks, e.g., the PLC rootkits [11] and pin tampering attacks [15]. There is no forensics method that provides guidelines against PLC firmware tampering attacks using JTAG. This work is an effort to fill this gap.

We present Aye, a novel, reliable forensics method against PLC firmware tampering attacks based on the JTAG interface and memory comparison. Aye can accurately detect firmware tampering attacks by reading the memory content of specific PLC blocks and comparing them with the existing original samples. As long as the PLC has a JTAG interface, Aye can detect them without hardware changes. At the same time, Aye does not occupy the PLC computational power nor affect its operation and can detect targets efficiently and quickly.

This work provides the following main contributions:

- We present Aye, a novel firmware forensics method based on JTAG and memory comparison, which can investigate attacks efficiently and accurately.
- We have reversed the central control loop mechanism of Allen-Bradley CompactLogix L18ER PLC and deployed the most advanced PLC rootkits on it to evaluate the forensic effect of Aye.
- We built an actual simulation ICS scenario and verified the effectiveness of Aye in PLC firmware tampering attack forensics.

The rest of this paper is structured as follows. First, we provide the background of our work in Section 2. Section 3 reviews the related work in related forensic methods before providing a general concept of ours in Section 4. Section 5 evaluates our detection method with the most advanced firmware tampering attacks on Allen-Bradley CompactLogix PLC and is concluded in Section 6.

2. Preliminaries: PLC Structure and Security

Unlike traditional security, the most critical to ICS security is not data but rather the continued availability and safe operation of their facilities [16]. A malfunction threat

or attack may cause substantive damages with economic and even life losses in the real world [4]. PLC is the main target in almost all security events against the ICS [17], relating to most potential attacks and threats [18]. Although different PLC manufacturers have significant differences, such as protocols, programming languages, and firmware [19], the firmware tampering attack is a general attack method. To highlight how the firmware tampering attack works and how to investigate such threats, we need to clarify some relevant issues. This section will mainly focus on the PLC architecture and the security risks of the PLC firmware layer. Since Aye utilized the JTAG interface to detect and investigate firmware tampering attacks, we will further illustrate the association between the PLC and JTAG.

2.1. What Is the Architecture of PLC?

The International Electrotechnical Commission (IEC) [20] defines PLC as an electronic system designed for digital operation in an industrial domain. PLC executes user-oriented instructions such as calculation and counting, sequence control, and logical operations [21]. Moreover, a PLC can also control various machinery or production processes in the production environment through digital or analog circuit input/output [22,23]. The PLC's structure primarily comprises three layers: the programming layer, the firmware layer, and the hardware layer, as shown in Figure 2.

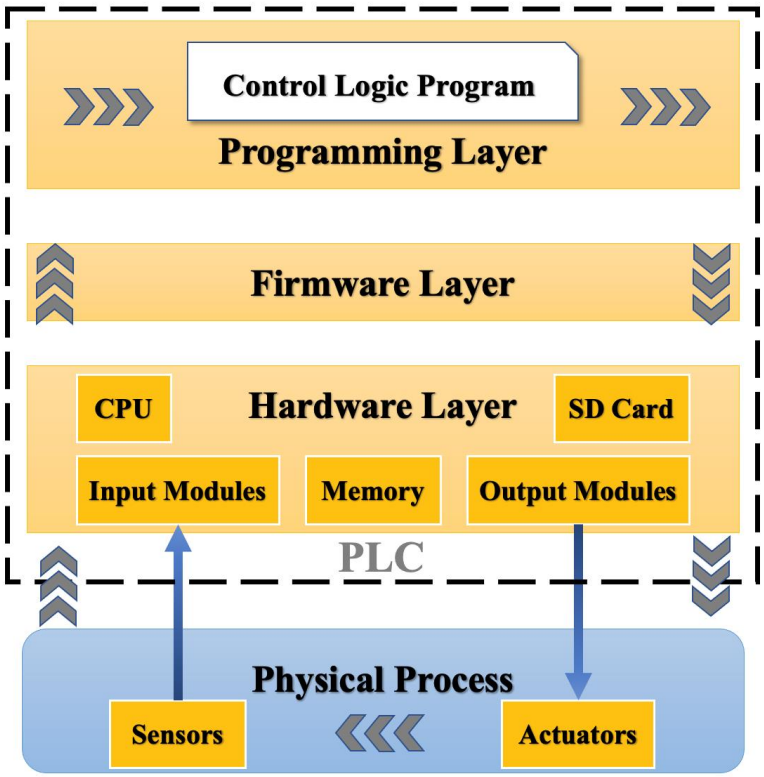


Figure 2. The structure of PLC, which can ultimately construct a control loop.

The programming layer is the primary interaction model between the operator and PLC. Different PLC manufacturers use different programming software to compile the programming language, such as Ladder Diagram (LD), into a lower-level code and load the code into the PLC memory. The code runs in the PLC and determines how to calculate the corresponding output based on the input of the field device.

The firmware layer is the connection between the programming and hardware layers. Firmware is the low-level software that runs on a device that handles all interactions between the user and the device, including physical input and output. Typically, the firmware is referred to as the operating system of an embedded device.

The hardware layer of PLC, like a personal computer, also includes the microprocessor, memory (volatile and non-volatile), and bus. The PLC microprocessor can receive inputs from the operator, collect the status data from filed equipment, read the instruction from PLC memory and generate control signals to control the corresponding circuits through the bus. PLCs are connected to physical devices through input/output modules. The input module receives the electrical signal and transmits it to the internal memory. The output module drives external loads, such as indicator lights, relays, air (oil) pressure valves, and electromagnetic.

2.2. PLC Firmware Layer Security and PLC Rootkits

PLC firmware is similar to the PLC's operating system. An attacker with access to the PLC firmware could control the device without restrictions and even possess the ability to alter the device's behavior covertly. For most PLCs, firmware updating is typically the responsibility of users, which means the user must have proper access to the device. However, the attacker can update malicious firmware to the device simultaneously. Once an attacker can tamper with the firmware and upload it to the PLC, it will be entirely under hostile control. All equipment and production processes configured and controlled by the PLC, such as the external device, will be exposed to attacks.

PLC rootkits are the latest firmware tampering attacks that can covertly damage industrial control systems. A rootkit in traditional PC operating systems refers to a powerful malware that can hide traces of presence. Rootkits are usually implanted by modifying the kernel or drivers and run under the highest privileges. PLC rootkits are very similar while running in PLCs. Since most PLCs do not have advanced functions, e.g., file management and process management, there is no measure to detect whether the system has been invaded. Therefore, the concealment of PLC rootkits mainly refers to the ability to deceive the Human–Machine Interface (HMI) software that monitors the PLC and the hardware characteristics (such as LED lights, etc.) to hide its existence.

2.3. What Is the JTAG Interface of a PLC?

JTAG is an Institute of Electrical and Electronics Engineers (IEEE) standard (IEEE std. 1149.1), which has been adopted by global electronics companies [24]. The main functions of JTAG include debugging, storing firmware, and boundary-scan testing. Debugging based on JTAG allows for the debugging of embedded system software at the machine instruction level. Many CPU architectures (e.g., PowerPC, MIPS, ARM, x86) have built a complete software debug infrastructure, including software debugging, instruction trace, and data trace around the JTAG protocol. Under the control of JTAG, the processor can be halted, single-stepped, or run autonomously, as well as set breakpoints in Random-Access Memory (RAM), Read-Only Memory (ROM), and flash memory.

Through the JTAG interface, the device programmer hardware can transfer data to internal non-volatile device memory and write software and data to flash. The JTAG boundary-scan technology provides access to many logic signals of complex integrated circuits, including device pins. A standard JTAG includes 4-5 pins, which are TDI (Test Data In), TDO (Test Data Out), TCK (Test Clock), TMS (Test Mode Select), and, optionally, TRST (Test Reset). The TRST pin is an optional active-low reset pin. Data are transferred from TDI and output to TDO on every rising edge of the TCK clock, and the clock input is on the TCK pin. A device exposes one or more Test Access Ports (TAPs) in the JTAG interface, which can communicate with the host. For example, to manipulate TMS and TDI in conjunction with TCK for debugging and reading the result through TDO.

