Research on Programmable Logic Controller Security

IOP Conf. Series: Materials Science and Engineering **569** (2019) 042031 doi:10.1088/1757-899X/569/4/042031

Research on Programmable Logic Controller Security

Haolan Wu, Yangyang Geng, Ke Liu, Wenwen Liu*

State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, 450001, China

*LanSaphire@hotmail.com

Abstract. With the convergence of computer technology and industrial networks, attackers are not limited to attacking only individual users' computers, turning to attack industrial control systems that can cause major infrastructure problems. Programmable Logic Controllers (PLC) are the core components of industrial control systems. Its safety has a profound impact on the safety of the entire industrial system. This paper firstly classifies the security research of PLC according to the structure and function, and expounds the existing security defects of PLC from the aspects of firmware security, operation security and program security. Then it summarizes and analyzes four types of security protection measures: the integrity of verification firmware, protocol security encryption, code formal verification, and program security defence detection. Finally, according to the overall safety of the industrial system and the actual development of the current PLC, we discuss the development trend of safety research.

1. Introduction

Industrial control systems (ICS) are usually highly interconnected and interdependent systems that are widely used in key national infrastructure industries such as natural gas, electric power, and nuclear facilities. Therefore, the security of the ICS is the primary prerequisite for ensuring the normal operation of the infrastructure. Unlike traditional computer attacks, which only cause data leakage, network denial of service, and computer damage, attacks against critical infrastructure control devices can even destroy physical equipment and cause irreparable damage to enterprises and even countries.

Since the "Stuxnet" virus outbreak, there have been dozens of attacks on industrial networks. Because PLC is the core component of ICS, it has also been found from events and literature that attacks are all around PLC. For example, in 2010, Iran's nuclear facilities suffered from the "Stuxnet" virus[1]. The attack made the logic of the PLC change, and caused huge losses to the Iranian nuclear program. At the end of 2015, the Ukrainian national grid suffered a "BlackEnergy" malicious virus attack[2], and the Supervisory Control And Data Acquisition (SCADA) system was hit so that a large amount of key storage data was destroyed. In November 2017, Schneider Electric's Triconex Safety Instrumented System (SIS) was attacked by malware "TRITON", which crashed the SIS system by attacking control components such as PLC, and attacked the Distributed Control Systems (DCS) to expand the impact of the attack. It caused many energy plants in the Middle East to stop production.

It is known from the frequent attacks on ICS in recent years that since the replacement of the early relay control device by PLC, the PLC security problem is worthy of attention because the PLC that loses part of the security function to ensure the practicability has become increasingly unable to resist attacks from the network. Therefore, this paper will classify the structure and functionality of PLC, and discuss the research focusing on the security aspects of firmware, operation and program. Finally,

we discuss the two aspects of attack and defense and accord the prospect of future PLC security research.

2. PLC Overview

PLC is a network physics system specifically designed to control industrial systems, and its hardware structure is similar to that of a microcomputer. It is a kind of programmable memory for storing programs internally, performing logic operations, sequence control, timing, counting and arithmetic. One of the main uses of PLC is to control physical equipment in industrial sites. Figure 1 shows the structure of the PLC[3].

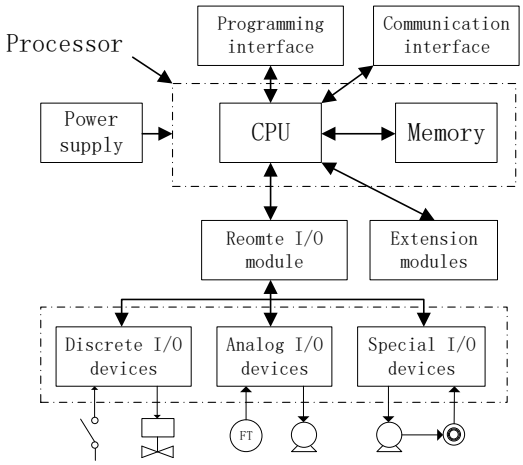

Figure 1. Structure of the PLC

The internal PLC mainly consists of a power supply, a central processing unit, a memory, an input/output interface, a communication interface, and an expansion interface. Based on the internal structure of PLC and the functionality of link interaction, security issues can be studied in three ways:

2.1. PLC firmware

Firmware is the core of PLC, which determines the functional direction and performance of the device. It mainly includes the hypervisor and the instruction interpreter. It is written and solidified by the manufacturer in the memory, and the user cannot access and modify the system program. The function of the hypervisor is to manage the entire PLC so that the internal circuits can work in an orderly manner. The function of the instruction interpreter is to translate a user-written program into a program that the CPU can recognize and execute.

2.2. The operational of PLC

The operational of PLC refers to the task of completing program delivery in a firmware environment. It usually includes the input and output of the status signal exchanged with the peripheral device through the I/O interface; using a proprietary communication protocol to realize the communication with the monitor, the host computer, or other devices; or other operations.

2.3. Program control flow of PLC

The program control flow of PLC mainly refers to the execution process of the running process, usually serving the system and hardware, and plays a vital role in the logic, communication, interaction and connection of the entire PLC.

