How Are Industrial Control Systems Insecure by Design? A Deeper Insight Into Real-World Programmable Logic Controllers

Adeen Ayub, Wooyeon Jo, Syed Ali Qasim, and Irfan Ahmed | Virginia Commonwealth University

Programmable logic controllers (PLCs) have design features to enable operations, such as real-time control of physical processes. These features have weaknesses, making PLCs vulnerable to attacks (network/firmware based). We study these features and attacks and suggest security requirements for designing a PLC.

Industrial control systems (ICSs) monitor and control industrial physical processes, such as nuclear plants, oil and gas pipelines, traffic lights, etc.[1] Figure 1 presents a typical example of an ICS environment. It consists of a control center and a field site. The field sites use programmable logic controllers (PLCs), sensors, and actuators to control the physical processes. For instance, in the case of a conveyor belt that sorts metal and plastic objects, the PLC receives different sensor data and then processes these data using a control logic to make sure the belt runs and sorts the objects accurately. The control center runs ICS services, such as a human–machine interface (HMI), control server, historian, and engineering workstation. The HMI shows the current state of the physical process. At the same time, the historian keeps logs of the PLC's input and output data for forensic and analytic purposes. The control server communicates with the field site over the network. The engineering workstation runs the engineering software, which is provided by the PLC vendor. A control engineer uses the engineering software to download (write) and upload (read) a control logic program on and from a PLC, respectively, to control and maintain the connected physical process. The IEC 61131-3 standard defines five languages to write a control logic: ladder logic, instruction list, functional block diagram, structured text, and sequential flowchart.

PLCs have common design features across different vendors to enable engineering operations, such as real-time control and monitoring of a physical process, use of the scan cycle to run a control logic continuously, etc. The security capabilities are, however, compromised and neglected, thereby making the PLCs inherently insecure. The vulnerability of PLCs to security threats has been a known issue for some time, and researchers have highlighted this problem in their studies.[11] In this article, we focus on common design features in PLCs and study how these features have weaknesses that make them exploitable. For each feature, we show attacks exploiting that particular feature. For instance, the scan cycle of a PLC is needed to run

Digital Object Identifier 10.1109/MSEC.2023.3271273
Date of current version: 23 May 2023

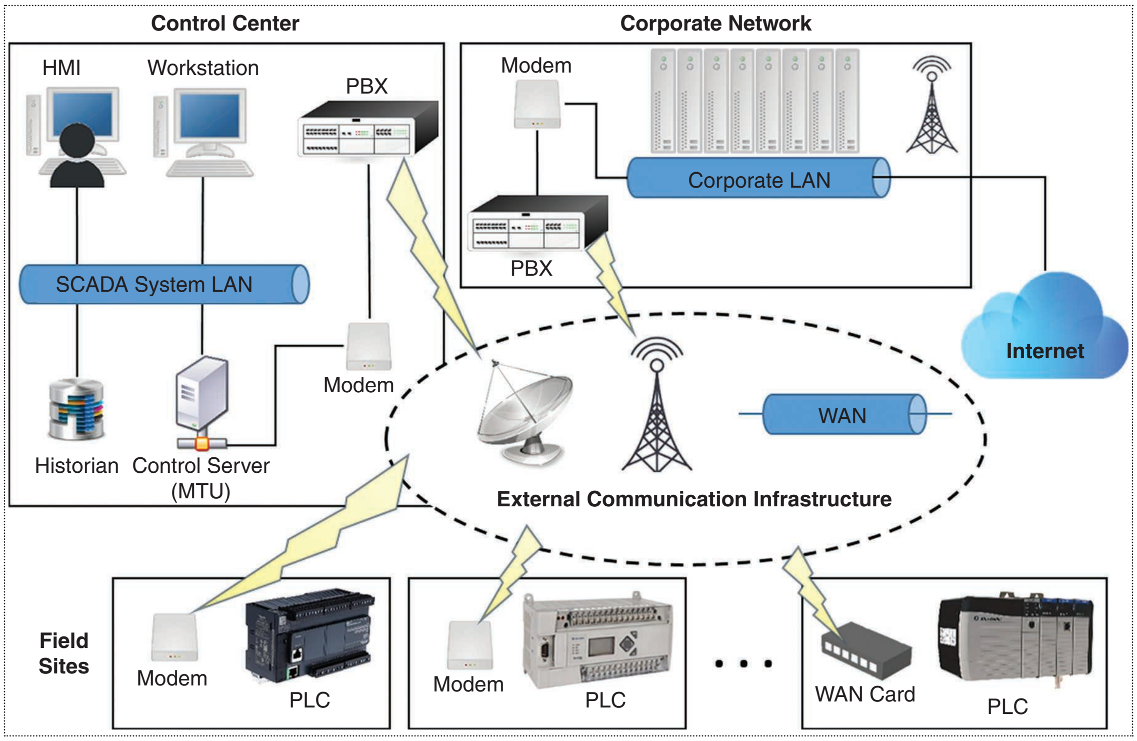


Figure 1. Overview of an ICS. PLC: programmable logic controller; HMI: human–machine interface; PBX: private branch exchange; LAN: local area network; MTU: maximum transmission unit; SCADA: supervisory control and data acquisition.

a control logic continuously. This feature is exploited in a direct firmware object manipulation attack, where the attacker injects malicious code to the existing control logic to make the malicious change effective with every scan cycle. The malicious code modifies a jump table entry (in the firmware) to manipulate the timer function. Finally, we discuss security requirements that should be taken into consideration for PLC design.

PLCs

PLCs are embedded devices that are programmed to automate and control the physical processes in an ICS environment.