Although most PLC vendors remove the header from the JTAG interface on a circuit board, the contact pad is still visible. Many PLCs hold the contact pad with 12-24 JTAG pins organized in 2 rows, e.g., ControlLogix 1756, CompactLogix 1769, Modicon M221, and MicoLogix 1100 [10]. Furthermore, if the PLC processor's pins and datasheets are accessible, JTAG pins can be identified through connectivity tests between the processor-designated pins and the candidate contact-pad pins. There are many JTAG pin detection methods, e.g.,

the JTAGulator [25]. In a sense, JTAG connectivity is possible unless PLC vendors disable the JTAG interface after the circuit testing.

3. Related Work

There are numerous firmware tampering attacks on various types of embedded devices. Cui et al. inject malicious firmware into HP printers with the HP Remote Firmware Update (HP-RFU) protocol [26]. Traynor et al. hack embedded devices and create botnets by manipulating firmware [27]. On the side of the ICS equipment, Wegner exploited a vulnerability in the firmware verification system of the Siemens telephony communication device and installed a backdoor [28]. Peck et al. compromised the Ethernet module of PLC by uploading malicious firmware [29]. Basnight et al. analyzed the PLC firmware update mechanism with reverse engineering techniques, showing that PLCs are vulnerable to firmware tampering attack [30]. Schuett et al. added an exploitable malicious code module to the firmware that can remotely shut down a physical device [31] under specific circumstances (such as a particular time or receiving an exceptional control signal, etc.).

There are two prerequisites for the implementation of firmware tampering attacks. First, verifying firmware integrity and validity is always necessary during updating, so the attacker must bypass the verification mechanism. The second is that the firmware is always a black box, which is hard to evade verification. Santamarta [32] and Peck et al. [29] made essential contributions to PLC firmware reverse engineering, discovered backdoors in the firmware, and determined the verification algorithm used by the ControlLogix Ethernet module. Z. Basnight [30] verified the feasibility of the PLC firmware tampering attack with the reverse engineering method. In this work, we will try to deploy the two latest firmware tampering attacks with such technologies.

The latest firmware tampering attacks, e.g., PLC rootkits, are always based on PLC hardware, which can be more harmful and covertly damage ICSs. Abbasi et al. attacked PLC's I/O ports by tampering with the pin configuration [15]. PLC I/O ports are connected to general-purpose I/O (GPIO) pins of the PLC System on Chip (SoC). The pin must be configured with input or output properties before use (pin configuration). In contrast, they can be configured again by writing to registers mapped into the memory during operation. Since the pin configuration does not trigger hardware interrupts, malware can tamper with the I/O port properties, resulting in I/O truncation and damage to equipment. Garcia et al. proposed HARVEY, a PLC rootkit for smart grid industrial control systems bypassing most network traffic-based defenses [11]. HARVEY replaces legitimate control commands with malicious commands specified by the attacker to maximize damage to electrical equipment and cause massive failures. At the same time, HARVEY uses legitimate control commands to calculate and inject false sensor measurement values into the power system and conceal the operator.

On the JTAG-based defense side, Rajput et al. [33] present ORRIS, a lightweight and out-of-the-device framework that detects Linux-based PLC malware at both kernel and user-level by processing the information collected using the JTAG interface. Guri et al. [34] propose JoKER, a JTAG-based framework for detecting rootkits in the Android OS kernel. Konstantinou et al. [35] implement PHYLAX, a JTAG-based monitoring and detection mechanism for embedded devices. Zubair N et al. present PEM [36], which can remotely investigate and acquire PLC memory in ICSs. Rais and Awad et al. implement Kyros [10], a JTAG-based PLC memory acquisition framework that can collect forensic information from the PLC memory at the hardware level. One of the significant advantages of JTAG is that it can be applied to ICS devices with insufficient computational power. In terms of IoT, there are more defense options. For example, the watchdog timer [37] can reset firmware that does not know the actual circuitry. However, the watchdog timer needs to change PLC's software or hardware, which can not be applied to old equipment.

Compared with previous work, we have further expanded the role of JTAG in forensics: JTAG can be used not only for attack deployment or memory acquirement but also for forensics against firmware tampering attacks. The comparison of existing research and Aye

is shown in Table 1. Furthermore, drawing on previous work, we deploy and detect the two latest firmware tampering attacks, the HARVEY and pin tampering attacks, finally demonstrating the effectiveness of Aye.

Table 1. The Comparison of Existing Research and Aye.

Solution	Target Equipment	Function
ORRIS	Linux-based PLC	Kernel and user-level malware detection
JoKER	Android device	Android OS rootkit detection
PHYLAX	Embedded device	Malicious behavior monitoring and detection
PEM	PLC	Remote PLC memory forensic acquisition
Kyros	PLC	PLC memory forensic information collection
Aye	PLC	PLC firmware tampering attacks forensic and detection

4. Aye Methodology

This work targets firmware tampering attack forensics on PLCs with a JTAG interface. We assume the adversary has compromised the PLC with some methods and implanted malicious code, such as HARVEY or pin configuration tampering attack, through firmware tampering attacks. The malicious code may falsify the output of sensors, resulting in reduced production efficiency, increased costs, tampering with the properties of PLC I/O ports, and even causing physical damage or casualties.

To evaluate the effectiveness of Aye, we deployed HARVEY and pin tampering attacks, the latest firmware attacks. Compared with the previous ones, such advanced attack methods are more stealthy and difficult to be detected. It is an excellent forensic indicator that can verify the effectiveness of Aye. In order to ensure that the forensics results are credible, we have established a new security authentication mechanism and a trusted forensics chain. The forensics technology of Aye includes the following advantages:

More Practicability

Due to the limited computational power of PLC and conservative updates, the forensics method should not take up the PLC computational power nor modify the hardware so that it can support the old equipment. Such a requirement is also the advantage of Aye: Aye can effectively investigate the firmware tampering attack as long as the target PLC has a JTAG interface.

More Effectiveness

Firmware tampering attacks, such as HARVEY and pin configuration tampering attacks, are the most advanced PLC attack methods. Due to the limitations of the existing methods for acquiring memory content, such stealthy attacks are often tricky to investigate. This forensics technology of Aye can more effectively detect new firmware tampering attacks than existing defense methods, for Aye has full access to the PLC memory content.

More Credibility

The detection technology should minimize logical vulnerabilities to resist bypassing and confirm the authenticity and validity of the detection results, which cannot be tampered with by the attacker. Malware in firmware can easily bypass general detection methods through some specific techniques. For example, the HMI usually monitors PLCs in ICS, such as receiving data from PLC, and determines the PLC status, which can be deceived by the transmitted forged data [38]. With the PAM present in this work, we can evaluate the forensic result to make it more credible.

4.1. Establishment of the Trusted Forensics Chain

For any attacks that would leave traces in the memory [33], Aye establishes a trusted forensics chain to generate an authoritative result, as illustrated in Figure 3. The chain of forensics begins at the regular PLC's JTAG interface with an unattacked pristine state and

ends with the JTAG adapter, which has full access to PLC memory. The JTAG interface is defined at the hardware level that cannot be tampered with malicious code. Even though the device is invaded and the malicious code has obtained the executable permission of the device, it cannot tamper with the JTAG module. In addition, the PLC is connected through an adapter, which means no attack vector for a man-in-the-middle attack.

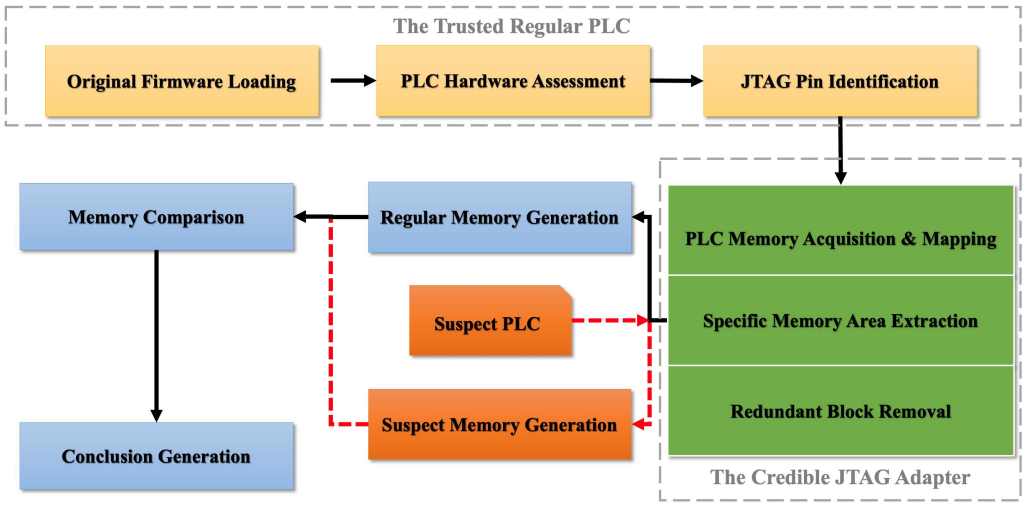


Figure 3. The trust forensics chain begins from the trusted regular PLC (yellow part), ending with a credible JTAG adapter (green part). The orange part is the suspect PLC, and its memory is acquired through Aye. The final step is to compare the acquired memory with the original memory and generate the result, as shown in the blue part.