3. PLC security defects classification

At the BlackHat European Conference in November 2016, according to the structure of the PLC, Ali Abbasi[4] proposed three attack methods for PLC, namely Firmware Modification Attacks (FMA), Configuration Manipulation Attacks (CMA), and Control-flow Attacks (CFA). Like traditional computers, PLC has the security problems of traditional protocol communication and configuration

parameters. Because PLC is also a series of embedded devices, there are also problems such as security defects, memory corruption, and data signal storage. Therefore, the safety of PLC is becoming more and more serious.

This section systematically studies the security flaws of PLC from three aspects: PLC's firmware, operation and program.

3.1. PLC firmware security defects

PLC firmware is vulnerable to Firmware Modification Attacks (FMA), which is caused by an attacker replacing a legitimate functional firmware with malicious firmware. For devices with reprogrammable firmware, the attacker has the opportunity to upload malicious firmware to the device because updating firmware requires appropriate access to the device.

The firmware layer is the core of the bridging operation layer and user program, and is often regarded as the operating system of the embedded device. In a broader sense, the firmware also includes lower-level functions such as initialize and loads the operating system. In some embedded devices, the firmware is installed at the factory and the device cannot be reprogrammed by the user. However, PLC typically has a firmware update feature that enables vendors to fix bugs and upgrade firmware without requiring physical changes to the hardware. The attacker exploited the PLC firmware update feature to develop a firmware replacement attack and firmware tampering attack.

In 2009, by studying the functionality of the upgrade or update firmware provided by the controller, Daniel Peck[5] demonstrated how to load the written malicious firmware into the Ethernet card of two different field devices and showed the firmware replacement attack. And in 2013, Basnight[6] discovered the PLC firmware modification method. The PLC firmware provides a software-driven interface between the system input and the physical output, resulting in easy firmware modification at the user level. Basnight proposed a new firmware analysis method and proof of how to update the legal firmware and upload to the PLC. Furthermore, Schuett[7] analyzed the firmware structure by reverse firmware to modify and add the ability to remotely disable PLC, and suggested some potential mitigations for future firmware development.

The firmware layer controls the basic behavior of the device, including communication with the management system and the execution of compiled user-level programs loaded on the device. Because of it, whether it is firmware replacement or firmware tampering, the attacker's intention is often found by the inspector. Therefore, in order to achieve a more secret attack, at the NDSS conference in 2017, Luis A. Garcia[8] showed how to tamper with the input and output data in the firmware layer, so that the data obtained by the upper application is falsified false data, and implemented a malicious controller and virtualized object model in the firmware layer. It can replace the control signal according to the attacker's intention, and upload the feedback of the object expected by the engineer to the application, which can achieve more accurate concealed attacks. Since the firmware handles all interactions between the user and the device hardware, including physical inputs and outputs, an attacker accessing the PLC firmware has potentially unlimited control over the device, including the ability to secretly change device behavior.

In summary, firmware tampering will allow an attacker to run in a privileged mode with unrestricted access to peripheral devices so that hackers can operate or damage the device. By intercepting and controlling the firmware signal, it is possible to implement a Man-In-The-Middle (MITM) attack deceiving engineer to achieve the purpose of a concealed firmware attack with unpredictable effects. So how to ensure the integrity of the PLC firmware layer has a huge impact on ICS.

3.2. PLC operation security defects

During the operation, the PLC communicates with the PC control terminal through the network communication protocol, and accepts the instruction execution action of the host computer. As figure 2 shown, in general, the PLC has code control system overall logic, use registers to store pin data, and use communication protocols to modify operations. Therefore, PLC is vulnerable to Configuration

Manipulation Attacks (CMA) during operation, which allows the attacker to modify key configuration parameters in the embedded device, and then achieve the purpose of completing the remote operation. According to the modification configuration, it can be divided into the exploitation attack of protocol defects, the tampering attack of I/O interface, and the injection attack of PLC code.

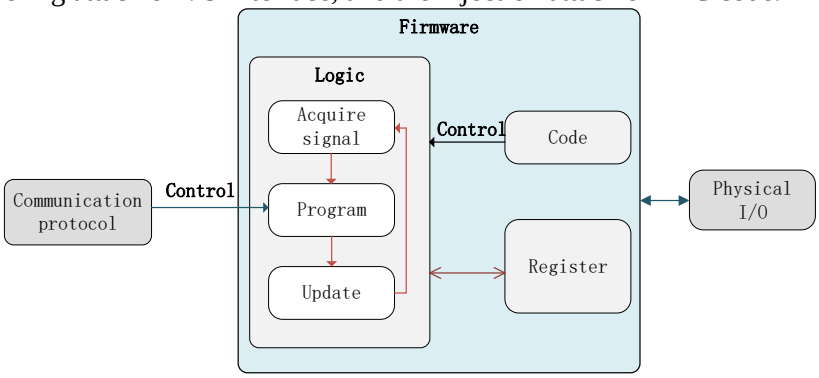

Figure 2. Overview of PLC operation flaws

3.2.1. The exploitation attack of protocol defects. Attackers modify the key configuration parameters of the PLC by utilizing the problems existing in the communication protocol. For instance, in 2011, at the Black Hat conference, Dillon Beresford[9] demonstrated how to use the lack of permissions of the Siemens S7Comm protocol, to bypass the Siemens PLC certification and obtain sensitive information. Since the S7Comm protocol is not encrypted and the data packet is transmitted in plaintext, attackers can capture and carefully constructing the data packet so that PLC can be attacked by replay attack, denial of service, etc. And for the most commonly used Modbus protocol, Morris[10] proposed 17 kinds of attacks against Modbus protocol by studying the function code specification, and divided the attacks into four categories: sniffing attack, response and measurement injection attack, command injection attack and denial of service attack, and described these implementations. In addition to the loopholes in the protocol, its lack of access control is fundamental. Haroon Wardak[11] studied the PLC access control problem and analyzed the PLC's access control mechanism. Stanislav Ponomarev[12] proposed a method for invading ICS by measuring and verifying data transmitted through a network.

The vulnerability of the communication protocol is the premise of the attack and controlling the PLC is the purpose of the attacker. The PLC uses a proprietary protocol for communication. To support Ethernet, the PLC also implements the communication protocol on the TCP/IP protocol stack. It commonly used protocols are Profibus, Modbus, S7, DNP3, and EIB.

Most communication protocols are not encrypted, so sensitive data can be retrieved and the data store of the registers can be read. In addition, the authentication and authorization functions of PLC are missing, so that PLC is vulnerable to replay, traditional hijacking and other traditional attacks.

In this paper, by studying a PLC and analyzing the traffic of communication with the host computer, it can be seen from figure 3 that the Modbus protocol is not encrypted and can obtain sensitive values by capturing packets.

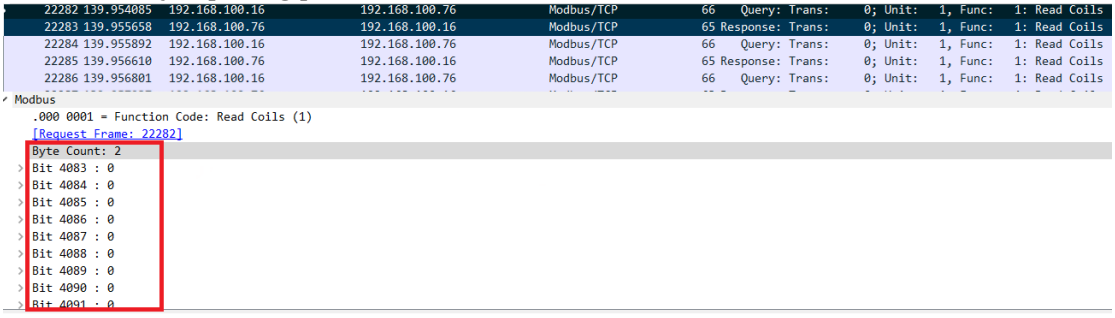

Figure 3. Communication packages of obtaining sensitive values

3.2.2. The tampering attack of I/O interface. It refers that attackers change the inherent behavior by modifying the memory data of the I/O pin controller. In 2016, Ali Abbas[13] researched how to tamper with the device I/O to manipulate the physical process of its control, and achieved privileged authority operation attacks and common user rights operation attacks. In the same year, at the Black Hat European Conference, Abbasi[4] achieved a new attack on the PLC's I/O interface, which changed the operating logic of the system by tampering with the output and input pins.

Analysis of the I/O interface shows that embedded SoCs typically use hundreds of pins connected to the circuit. Some of these pins have a single definition purpose. For example, some only provide power or clock signals. Because of the different I/O requirements, different SoC manufacturers produce a variety of mutually exclusive functions based on the application using a physical pin. Since the I/O pins are usually considered for the overall communication performance of the PLC, there is no protection encryption.

An attacker can manipulate the value read or written from the peripheral by a legitimate process in the PLC, and the I/O pin will ignore the request without throwing an exception. The hardware does not issue an alarm even if the I/O interface status is changed. Therefore, an attacker can manipulate the PLC to read or write to its I/O by using a pin configuration. As shown in figure 4, we use Wireshark to intercept the data of a kind of PLC normal operation, and by modifying the pin configuration and data packets, the purpose of modifying the parameter values is achieved.

3064	8.483996	192.168.100.76	192.168.100.16	Modbus/TCP	66	Response: Trans: 0; Unit: 1, Func: 6: Write Single Register
3065	8.484151	192.168.100.16	192.168.100.76	Modbus/TCP	66	Query: Trans: 0; Unit: 1, Func: 6: Write Single Register
3066	8.485022	192.168.100.76	192.168.100.16	Modbus/TCP	66	Response: Trans: 0; Unit: 1, Func: 6: Write Single Register
3067	8.485218	192.168.100.16	192.168.100.76	Modbus/TCP	66	Query: Trans: 0; Unit: 1, Func: 6: Write Single Register
3068	8.485948	192.168.100.76	192.168.100.16	Modbus/TCP	66	Response: Trans: 0; Unit: 1, Func: 6: Write Single Register
3069	8.486111	192.168.100.16	192.168.100.76	Modbus/TCP	66	Query: Trans: 0; Unit: 1, Func: 6: Write Single Register
3070	8.487432	192.168.100.76	192.168.100.16	Modbus/TCP	66	Response: Trans: 0; Unit: 1, Func: 6: Write Single Register
3071	8.487626	192.168.100.16	192.168.100.76	Modbus/TCP	66	Query: Trans: 0; Unit: 1, Func: 6: Write Single Register
3072	8.495388	192.168.100.76	192.168.100.16	Modbus/TCP	66	Response: Trans: 0; Unit: 1, Func: 6: Write Single Register
3073	8.495850	192.168.100.16	192.168.100.76	Modbus/TCP	66	Query: Trans: 0; Unit: 1, Func: 6: Write Single Register
3074	8.496637	192.168.100.76	192.168.100.16	Modbus/TCP	66	Response: Trans: 0; Unit: 1, Func: 6: Write Single Register
3075	8.496878	192.168.100.16	192.168.100.76	Modbus/TCP	66	Query: Trans: 0; Unit: 1, Func: 6: Write Single Register