To evaluate the credibility of the forensics method, we present the PLC Authentication Mechanism (PAM), as shown in Figure 4. There are four entities in PAM; V (Verifier), P (Prover), M (Measurement), and S (Status). V refers to the detection software, and P is the software or hardware running on the PLC that responds to V. P measures the PLC status, S, under the request of V, generates the measurement M, and then transmits it to V. The PLC is not invaded if V considers M valid under any S. On the contrary, if there is a state S' in which M is deemed invalid by V, the malicious intrusion affects the PLC. The credibility of P is the foundation of PAM, which runs on PLC and calculates M. If P can tamper with malicious code, M will not be trustworthy. In addition, the attacker must not reproduce any calculation of M completed by P, even if the attacker knows the calculation method of M and the state data used by P but cannot forge M.

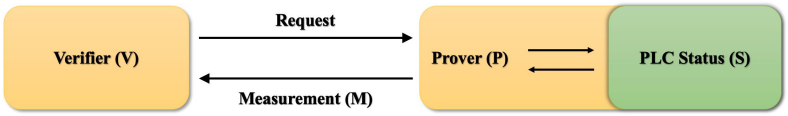


Figure 4. The PLC Authentication Mechanism (PAM), which includes four entities; V (Verifier), P (Prover), M (Measurement), and S (Status).

Except for the tampered firmware, the suspect PLC is consistent with regular PLC in other aspects. While Aye connects to the suspect PLC, it can automatically acquire the memory content from the suspect PLC and determine whether it is under attack. All operations meet the PAM of trustworthiness, as shown in Algorithm 1, meaning the result generated from Aye must be trusted.

Algorithm 1: Verification of PLC Status.

Input: PLC Status
Output: Verification Result(Trusted or Untrusted)

```
Measurement ← getProver(PLC_Status);
while Verifier(Measurement ≠ 'NULL') do
    if Verifier(Measurement) = 'Mismatch' then
        return 'Untrusted';
    end
    else
        sendRequest(PLC_Status);
        Measurement ← getProver(PLC_Status);
    end
end
return 'Trusted';
```

4.2. Forensic Indicators of Aye

In order to ensure the authenticity and effectiveness of the forensic results, we must first clarify the essential characteristics and classification of firmware tampering attacks, which are also Aye's fundamental basis and indicators. The firmware tampering attacks against PLCs are divided into two categories: (1) Preset malicious programs, such as backdoors in the firmware's executable code area. (2) Tamper with the I/O pin configuration to confuse the PLC. All such attacks can make a difference with the original memory, even if the difference is tiny. For the first category, the bootloader reads the firmware's executable code from the flash into the device memory and jumps to the entry point. The malicious codes in firmware will also be mapped, making a difference in memory. For example, Figure 5 shows the comparison between the original memory and the tampered one. The primeval instruction is *ORR*, and the tampered one is *BRANCH*, which only modifies two bytes but still can find traces in the memory. Similar to the first category, the arbitrary nature of pins will not change once PLCs are initialized, which means the corresponding memory will not change.

Figure 5. The result of memory comparison has shown that if even only one instruction (the red brackets) is tampered with, it will leave some traces.

Since various firmware tampering attacks change some PLC memory areas, these changes should never happen without attacks. Under this premise, as long as we can extract the specific memory information of the target through credible methods and compare it with reliable memory not being attacked, we can determine whether the device has been shot.

After clarifying the feasibility of Aye's forensic objectives, some PLC hardware-related issues need to be addressed. The JTAG standards of PLCs of different device types are not consistent. Identifying and effectively obtaining memory is another urgent problem that needs to be solved. The PLC memory can be acquired and mapped only by correctly identifying and connecting the JTAG interface. For the identification and connection of the PLC's JTAG interface, the first step is to evaluate the relevant information of the PLC, such as architecture, hardware, firmware information, etc., which can always be found in PLC vendors' manuals or official websites. Next, which is the most crucial step, is to identify and connect JTAG pins.

There is no official standard for the order of the JTAG physical interface. Although the pins defined by the JTAG standard have the same functions, the JTAG interface reserved by the manufacturer often does not mark the corresponding relationship of pins. This work summarizes three identification methods to confirm the complementary relationship; JTAG pins identification, JTAG connector detection, and the datasheet. If the JTAG arrangement is not the usual one, it can also be probed by the JTAG connector detecting tool, such as the JTAG Finder [39]. The last method [40] is to refer to the chip's datasheet, which provides the JTAG debugging function and indicates the corresponding pins of the JTAG interface. Once a suspected JTAG pin is found, the multimeter or oscilloscope is helpful in the connectivity measurement of pins.

There are two significant advantages of the debugging interface: (1) Memory acquisition is mainly implemented through the debugging tool's memory read/write functions, which is a non-intrusive operation and will not affect the execution of the typical PLC program. (2) The JTAG-based method can read/write all memory and registers with higher permissions, preventing malware from modification.

4.3. JTAG-Based PLC Memory Content Acquisition and Mapping

Although memory content is an essential forensic indicator, which part of the memory content is investigated, what the memory layout of a firmware tampering attack looks like, and how to acquire/map the memory content are still some technical issues that Aye needs to solve.

The PLC firmware is a set of machine-language instructions (opcodes) held in non-volatile memory devices, such as ROM, Erasable Programmable ROM (EPROM), and Electrically Erasable Programmable ROM (EEPROM) [30]. When a PLC starts up, firmware opcodes are loaded into particular memory areas, such as On-chip Static RAM (SRAM), which is also a critical zone for forensics [41]. Each firmware in a specific architecture contains specific amounts of instructions, which means the memory layout between the original firmware and a suspect one can be very different [42]. For example, as shown in Figure 6, the amounts of instructions loaded into memory differ from the original and suspect firmware.

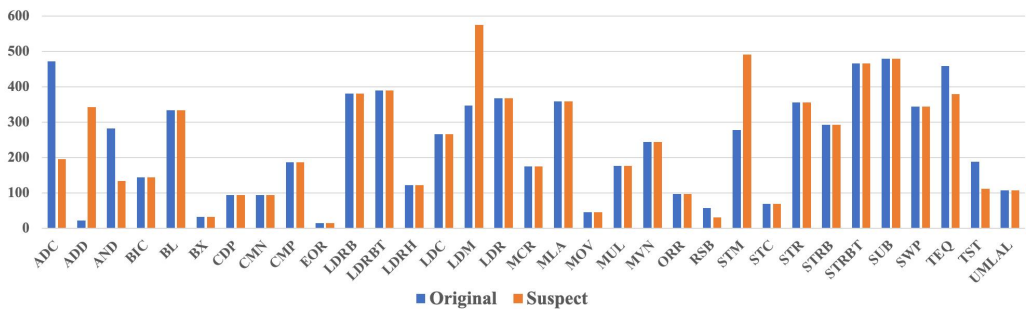


Figure 6. This figure shows the specific instructions and amounts of the suspicious and original firmware. The suspect firmware behaves very differently from the original, so there is also a large difference in the amounts of specific instructions, e.g., ADC, ADD, AND, STM, etc.

For different types of firmware tampering attacks, this paper divides memory acquisition into three categories, the first is overall code extraction, the second is critical code extraction, and the third is pin register extraction [43]. The overall code extraction extracts the entire code segment in the target memory. Firmware tampering attacks mainly tamper the original legal code in the firmware into malicious code. This method can cover all executable codes of the firmware while the extraction time is extended. Critical code extraction refers to only a pre-defined essential segment of memory that is decisive for the device security, such as the ladder diagram operation code, the I/O interface processing code, etc. Compared with the overall code extraction, the memory segment for extraction and execution time of critical code extraction is more diminutive and shorter. Pin register extraction can find abnormal pin configurations, mainly used for forensics of pin tampering attacks.