Figure 4. Packages of modifying the pin configuration

Because it can be used to change the process of interaction between PLC and physical processes, control the change of its state, and does not cause PLC to issue an alarm, attackers can often control the PLC silently.

3.2.3. The injection attack of PLC code. An attacker can change the operation flow in the industrial control system by modifying the PLC code block or injecting an error instruction, malicious code, etc. into the PLC. Valentine[14] conducted in-depth research on the PLC code design vulnerabilities, and divided the PLC code design-level errors into two types: hardware-based errors and software-based errors. The attacker could exploit the vulnerability to break code logic, perform intermediate code instrumentation, arbitrary code execution, and so on.

Code injection attacks on PLCs usually produce two effects, one is the acquisition of control, such as injection agents, worms, and so on. McLaughlin[15] studied in detail how to construct a malicious payload of PLC in 2011, and in 2012, developed a "SABOT"[16] malware that maps the provided behavioral specification to the victim PLC's code, allowing instantiation of malicious payloads to be effective. In 2015, Johannes Klick[17] showed the results of downloading the SNMP scanner from the PLC, and finally injected a socks proxy, so that the attacker could extend access to all of PLC in the production network through the proxy. And in 2016, Ralf Spenneberg[18] also demonstrated a worm that spreads only in the PLC, which scans the network to acquire new targets and copy itself into new targets that are discovered. The attacker can discover the equipment in the industrial control network through the worm, remotely control the start and stop of the PLC, and change the output value of the PLC.

Another effect is a denial of service attack, the purpose of which is to smash the entire industrial control system. Like in 2017, Govil[19] showed a PLC logic bomb for industrial control systems. The logic bomb is a malware written in a ladder diagram and can be injected into the existing control logic

on the PLC by the attacker. By changing the control action or waiting for a specific trigger signal to activate the malicious behaviour, the PLC can be denial of service, etc.

For PLC, code is the fundamental element of controlling PLC logic. As shown by figure 5, if the code is modified or bypassed, the established logic of PLC can be changed to achieve the attacker's desired purpose.

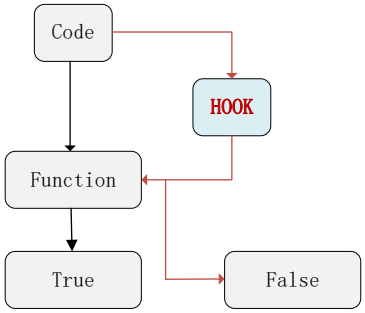

Figure 5. A flow chart of PLC code injection

Usually the attacker exploits the logic of the original code. For example, an object that is defined but not used can be hacked into an attacker's target object. The conditional competition vulnerability in some code can cause unpredictable competition errors, and even the loop in code may become unstoppable. The PLC code is usually programmed by engineers to implement functions, but often ignores the logic adaptation between codes. Therefore, it often suffers from serious problems such as MITM attacks, denial of service attacks, and malicious tampering of controller processes.

The operation configuration defects of PLC are complex and concealed. Because of the endless attacks on the operation configuration, the operation configuration directly affects the normal operation of the controlled device. In the ICS, if only one shutdown occurs so that the execution process is changed, it may have serious consequences.

3.3. PLC program security defects

At present, the PLC program operation is not safeguarded by strong security protection measures like traditional PCs' programs, so it is vulnerable to Control-flow Attacks (CFA). In the case of a normal operation of the PLC controller, the attacker hijacking the control flow of the program, to make the running logic of the program violate the original design goal of the program. It is usually mainly through stack overflow vulnerabilities, release and reuse exploits, bypassing security mechanisms, allowing attackers to execute arbitrary code.

Due to the similarity between embedded devices and real PCs, many studies have shown the possibility of controlling flow attacks in embedded devices. Beresford[20] found multiple vulnerabilities in Siemens PLCs that could allow an attacker to perform a remote code execution attack. Wightman[21] proof Schneider Electric PLC is vulnerable to buffer overflow attacks. Heffner[22] discovered multiple memory corruption vulnerabilities router.

Although there are a variety of techniques currently to detect or prevent control flow attacks, such attacks are still one of the most dangerous attacks. With the development of Internet technology, program flow hijacking attacks of traditional computers have spread in the industrial network, and no means have been developed that can effectively evade attacks without affecting the implementation of functions. In 2012, Vasilis Pappas[23] proposed the kBouncer technology to implement efficient and completely transparent ROP mitigation techniques without source code or debug symbols. In the same year, Ivan Fratrić[24] proposed the ROPGuard framework to protect the program in real time. Cheng[25] proposed the ROPecker framework to effectively defend against ROP attacks without relying on any other auxiliary information or binary rewriting.

But in 2014 Schuster[26] evaluated several detection techniques for controlling flow attacks and believe that an attacker can still bypass them using code sequences in the executable module of the target program. Davi[27] introduced several techniques for bypassing control flow attack detection technology in multiple system security products. The attacker can perform the reverse analysis of the

program and find the applicable way in the binary code to complete the construction of the rop chain to achieve the purpose of hijacking. Therefore, the security of the PLC program is still very serious.

4. PLC security protection measures

Based on the above analysis of the security defects of the PLC firmware layer, operation layer, and program security layer, this section will systematically explore four defense methods.

4.1. Verify the integrity of PLC firmware

Due to the similarity between PLC controllers and traditional computers, most firmware studies often follow the research methods of traditional computer programs, such as Drew Davidson[28], who proposed to use the symbolic execution method to check for vulnerabilities in the firmware program, and tested 99 open source MSP430 firmware programs and found 21 memory-related vulnerabilities. Jonas Zaddach[29] find a way to detect firmware. The article runs firmware on an emulator and interacts with physical I/O devices to dynamically detect vulnerabilities in the firmware. And in 2018 Marius Muench et al.[30] compared the firmware detection framework proposed by Jonas Zaddach and analyzed the shortcomings and challenges in each method.

Another firmware study looks at the firmware itself, how to verify that the firmware itself has been replaced or modified. Mcminn et al.[31] presented a verification tool for PLC firmware in a SCADA system. The tool captures data during the upload and download phase of the firmware and is validated by known legitimate firmware, without any modifications to the SCADA system. In addition, it can analyze firmware using playback capture data without a PLC. Garcia[32] proposed an analysis technique that performs static differential analysis of suspected changed PLC firmware with good firmware, using a variety of test methods to compare firmware versions, models, and code differences, such as deleting, adding, or modifying existing in the original features.

In the detection of the PLC firmware, it is difficult to achieve both the purpose of ensuring the security and ensuring that the performance is not reduced. The firmware detection method is usually adopted to ensure that the firmware is not replaced by detecting the integrity of the firmware, as shown in figure 6. Adelstein et al.[33] introduced a human-based signature-based detection method, which is tested its execution flow and integrity by the detector when it is running.

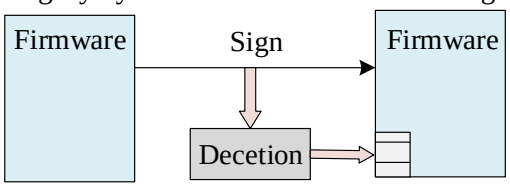

Figure 6. The principle of detecting the integrity of the firmware

4.2. The security encryption of the PLC communication protocol

At present, most PLC communication protocols do not have mechanisms such as encryption and authorization authentication. Therefore, it is very convenient for an attacker to analyze the packets and construct malformed data to change the communication authentication, thereby achieving the malicious purpose of the attack.

Achieved the authentication function is through the interaction of the handshake packet, so that some of the traffic packets intercepted by the attacker cannot be performed without authentication, as shown in figure 7. And the MAC address of the host computer is fixed. If the IP address and the MAC address are bound to the computer, it is difficult for the attacker to conduct a MITM attack from the third-party machine, as shown in figure 8. Nelson[34] proposed to bind the MAC address to ensure security.