5. Experiments

To evaluate the practical effect of Aye, we deployed the HARVEY and pin configuration tampering attack on an Allen-Bradley CompactLogix L18ER PLC, as shown in Figure 7. Allen-Bradley CompactLogix L18ER is one of ICS's most widely used PLCs, such as tap water systems, smart manufacturing, and electricity. The PLC firmware was downloaded from the official Allen-Bradley page, and the version is 12.14. The first step is to perform a hardware assessment of the PLC and load the original firmware to extract a safe memory sample. The next step is implementing firmware tampering attacks while investigating such attacks with Aye in the last step. Figure 8 shows the target PLC and ancillary equipment, e.g., the JTAG adaptor, target PLC, and power module (more details are shown in Table 2). For most PLCs that do not reserve a default Command Line Interface (CLI), Aye can connect to the target PLC's JTAG interface and start a CLI through the JTAG debugger software. After that, Aye can perform memory extraction, mapping, etc., through the CLI and finally complete the detection and investigation of firmware tampering attacks.

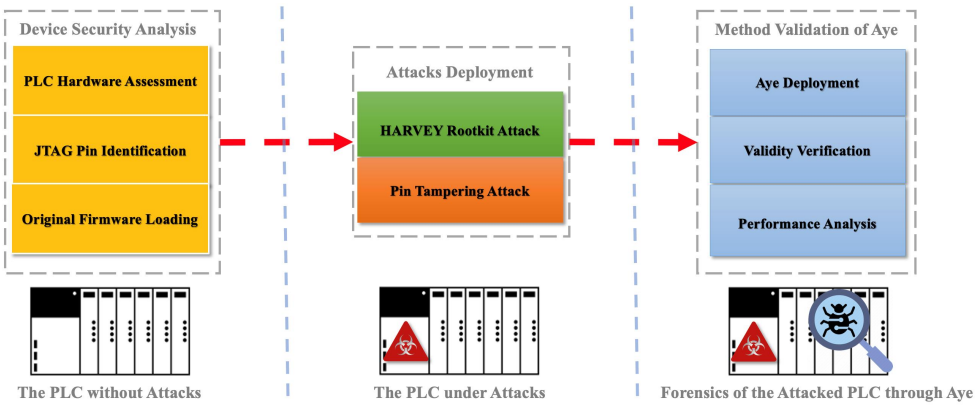


Figure 7. The evaluation scheme of Aye includes the target PLC's different status in different stages.

Table 2. Device Setup for the Evaluation of Aye.

Item	Description
Device Type	Allen-Bradley CompactLogix L18ER
Firmware Version	12.14
PLC Control Software	RSLogix 5000
Device Power	MEANWELL 100–240 V
JTAG Interface Finder	JTAGulator
JTAG Adapter	J-Link JTAG Adapter
JTAG Debugger (CLI)	JLinkExe and OpenOCD
Project PC	core i7-5600. 16GB RAM, Ubuntu 16.04 connected to JTAG interface
Auxiliary Software	IDA Pro and ControlFlash and Keystone
Miscellaneous	Breadboard and Connecting Cables and Soldering Iron and Multimeter

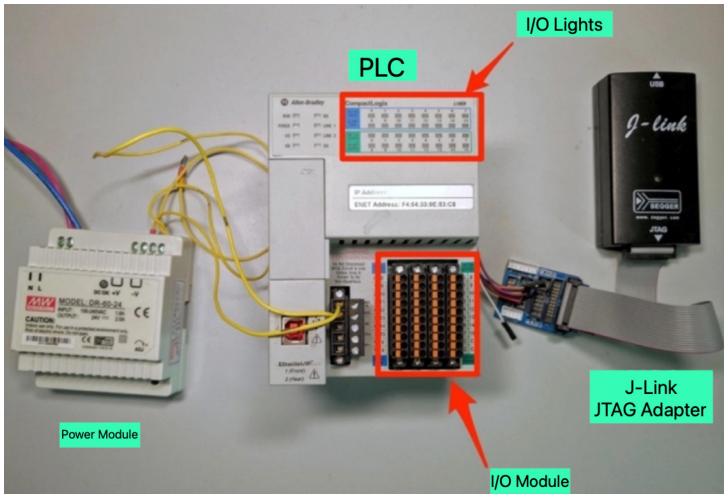


Figure 8. The equipment in evaluating Aye includes the target PLC, a power module of PLC, and a JTAG adapter.

5.1. Device Security Analysis

The general steps for the security analysis of embedded devices are component identification and Printed Circuit Board (PCB) analysis, firmware acquisition and analysis, and hardware debugging, which are the basic steps in this work.

5.1.1. Hardware Analysis

This step aims to analyze the device intelligence, such as processor architecture, debug interface, and flash memory. We disassembled and analyzed the target PLC and found the following components shown in Table 3.

Table 3. Component Chips of Allen-Bradley CompactLogix L18ER PLC.

Chip	Description
FGPA Chip 1	Actel proasc3
FPGA Chip 2	Xilinx spartan xc3s1200e
Flash Memory Chip 1	Miron 29f2g08abaeawp
Flash Memory Chip 2	Mxic a170652
DRAM	Micron d9sbv
ARM Processor	TI lm3s2793 (ARM Cortex-m3)
Other chips	ICE PN-27724

Further analysis of the circuit board shows that the TI lm3s2793 is the CPU chip responsible for the PLC I/O module. The two Field-Programmable Gate Array (FPGA) chips are the high-speed Ethernet and the high-speed USB chip. ICE PN-27724 is the chip responsible for Ethernet and USB interfaces. In addition, there are pads for the JTAG debug interface outside the TI lm3s2793 chip.

5.1.2. Firmware Acquisition and Analysis

The PLC firmware comes from the manufacturer's official website and can be updated to the PLC with the *ControlFlash* software, which Allen-Bradley provides. There are four files in the PLC firmware, as shown in Table 4. The firmware encloses the ARM v7 instruction set and the binary code of the I/O module, indicating that the firmware of the CPU and I/O module are both in the binary file, running an ARM-based operating system, which provides network services. The main CPU is also responsible for updating the firmware of the I/O modules.

Table 4. The Details of Target PLC Firmware.

File Extension	Description
.nvs	Firmware version, Device type
.res	Firmware upgrade verification
.bin	Executable for ARM architecture
.der	Public key certificate

5.1.3. Hardware Debugging

By consulting the datasheet and instrument detection, we found the corresponding relationship between the pads of the TI lm3s2793 and the JTAG pins. After connecting with the J-Link adapter through the JTAG adapter board, the J-Link software can debug the hardware and read the FLASH, ROM, and SRAM of the lm3s2793 chip.

5.1.4. Memory Acquisition and Mapping

For different chip structures, the memory distribution is very different [44]. To emphasize, there are many chips in a PLC, but not all require memory acquisition and analysis. Firmware tampering attacks are mainly against the control logic and I/O ports, so it is only necessary to acquire the memory of the relevant specific chip [45]. In this work, we find the memory distribution of TIlm3s2793 with the datasheet, which is responsible for the control and I/O ports logic. After the analysis of PLC, we will try to verify our method with the two latest firmware tampering attacks in the following section.

5.2. Deployment of Aye

The correct deployment of Aye is a prerequisite for method validation, and we start by connecting the JTAG adapter to the device's JTAG interface. Once the adapter is properly connected, Aye will develop further attack monitoring and surveillance details based on the detected chip information.

For the extracted memory address range of the suspect PLC, the memory layout of the TIlm3s2793 chip is shown in Table 5. On-chip Flash and ROM cannot be modified after leaving the factory [46], while on-chip SRAM is the memory area at risk of firmware tampering [30].

Table 5. The Memory Layout of the lm3s2793 Chip.