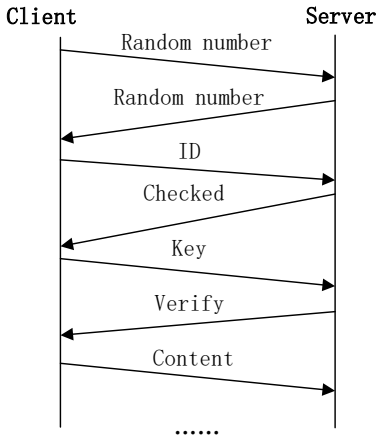

Figure 7. A kind of certified communication

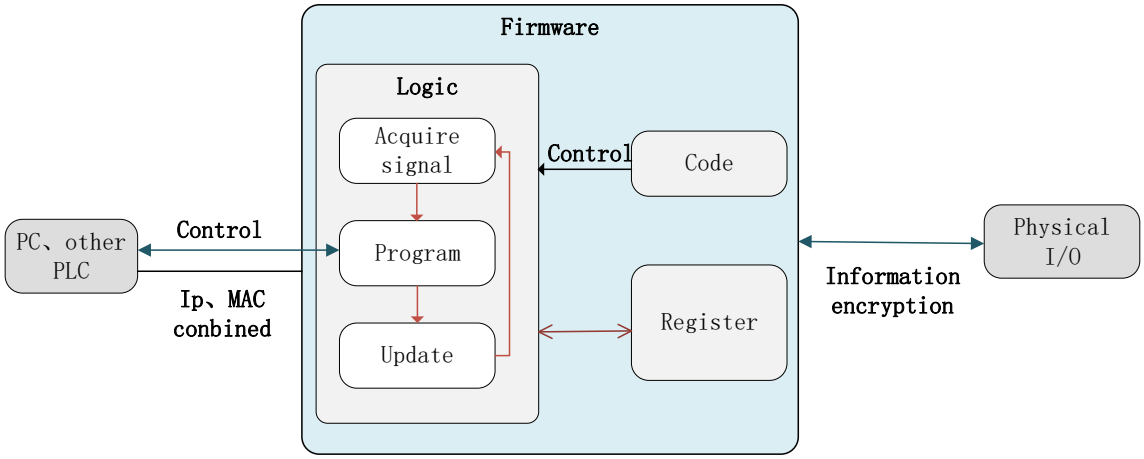

Figure 8. Overview of PLC reliable communication

Most researchers are still looking at how to encrypt communications to ensure data integrity. Heo et al.[35] presented that the PLC communication network in the automation control can be encrypted to ensure the authenticity of the data. Bestak et al.[36] proposed an encryption algorithm used in the PLC network to increase the difficulty of reverse analysis of the communication protocol. However, how to apply the encryption algorithm to the entire industrial control network on a large scale and ensure the normal operation of the service network is still an urgent problem to be solved.

4.3. The formal verification of PLC code

Usually, the defects of the PLC code are extremely difficult to find. The existing methods mainly rely on the security personnel to test the auditing method to avoid the existing problems. But gradually began to use the PLC code formal verification method, which can find a large number of logical defects in the code. The main purpose of code formal verification is to detect PLC code defects and avoid them from being invaded by malicious code. However, because PLC has many programming languages and is not a high-level language, the standards are different and the semantics are complex. It is difficult to analyze and correspondingly model.

Saman Zonouz et al.[37] presented a study for PLC code analysis that used safety engineering to detect and characterize PLC infections for physical damage to power plants. It also draws on control theory, the engineering and mathematics field that deals with dynamic system behaviour, and reverse safety-critical code to identify complex and highly dynamic safety attributes for mixed code analysis methods. However, due to the high cost, it cannot be widely used in code analysis. Malchow[38] proposed the PLC Guard framework technology, which intercepts the flow between the engineering

workstation and the PLC. And Malchow used various levels of graphical abstraction and generalization for formal comparison, which helped the operation and maintenance personnel to correctly handle the accepted code commands, greatly reducing the analysis cost.

Although there is no unified and effective framework for formal verification of PLC code, it is still an important and effective way for manual code auditing. But due to the complexity of its work and the difficulty of modelling, it is extremely difficult to extend the application, so the research prospects are still very broad.

4.4. The security defence detection of PLC program

Ali Abbasi et al.[39] presented a control flow integrity check tool to effectively detect control flow hijacking attacks while ensuring the real-time and availability of the PLC. As figure 9 shows, detect the assembler returns the address and the jump address, etc., and an alarm is issued when the control flow changes, which greatly ensures that the PLC program stream is not tampered with. In the real-time operating system, the priority of the detection task is lower than the priority of the control task, that is, the jump address is detected only when the CPU is idle, so that the real-time control effect of the PLC is ensured to the greatest extent. Saman Zonouz[40] proposed a method of using PLC code symbols to detect malicious code.

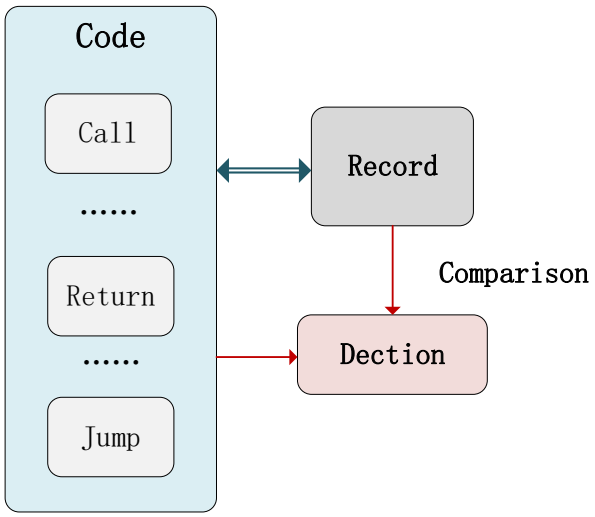

Figure 9. The principle of the detection assembler code

The program flow control problem of PLC is still a problem that plagues security personnel. It is a research direction to learn the protection method of traditional computer programs, randomize the program address, prohibit execution of jumps, etc. However, the PLC program still has some different traditional PC programs. Therefore, control flow integrity detection is an effective means, but the technical means of accurately detecting and reducing overhead has always been the research direction of researchers.

5. Outlook

In summary, there are a large number of research scholars on the safety protection of PLC, but the security protection of PLC is a whole system engineering, and it is not possible to conduct one-sided research from a certain aspect. A simple study from a certain aspect cannot completely protect the vulnerability of the PLC, and will increase the cost. Therefore, in order to protect the safety of PLC, the key research directions of the future research on PLC security research are as follows:

(1) In terms of defence, develop a unique protection framework for PLC holistic research, comprehensively consider cost and functionality, and protect the integrity of PLC integrity from being destroyed. Figure 10 shows a kind of framework.

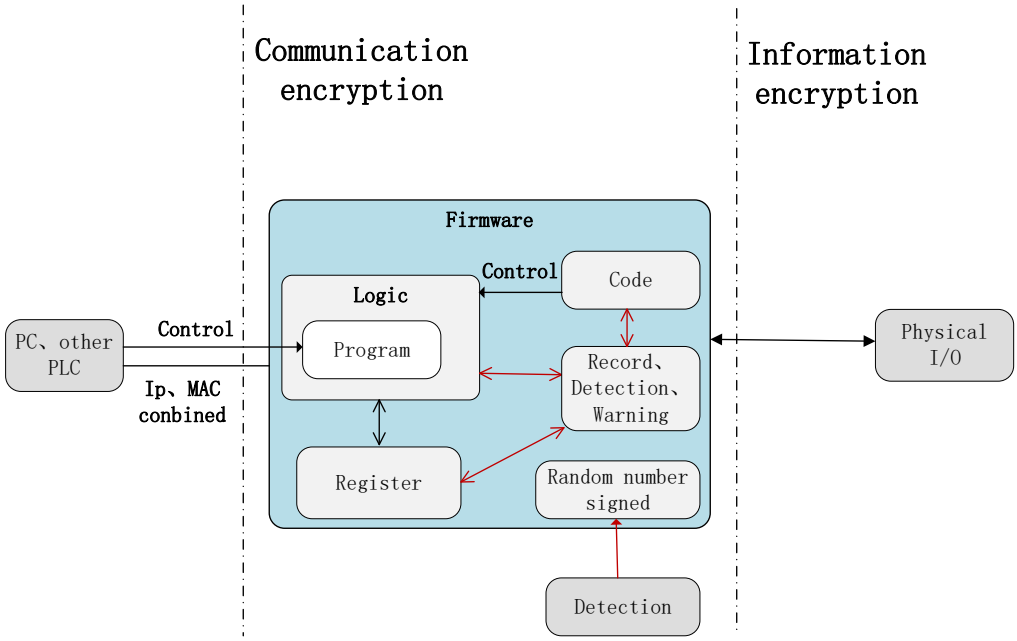

Figure 10. A protection framework for PLC holistic research

First of all, this framework has information encryption to prevent sensitive data from being stolen. Communication transmission encryption is carried out to ensure the authenticity of the data. Secondly, the PLC firmware program is signed online and verified by the detection device to ensure the integrity of the firmware and avoid the tampering of the firmware program. Finally, there should be a test record for the execution flow of the program to prevent tampering of the program control process. Need to consider the structure of PLC, study how to protect the system, and ensure the functional integrity of PLC in real time.

(2) In terms of attacks, the entire system industrial process of ICS needs to be considered to further evaluate the security of the PLC. Nowadays, simply studying the security problem from the structure of PLC itself cannot systematically analyze the existing problems. Any kind of ICS security incident not only attacks the vulnerability from PLC itself, but studies its attack link to form a complete problem. The killing chain completes the attack, achieving the most destructive and influential. Study How to use the Human Machine Interaction (HMI) to attack PLC, to deceive engineering workstations to attack PLC, etc. From the ICS whole system to study its security is the focus of future research.

References

[1] James, P., Farwell, Rohozinski, R. (2011) Stuxnet and the Future of Cyber War. Survival, 53 (1): 23 - 40.
[2] Assante, M.J. (2016) Confirmation of a Coordinated Attack on the Ukrainian Power Grid. SANS Institute, Bethesda, USA, Jan. http://ics.sans.org/
[3] Erickson, K. (2010) Programmable logic controllers: Hardware, software architecture. https://www.isa.org/standardspublications/isa-publications/intechmagazine/2010/december/automation-basicsprogrammable-logic-controllers-hardware-softwarearchitecture.
[4] Abbasi, A., Hashemi M. (2016) Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack.
[5] Peck, D., & Peterson, D. (2009). Leveraging ethernet card vulnerabilities in field devices. In: SCADA security scientific symposium. pp. 1-19.
[6] Basnight, Z., Butts, J., et al. (2013) Firmware modification attacks on programmable logic controllers. International Journal of Critical Infrastructure Protection. 6(2):76-84.
[7] Schuett, C., Butts, J., Dunlap, S. (2014) An evaluation of modification attacks on programmable