Start Address	End Address	Description
0×00000000	0×0001FFFF	On-chip Flash
0×00020000	0×00FFFFFF	Reserved
0×01000000	0×1FFFFFFF	ROM
0×20000000	0×2000FFFF	On-chip SRAM

For port configuration monitoring, the target PLC uses GPIO E and GPIO F ports as input ports and GPIO G and GPIO H ports as output ports. It is only necessary to monitor the input and output mode registers GPIODIR corresponding to these four ports, whose memory addresses are 0×4005C400, 0×4005D400, 0×4005E400, and 0×4005F400. As shown in Figure 9, we set up a constant temperature fermentation tank and assume that the attacker knows the physical process and the mapping between the I/O pins and the logic. The PLC can adjust the fermenter temperature in real time by switching the heater stick (the temperature of the fermentation tank is greater than the outside). Next, we will verify the effectiveness of Aye with specific attacks.

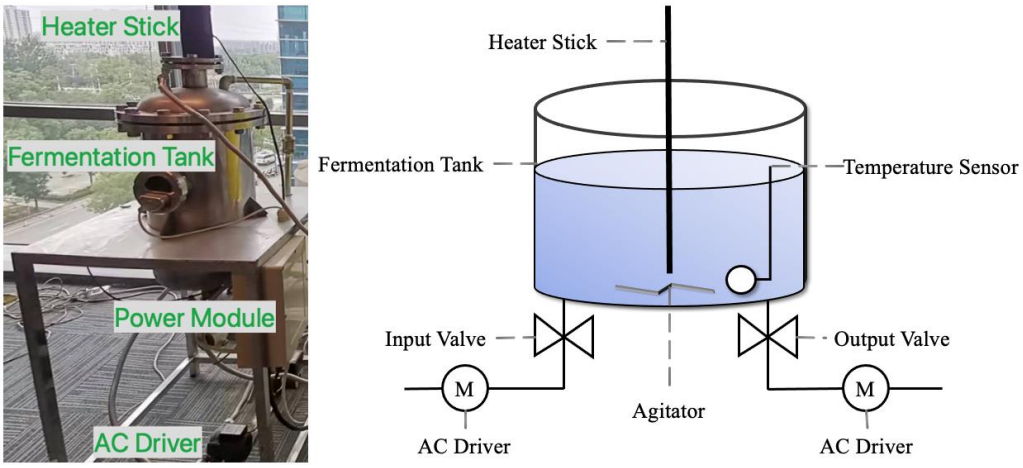


Figure 9. The constant temperature fermentation tank with its corresponding logical control diagram.

5.3. The HARVEY Attack

The PLC I/O ports are connected to the GPIO pins. PLC's processor can read and transmit data through the GPIO pins and output the I/O signals to LED lights and HMI for display. The attacker can hijack valid functions into malicious code and forge the I/O and HMI data by tampering with the relevant code in specific SoC. In this case, we found the related functions in the firmware through IDA PRO software and named them ReadInputPortToMemory and WriteOutputPort, as shown in Figures 10 and 11. The ReadInputPortToMemory function can read dates from GPIO E and GPIO F ports corresponding to the input ports 0–7 and 8–15, respectively. The input ports can control the status of PLC LED lights: a high level is displayed as on, and a low level is shown as off.

```
RAM:20001E18 ReadInputPortToMemory
RAM:20001E18
RAM:20001E18 ; FUNCTION CHUNK AT RAM:2000250E SIZE 00000806 BYTES
RAM:20001E18
RAM:20001E18                 PUSH    {R3-R5,LR}      ; Push registers
RAM:20001E1A                 MOVS    R0, #0          ; Rd = Op2
RAM:20001E1C                 MOVS    R4, R0          ; Rd = Op2
RAM:20001E1E                 LDR.W   R0, =0x4005D3FC ; GPIO Port F, Input Port 8-15
RAM:20001E22                 LDR     R0, [R0]        ; Load from Memory
RAM:20001E24                 LSLS    R0, R0, #8      ; Logical Shift Left
RAM:20001E26                 MOVS    R5, R0          ; Rd = Op2
RAM:20001E28                 LDR.W   R0, =0x4005C3FC ; GPIO Port E, Input Port 0-7
RAM:20001E2C                 LDR     R0, [R0]        ; Load from Memory
RAM:20001E2E                 ORRS    R5, R0          ; Rd = Op1 | Op2
RAM:20001E30                 CODE16
RAM:20001E30
RAM:20001E30 loc_20001E30                            ; CODE XREF: ReadInputPortToMemory+6FA↓j
RAM:20001E30                 LDR.W   R0, =LED_Input  ; Load from Memory
RAM:20001E34                 CODE32
RAM:20001E34                 STRH    R5, [R0]        ; Store to Memory
RAM:20001E36                 UXTH    R5, R5          ; Unsigned extend halfword to word
RAM:20001E38                 MVNS    R0, R5          ; Rd = ~Op2
RAM:20001E3A                 MOVS    R4, R0          ; Rd = Op2
RAM:20001E3C                 LDR.W   R2, =unk_200032F4 ; Load from Memory
RAM:20001E40                 LDR.W   R1, =byte_20003700 ; Load from Memory
RAM:20001E44                 MOVS    R0, R4          ; Rd = Op2
RAM:20001E46                 UXTH    R0, R0          ; Unsigned extend halfword to word
RAM:20001E48                 BL      sub_200021BA    ; Branch with Link
RAM:20001E4C                 MOVS    R5, R0          ; Rd = Op2
RAM:20001E4E                 POP     {R0,R4,R5,PC}   ; Pop registers
RAM:20001E4E ; End of function ReadInputPortToMemory
```

Figure 10. The ReadInputPortToMemory function shown in IDA pro software.

```
RAM:20002324 WriteOutputPort                         ; CODE XREF: sub_2000227A:loc_20002314↑p
RAM:20002324                 PUSH    {R4-R6}         ; Push registers
RAM:20002326                 CODE32
RAM:20002326                 LDR     R5, =dword_200034EC ; Load from Memory
RAM:20002328                 MOVS    R0, R5          ; Rd = Op2
RAM:2000232A                 MOVS    R5, #0          ; Rd = Op2
RAM:2000232C                 MOVS    R1, R5          ; Rd = Op2
RAM:2000232E                 LDR     R5, [R0]        ; Load from Memory
RAM:20002330                 MOVS    R1, R5          ; Rd = Op2
RAM:20002332                 LDR     R5, =LED_Output ; Load from Memory
RAM:20002334                 MVNS    R6, R1          ; Rd = ~Op2
RAM:20002336                 STR     R6, [R5]        ; Store to Memory
RAM:20002338                 MVNS    R5, R1          ; Rd = ~Op2
RAM:2000233A                 MOVS    R4, R5          ; Rd = Op2
RAM:2000233C                 MOVS    R5, #0          ; Rd = Op2
RAM:2000233E                 MOVS    R1, R5          ; Rd = Op2
RAM:20002340                 MOVS    R5, #0          ; Rd = Op2
RAM:20002342                 MOVS    R2, R5          ; Rd = Op2
RAM:20002344                 CODE16
RAM:20002344
RAM:20002344 loc_20002344                            ; CODE XREF: WriteOutputPort+40↓j
RAM:20002344                 UXTH    R2, R2          ; Unsigned extend halfword to word
RAM:20002346                 CODE32
RAM:20002346                 CMP     R2, #0x10       ; Set cond. codes on Op1 - Op2
RAM:20002348                 BCS     loc_20002366    ; Branch
RAM:2000234A                 UXTH    R4, R4          ; Unsigned extend halfword to word
RAM:2000234C                 ASRS.W  R5, R4, R2      ; Arithmetic Shift Right
RAM:20002350                 ANDS.W  R5, R5, #1      ; Rd = Op1 & Op2
RAM:20002354                 MOVS    R3, R5          ; Rd = Op2
RAM:20002356                 UXTH    R3, R3          ; Unsigned extend halfword to word
RAM:20002358                 RSBS.W  R5, R2, #0xF    ; Rd = Op2 - Op1
RAM:2000235C                 LSLS.W  R5, R3, R5      ; Logical Shift Left
RAM:20002360                 ORRS    R1, R5          ; Rd = Op1 | Op2
RAM:20002362                 ADDS    R2, R2, #1      ; Rd = Op1 + Op2
RAM:20002364                 B       loc_20002344    ; Branch
RAM:20002366 ; ---------------------------------------------------------------------------
RAM:20002366                 CODE16
RAM:20002366
RAM:20002366 loc_20002366                            ; CODE XREF: WriteOutputPort+24↑j
RAM:20002366                 LSRS    R5, R1, #8      ; Logical Shift Right
RAM:20002368                 LDR     R6, =0x4005F3FC ; GPIO Port H, Output Port
RAM:2000236A                 STR     R5, [R6]        ; Store to Memory
RAM:2000236C                 LDR     R5, =0x4005E3FC ; GPIO Port G, Output Port
RAM:2000236E                 STR     R1, [R5]        ; Store to Memory
RAM:20002370                 POP     {R4-R6}         ; Pop registers
RAM:20002372                 BX      LR              ; Branch to/from Thumb mode
```

Figure 11. The WriteOutputPort function shown in IDA pro software.

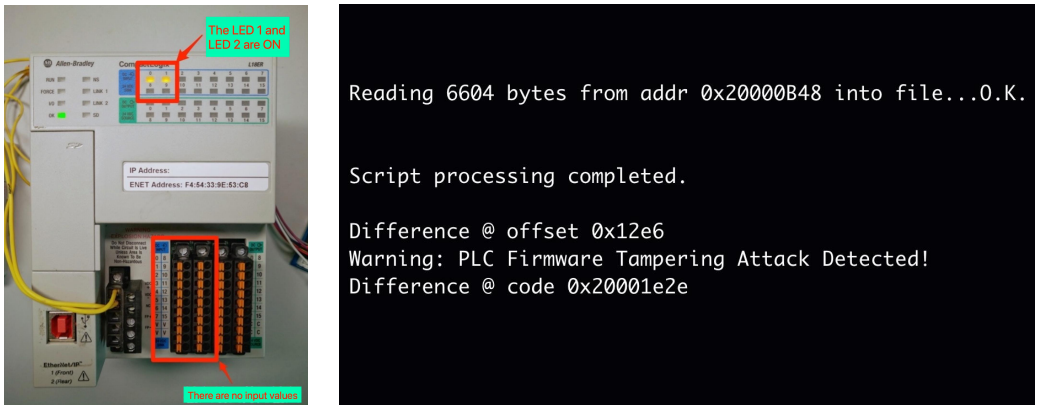
According to the description, HARVEY will falsify the PLC output data by tampering with the functions related to the GPIO, disrupting the production process, and modifying the state of the LED light, so that the PLC device appears to be running normally. This work hijacks the program flow to a new address by rewriting the corresponding code. There are 16 input status LED lights of the target PLC, corresponding to 16 bits of the memory. The value of the bit has two types: 0 and 1. Number 0 means the LED light is off, while 1 means on. We modified the ReadInputPortToMemory function, and the code in address 0×20001E2E has been modified to `B loc_2000250E`, as shown in Figure 12, which means that the program flow is directed to this new memory address. The new memory address stores the modified LED status in a binary value 0b11111100, which means the first and second LEDs will be lit.

```
RAM:20001E18 ReadInputPortToMemory
RAM:20001E18
RAM:20001E18 ; FUNCTION CHUNK AT RAM:2000250E SIZE 00000806 BYTES
RAM:20001E18
RAM:20001E18                 PUSH    {R3-R5,LR}      ; Push registers
RAM:20001E1A                 MOVS    R0, #0          ; Rd = Op2
RAM:20001E1C                 MOVS    R4, R0          ; Rd = Op2
RAM:20001E1E                 LDR.W   R0, =0x4005D3FC ; GPIO Port F, Input Port 8-15
RAM:20001E22                 LDR     R0, [R0]        ; Load from Memory
RAM:20001E24                 LSLS    R0, R0, #8      ; Logical Shift Left
RAM:20001E26                 MOVS    R5, R0          ; Rd = Op2
RAM:20001E28                 LDR.W   R0, =0x4005C3FC ; GPIO Port E, Input Port 0-7
RAM:20001E2C                 LDR     R0, [R0]        ; Load from Memory
RAM:20001E2E                 B       loc_2000250E    ; Branch
RAM:20001E30 ; ---------------------------------------------------------------------------
RAM:20001E30                 CODE16
RAM:20001E30
RAM:20001E30 loc_20001E30                            ; CODE XREF: ReadInputPortToMemory+6FA↓j
RAM:20001E30                 LDR.W   R0, =LED_Input  ; Load from Memory
RAM:20001E34                 CODE32
RAM:20001E34                 STRH    R5, [R0]        ; Store to Memory
RAM:20001E36                 UXTH    R5, R5          ; Unsigned extend halfword to word
RAM:20001E38                 MVNS    R0, R5          ; Rd = ~Op2
RAM:20001E3A                 MOVS    R4, R0          ; Rd = Op2
RAM:20001E3C                 LDR.W   R2, =unk_200032F4 ; Load from Memory
RAM:20001E40                 LDR.W   R1, =byte_20003700 ; Load from Memory
RAM:20001E44                 MOVS    R0, R4          ; Rd = Op2
RAM:20001E46                 UXTH    R0, R0          ; Unsigned extend halfword to word
RAM:20001E48                 BL      sub_200021BA    ; Branch with Link
RAM:20001E4C                 MOVS    R5, R0          ; Rd = Op2
RAM:20001E4E                 POP     {R0,R4,R5,PC}   ; Pop registers
RAM:20001E4E ; End of function ReadInputPortToMemory
```

The Modified Codes in Address 0x20001E2E

Figure 12. The modified function shown in IDA pro software.

When these two LED lights are on, the data of the first and second input ports are high. However, the input port is not connected to any input device. The attack effect is shown in Figure 13a. Although this is only a simple attack scenario, it is enough to prove the destructive capability of this type of firmware tampering attack in the virtual environment. For example, it can run the fermentation facility with extremely high energy consumption while the external display status (including LED lights and HMI interface) is expected. Then, we deployed HARVEY again, and our testing program successfully detected that the firmware had been modified, as shown in Figure 13b. Aye suggests that the tampered address offset is 0×000012E6, which calculates that the memory address at the tampered location is 0×20001E2E. The results show that after being attacked by HARVEY, Aye can perform efficient and accurate forensics on the target PLC and find traces of the attack.



(**a**) (**b**)

Figure 13. The HARVEY attack with its corresponding detection result. (**a**) The attack effect of HARVEY; (**b**) The detection result of HARVEY.

5.4. The Pins Configuration Tampering Attack

PLC I/O ports are connected to GPIO ports of the PLC SoC, and the related properties are set and managed through a series of GPIO registers. Pin configuration attacks will attack such registers, drastically changing the PLC I/O behavior. We need to emphasize that even though the pin configuration tampering attack is a general firmware tampering attack method, the specific implementation details will differ since the pins corresponding to different SoC GPIO registers are different. This is also the most significant difference from HARVEY: HARVEY is biased towards the firmware level, while pin configuration tampering attacks are closer to the hardware level.

The pin configuration tampering attack is implemented by modifying the value of the configuration register corresponding to the pin. For example, Abbasi and Hashemi change the 24th pin of Rasberry Pi to an input pin and the 22nd pin to an output pin [15]. The output pin connects to the LED light, turning it on and off. In this case, the input ports of the TI lm3s2793 are GPIO E and GPIO F, and the output ports are GPIO H and GPIO G. The I/O attribute of the pin corresponding to the GPIO port is defined by one byte (8 bits) of GPIO registers. When the value of a bit is 0, it corresponds to an input pin. Otherwise, it is an output pin.

For the simulation experiment, we set the outside temperature range to 35–65 °C, and the preset temperature of the fermenter is 70 °C. The fermenter's temperature will fluctuate due to the outside temperature and is maintained in a preset temperature range by the PLC. Figure 14 shows the fermenter's temperature without attack. Although the fermenter's temperature will fluctuate with the outside temperature, it will always remain within the preset temperature range. Then, we deployed a pin-tempering attack at minute 28, as illustrated in Figure 15. The corresponding pin was turned from an output property to an input, leading to the fermenter heating up and reaching its maximum temperature.

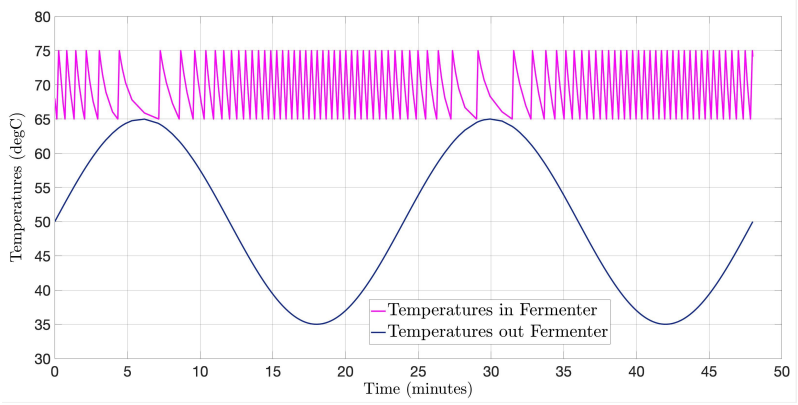


Figure 14. The temperature of the fermenter without attacks.