logic controllers. International Journal of Critical Infrastructure Protection. 7(1):61-68.
[8] Garcia, L., Brasser, F., Cintuglu, M.H., et al. (2017) Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. In: NDSS.
[9] Beresford, D. (2011) Exploiting siemens simatic s7 PLCs. Black Hat USA, 16(2): 723-733.
[10] Morris, T.H., Gao, W. (2013) Industrial Control System Cyber Attacks. In: International Symposium on ICS & Scada Cyber Security Research. BCS. 2013:22-29.
[11] Wardak, H., Zhioua, S., Almulhem, A. (2017) PLC access control: a security analysis. In: Industrial Control Systems Security. IEEE. 2017:1-6.
[12] Ponomarev, S. (2015) Intrusion Detection System of industrial control networks using network telemetry. Dissertations & Theses-Gradworks.
[13] Abbasi, A. (2016) Ghost in the PLC: stealth on-the-fly manipulation of programmable logic controllers' I/O. CTIT Technical Report Series, (TR-CTIT-16-02).
[14] Valentine, S.E. (2013) PLC code vulnerabilities through SCADA systems. In: University of South Carolina.
[15] McLaughlin, S.E. (2011) On Dynamic Malware Payloads Aimed at Programmable Logic Controllers. In: HotSec.
[16] McLaughlin, S., McDaniel, P., (2012) SABOT: Specification-based payload generation for programmable logic controllers. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, ser. CCS'12. New York, NY, USA: ACM, 2012, pp. 439–449.
[17] Klick, J.，Lau, S.，Marzin, D.，Malchow, J.，Roth, V. (2015) Internet-facing PLCs - A New Back Orifice. In: Black Hat.
[18] Spenneberg, R.，Brüggemann, M.，Schwartke, H. (2016) PLC-Blaster: A Worm Living Solely in the PLC. In: Black Hat.
[19] Govil, N., Agrawal, A., Tippenhauer, N.O. (2017) On Ladder Logic Bombs in Industrial Control Systems.
[20] Beresford, D. (2011) Exploiting Siemens Simatic S7 PLCs. In: Black Hat. USA.
[21] Wightman, R. (2012) Project basecamp at s4. SCADA Security Scientific Sympo- sium. [Online]. Available: https://www.digitalbond.com/tools/basecamp/ schneider-modicon-quantum/
[22] Rapid7. (2014) Linksys wrt120n tmunblock stack buffer overflow. [Online]. Available: http://www.rapid7.com/db/modules/auxiliary/admin/http/linksys tmunblock admin reset bof
[23] Pappas, V. (2012). kBouncer: Efficient and transparent ROP mitigation. Apr, 1, 1-2.
[24] Fratrić, I. (2012). ROPGuard: Runtime prevention of return-oriented programming attacks. Technical report.
[25] Cheng, Y., Zhou, Z., Yu, M., Ding X., and Deng, R. H. (2014) ROPecker: A generic and practical approach for defending against ROP attacks. In: Proc. 21st Annual Network & Distributed System Security Sym. (NDSS).
[26] Schuster, F., Tendyck, T., Pewny, J., Maaß, A., Steegmanns, M., Contag, M., and Holz T. (2014) Evaluating the effectiveness of current anti-ROP defenses. In: Research in Attacks, Intrusions and Defenses, A. Stavrou, H. Bos, and G. Portokalidis, Eds. Springer. pp. 88–108.
[27] Davi, L., Lehmann, D., Sadeghi, A.-R., and Monrose, F. (2014) Stitching the gadgets: On the ineffectiveness of coarse-grained control-flow integrity protection. In: USENIX Security Symposium.
[28] Davidson, D., Moench, B., Ristenpart, T., & Jha, S. (2013). FIE on Firmware: Finding Vulnerabilities in Embedded Systems Using Symbolic Execution. In: the 22nd USENIX Security Symposium (USENIX Security 13). pp. 463-478.
[29] Zaddach, J., Bruno, L., Francillon, A., & Balzarotti, D. (2014). AVATAR: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares. In: NDSS. pp. 1-16.
[30] Muench, M., Stijohann, J., Kargl, F., Francillon, A., & Balzarotti, D. (2018). What you corrupt is not what you crash: Challenges in fuzzing embedded devices. In: NDSS 2018, Network and Distributed Systems Security Symposium, 18-21 February 2018, San Diego, CA, USA.

[31] Mcminn L, Butts J. (2012) A Firmware Verification Tool for Programmable Logic Controllers. In: Critical Infrastructure Protection VI. Springer Berlin Heidelberg. 2012:59-69.
[32] Garcia, A., Mills, R., Butts, J., et al. (2014) Firmware Modification Analysis in Programmable Logic Controllers.
[33] Adelstein, F., Stillerman, M., Kozen, D. (2002) Malicious code detection for open firmware. In: Computer Security Applications Conference. Proceedings. 18th Annual. IEEE, 2002:403-412.
[34] Nelson, T. (2005) Common control system vulnerability. Idaho National Laboratory (INL).
[35] Heo, J., Hong, C.S., Ju, S.H., et al. (2007) A security mechanism for automation control in PLC-based net-works. In: Power Line Communications and Its Applications. ISPLC'07. IEEE International Symposium on. IEEE, 2007: 466-470.
[36] Bestak, I., Orgon, M. (2012) The use of encryption algorithms in PLC networks. Simulation, 2012, 3(64): 168.
[37] Zonouz, S., Rrushi, J., & McLaughlin, S. (2014). Detecting industrial control malware using automated plc code analytics. IEEE Security & Privacy, 12(6), 40-47.
[38] Malchow, J.O., Marzin, D., Klick, J., et al. (2015) PLC guard: A practical defence against attacks on cyber-physical systems. In: Communications and Network Security (CNS), 2015 IEEE Conference on. IEEE, 2015: 326-334.
[39] Abbasi, A., Holz, T., Zambon, E., & Etalle, S. (2017). ECFI: Asynchronous control flow integrity for programmable logic controllers. In: Proceedings of the 33rd Annual Computer Security Applications Conference. ACM. pp. 437-448.
[40] Zonouz, S., Rrushi, J., Mclaughlin, S. (2014) Detecting Industrial Control Malware Using Automated PLC Code Analytics. IEEE Security & Privacy. 12(6):40-47.