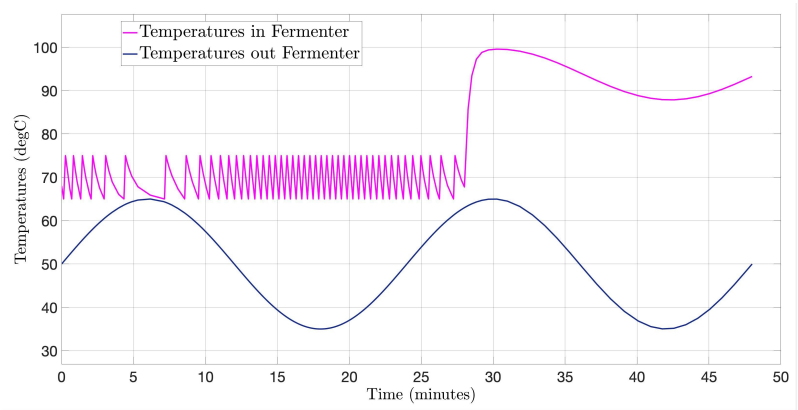


Figure 15. The temperature of the fermenter under attack.

Next, we restore the device state and access the detection device to continuously monitor the device state, followed by a pin tampering attack simulation. Aye can perform a memory extraction for pin registers and compare memory content with the original one. The comparison results are shown in Figure 16, which suggests that output ports 0 to 7 have been tampered with, becoming input ports.

Figure 16. The detection results of the pin tampering attack.

5.5. Results and Performance Analysis

The experimental results verify the actual effect of our detection method based on the JTAG interface and memory comparison. Detecting the most advanced PLC attack methods, such as PLC rootkit (HARVEY) and pin tampering attacks, is very practical. The following tests were conducted in terms of performance: (1) The link time of the JTAG adapter (Link Time, LT); (2) The time to read the entire memory code segment from the device (Read Time, RT); (3) The comparison time spent checking the memory (Comparison Time, RT). All the time is measured ten times and averaged, as shown in Figure 17 and Table 6.

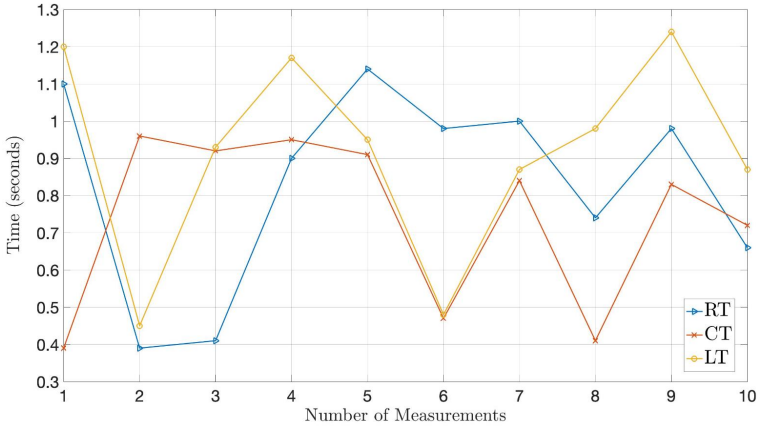


Figure 17. Seconds of measurement for 10 times.

Table 6. The Average Time for Each Measurement.

	Linking Time	Reading Time	Comparison Time
Time (s)	0.74	0.83	0.91

The measurement results show that the average detection time of Aye is within 2.5 s, which ensures that the production environment can quickly respond to emergencies after the equipment is attacked.

5.6. Limitations and Future Work

The utilization of Aye requires establishing a small one-time setup comprising items mentioned in Table 2. Aye's significant limitation requires a regular PLC of the same model as the suspect PLC to extract a piece of "Safe Original Memory". However, as the community of interest involves supporting Aye, more memory will be available, reducing the requirement for the "Memory Acquisition and Mapping" phase. We intend to share the memory dumps for more PLCs of different vendors in the future.

6. Conclusions

With the advent of the industrial Internet era, while intelligence and interconnection have brought about improvements in production efficiency, it has also brought many potential security threats. This paper focuses on the security status of industrial control systems, researches and reproduces the most advanced PLC firmware tampering attacks, and proposes Aye, a novel detection method based on JTAG and memory comparison. We combined software and hardware techniques, such as soldering, debugging, and programming, to prototype this technology. To validate the convincingness, we have deployed firmware tampering attacks on a widely used Allen-Bradley CompactLogix-type PLC as a testbed. The result has shown that our detection method can successfully detect malicious tampering with the firmware within 2.5 s, which verifies the effectiveness. Aye avoids changing the PLC hardware and software as much as possible, and the JTAG memory extraction technology does not interfere with the operation of the equipment. However, in order to connect to the JTAG interface normally, it is still necessary to find the JTAG interface of the circuit board and identify it correctly, and sometimes even soldering is required. The hardware limitation determines that Aye is unsuitable for large-scale detection but appropriate for critical equipment or forensics after an attack.

Author Contributions: Conceptualization, Y.Z. and Z.L.; methodology, Y.L.; software, Y.L.; validation, Y.Z., Y.L. and Z.L.; formal analysis, Y.Z.; investigation, Y.L.; resources, Y.L.; data curation, Y.Z.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z.; visualization, Y.Z.; supervision, Z.L.; project administration, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key R&D Program of China (Grant No. 2021YFB3100905), the National Natural Science Foundation of China (Grant Nos. 62276017, U1636211, 61672081), the Fund of the Key Laboratory of Power Grid Automation of China Southern Power Grid Co., Ltd. (Grant No. GDDKY2021KF03) and the Fund of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2021ZX-18).

Data Availability Statement: Not applicable.

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

References

1. Pan, X.; Wang, Z.; Sun, Y. Review of PLC security issues in industrial control system. *J. Cybersecur.* **2020**, *2*, 69. [CrossRef]
2. Hadžiosmanović, D.; Sommer, R.; Zambon, E.; Hartel, P.H. Through the eye of the PLC: Semantic security monitoring for industrial processes. In Proceedings of the 30th Annual Computer Security Applications Conference, New Orleans, LA, USA, 8–12 December 2014; pp. 126–135.
3. Hareesh, R.; Senthil Kumar, R.; Kalluri, R.; Bindhumadhava, B. Critical Infrastructure Asset Discovery and Monitoring for Cyber Security. In *ISUW 2020*; Springer: Singapore, 2022; pp. 289–300.
4. Zhu, B.; Sastry, S. SCADA-specific intrusion detection/prevention systems: A survey and taxonomy. In Proceedings of the 1st Workshop on Secure Control Systems (SCS), Stockholm, Sweden, 12 April 2010; Volume 11, p. 7.
5. Feng, B.; Mera, A.; Lu, L. P^2IM: Scalable and Hardware-independent Firmware Testing via Automatic Peripheral Interface Modeling. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020; pp. 1237–1254.
6. Zaddach, J.; Costin, A. Embedded devices security and firmware reverse engineering. In Proceedings of the Black-Hat USA, Las Vegas, NV, USA, 27 July–1 August 2013.
7. Awad, R.A.; Beztchi, S.; Smith, J.M.; Lyles, B.; Prowell, S. Tools, techniques, and methodologies: A survey of digital forensics for scada systems. In Proceedings of the 4th Annual Industrial Control System Security Workshop, San Juan, PR, USA, 4 December 2018; pp. 1–8.
8. Qasim, S.A.; Lopez, J.; Ahmed, I. Automated reconstruction of control logic for programmable logic controller forensics. In Proceedings of the International Conference on Information Security, Kuala Lumpur, Malaysia, 19–21 January 2019; pp. 402–422.
9. Senthivel, S.; Ahmed, I.; Roussev, V. SCADA network forensics of the PCCC protocol. *Digit. Investig.* **2017**, *22*, S57–S65. [CrossRef]
10. Rais, M.H.; Awad, R.A.; Lopez, J., Jr.; Ahmed, I. JTAG-based PLC memory acquisition framework for industrial control systems. *Forensic Sci. Int. Digit. Investig.* **2021**, *37*, 301196. [CrossRef]
11. Garcia, L.; Brasser, F.; Cintuglu, M.H.; Sadeghi, A.R.; Mohammed, O.A.; Zonouz, S.A. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In Proceedings of the Network and Distributed System Security Symposium, San Diego, CA, USA, 26 February–1 March 2017.
12. Malchow, J.O.; Marzin, D.; Klick, J.; Kovacs, R.; Roth, V. PLC Guard: A practical defense against attacks on cyber-physical systems. In Proceedings of the 2015 IEEE Conference on Communications and Network Security (CNS), Florence, Italy, 28–30 September 2015; pp. 326–334.
13. Lanotte, R.; Merro, M.; Munteanu, A. A process calculus approach to detection and mitigation of PLC malware. *Theor. Comput. Sci.* **2021**, *890*, 125–146. [CrossRef]
14. Stollon, N. *On-Chip Instrumentation. Design and Debug for Systems on Chip*; Springer Publishing Company: New York, NY, USA, 2011.
15. Abbasi, A.; Hashemi, M. Ghost in the plc designing an undetectable programmable logic controller rootkit via pin control attack. In *Black Hat Europe*; Black Hat: London, UK, 2016; pp. 1–35.
16. Formby, D.; Durbha, S.; Beyah, R. Out of control: Ransomware for industrial control systems. In Proceedings of the RSA Conference, San Francisco, CA, USA, 14–17 February 2017; Volume 4.
17. Smith, K.; Wilson, I. The Challenges of the Internet of Things Considering Industrial Control Systems. In *Privacy, Security And Forensics in The Internet of Things (IoT)*; Springer: Cham, Switzerland, 2022; pp. 77–94.
18. Alanen, J.; Linnosmaa, J.; Malm, T.; Papakonstantinou, N.; Ahonen, T.; Heikkilä, E.; Tiusanen, R. Hybrid ontology for safety, security, and dependability risk assessments and security threat analysis (STA) method for Industrial Control Systems. *Reliab. Eng. Syst. Saf.* **2022**, *220*, 108270. [CrossRef]
19. Ma, R.; Wei, Q.; Wang, Q. A survey of offensive security research on PLCs. *J. Phys. Conf. Ser.* **2021**, *1976*, 012025. [CrossRef]
20. Tiegelkamp, M.; John, K.H. *IEC 61131-3: Programming Industrial Automation Systems*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 166.
21. Jadidi, Z.; Foo, E.; Hussain, M.; Fidge, C. Automated detection-in-depth in industrial control systems. *Int. J. Adv. Manuf. Technol.* **2022**, *118*, 2467–2479. [CrossRef]
22. Erickson, K.T. Programmable logic controllers. *IEEE Potentials* **1996**, *15*, 14–17. [CrossRef]
23. Bolton, W. *Programmable Logic Controllers*; Newnes: London, UK, 2015.

24. Dahbura, A.T.; Uyar, M.U.; Yau, C.W. An optimal test sequence for the JTAG/IEEE P1149. 1 test access port controller. In Proceedings of the 'Meeting the Tests of Time', International Test Conference, Washington, DC, USA, 29–31 August 1989; pp. 55–62.
25. Gupta, A. JTAG debugging and exploitation. In *The IoT Hacker's Handbook*; Apress: Berkeley, CA, USA, 2019; pp. 109–138.
26. Cui, A.; Costello, M.; Stolfo, S. When firmware modifications attack: A case study of embedded exploitation. In Proceedings of the 20th Annual Network & Distributed System Security Symposium 2013, San Diego, CA, USA, 24–27 February 2013.
27. Traynor, P.; Butler, K.; Enck, W.; McDaniel, P.; Borders, K. malnets: Large-scale malicious networks via compromised wireless access points. *Secur. Commun. Netw.* **2010**, *3*, 102–113. [CrossRef]
28. Wegner, S. Security-Analysis of a Telephone-Firmware with Focus on Backdoors. Ph.D. Thesis, Ruhr-Universität Bochum, Bochum, Germany, 2008. Available online: https://git.fabrik17.de/mrgitlab/embedded-multimedia/raw/437afd92da4b438f95fa3efad28564a9d0baffbd/Dokumentation/thesistemplate.pdf (accessed on 1 December 2020).
29. Peck, D.; Peterson, D. Leveraging ethernet card vulnerabilities in field devices. In Proceedings of the SCADA Security Scientific Symposium, Miami, FL, USA, 2009; pp. 1–19. Available online: https://link.springer.com/chapter/10.1007/978-3-642-28920-0_8 (accessed on 26 November 2022).
30. 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]
31. 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]
32. Santamarta, R. Here be backdoors: A journey into the secrets of industrial firmware. In Proceedings of the Black Hat USA, Las Vegas, NV, USA, 21–26 July 2012.
33. Rajput, P.H.N.; Sarkar, E.; Tychalas, D.; Maniatakos, M. Remote Non-Intrusive Malware Detection for PLCs based on Chain of Trust Rooted in Hardware. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS&P), Vienna, Austria, 6–10 September 2021; pp. 369–384.
34. Guri, M.; Poliak, Y.; Shapira, B.; Elovici, Y. JoKER: Trusted detection of kernel rootkits in android devices via JTAG interface. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/ISPA, Washington, DC, USA, 20–22 August 2015; Volume 1, pp. 65–73.
35. Konstantinou, C.; Chielle, E.; Maniatakos, M. Phylax: Snapshot-based profiling of real-time embedded devices via jtag interface. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 869–872.
36. Zubair, N.; Ayub, A.; Yoo, H.; Ahmed, I. PEM: Remote forensic acquisition of PLC memory in industrial control systems. *Forensic Sci. Int. Digit. Investig.* **2022**, *40*, 301336. [CrossRef]
37. Unni, R.K.; Vijayanand, P.; Dilip, Y. FPGA Implementation of an improved watchdog timer for safety-critical applications. In Proceedings of the 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), Pune, India, 6–10 January 2018; pp. 55–60.
38. Faas, M.S.; Kraus, J.; Schoenhals, A.; Baumann, M. Calibrating Pedestrians' Trust in Automated Vehicles: Does an Intent Display in an External HMI Support Trust Calibration and Safe Crossing Behavior? In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, New York, NY, USA, 8–13 May 2021; pp. 1–17.
39. Domke, F. Blackbox JTAG reverse engineering. *Update* **2009**, *1*, 1.
40. Breeuwsma, M. Forensic imaging of embedded systems using JTAG (boundary-scan). *Digit. Investig.* **2006**, *3*, 32–42. [CrossRef]
41. Gao, J.; Xu, Y.; Jiang, Y.; Liu, Z.; Chang, W.; Jiao, X.; Sun, J. Em-fuzz: Augmented firmware fuzzing via memory checking. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2020**, *39*, 3420–3432. [CrossRef]
42. Taylor, J.; Turnbull, B.; Creech, G. Volatile memory forensics acquisition efficacy: A comparative study towards analysing firmware-based rootkits. In Proceedings of the 13th International Conference on Availability, Reliability and Security, Hamburg, Germany, 27–30 August 2018; pp. 1–11.
43. Muduli, S.K.; Subramanyan, P.; Ray, S. Verification of authenticated firmware loaders. In Proceedings of the 2019 Formal Methods in Computer Aided Design (FMCAD), San Jose, CA, USA, 22–25 October 2019; pp. 110–119.
44. Benkraouda, H.; Chakkantakath, M.A.; Keliris, A.; Maniatakos, M. Snifu: Secure network interception for firmware updates in legacy plcs. In Proceedings of the 2020 IEEE 38th VLSI Test Symposium (VTS), San Diego, CA, USA, 5–8 April 2020, pp. 1–6.
45. Park, J.; Jang, Y.H.; Park, Y. New flash memory acquisition methods based on firmware update protocols for LG Android smartphones. *Digit. Investig.* **2018**, *25*, 42–54. [CrossRef]
46. Stüttgen, J.; Vömel, S.; Denzel, M. Acquisition and analysis of compromised firmware using memory forensics. *Digit. Investig.* **2015**, *12*, S50–S60. [CrossRef]