Figure 2 shows a typical architectural layout of a PLC.[2] It has input–output (I/O) ports through which the physical process is connected. For instance, push buttons, sensors, and switches are connected to the input ports of a PLC, while the lights, relays, etc., are connected to the output ports. It has a firmware as well as a hardware component with a random-access memory (RAM) and CPU. Some part of the PLC's memory (nonvolatile) is assigned to the control logic program, which is written via the engineering software. The PLC communicates with the engineering software in the form of request/response messages with the PLC acting as a server while the engineering software acts as a client. Each PLC uses an ICS protocol for communication. Some embed their proprietary protocols over well-known protocols, such as Modbus, ENIP, and DNP3.

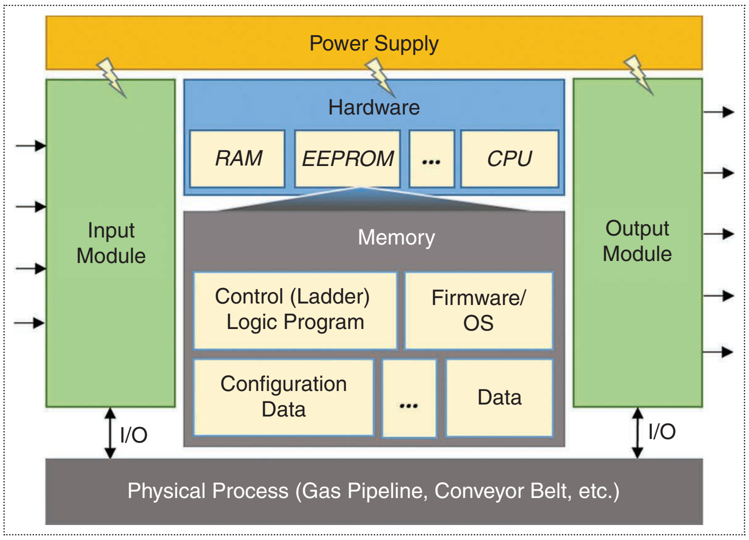


Figure 2. All PLCs typically have I/O modules and hardware including nonvolatile memory (EEPROM). I/O modules, such as sensors and actuators, are attached to perform physical processes. I/O: input–output; EEPROM: electrically erasable programmable ROM; RAM: random-access memory; OS: operating system.

PLC Design Features

PLCs have common design features to enable engineering operations, including the following. The subsequent sections elaborate on them along with their potential exploitation.

- *real-time control and monitoring* of a physical process, such as controlling the gas pressure in a gas pipeline
- *user authentication* to allow only authorized remote access to a PLC
- *scan cycle* to run a control logic repeatedly
- *remote PLC maintenance* through an engineering workstation
- *control logic decompilation* by engineering (programming) software to retrieve and decompile a control logic from a PLC
- *support for ICS network protocols* to enable communication between field sites and the control center and among PLCs
- *device operation modes* consisting of program, run, remote, and test
- *connection to I/O devices*, such as sensors and actuators attached to a physical process.

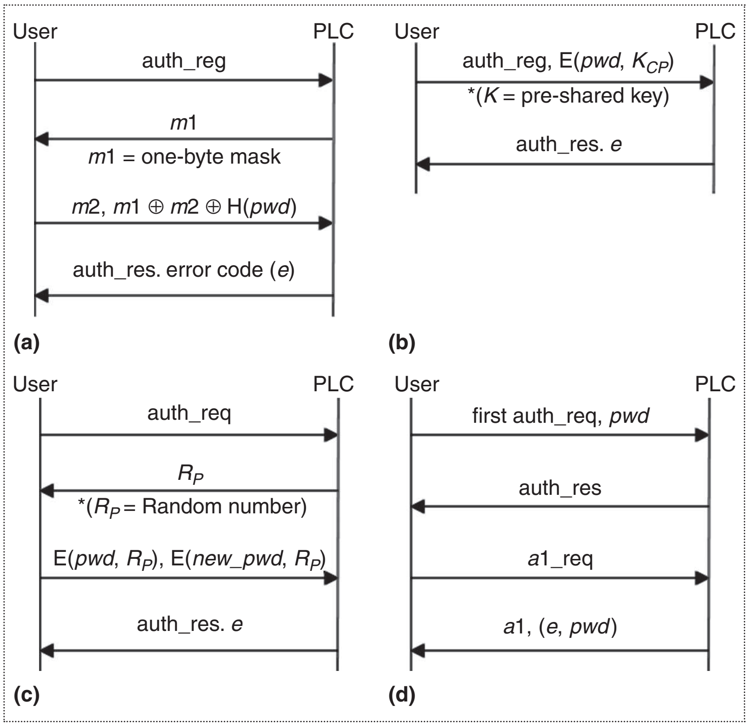


Figure 3. Authentication mechanisms may vary from vendor to vendor, but the design issues with each protocol are quite similar.[3] (a) Schneider Electrics' Modicon M221 authentication protocol, (b) Siemens' S7-300 authentication protocol, (c) Allen-Bradleys' MicroLogix 1400 (Enhanced Password Security) password set/reset protocol, and (d) AutomationDirect's CLICK authentication protocol.

Real-Time Control and Monitoring

A PLC's primary design requirement is the real-time control of changes in a monitored physical process. For instance, the Schneider Electric Modicon M221 PLC is supposed to control motion at the submillisecond level. Any device features that may compromise the real-time control are unsuitable for PLC engineering design.

Since ICSs were originally isolated environments with no connectivity to the outside world, they were not resilient against cyberattacks. In recent years, ICSs are increasingly connected to corporate intranets and other IT networks to gain economic advantages. However, the integration of ICSs and the IT world has exposed ICS environments to cyberattacks.

There is a dire need to incorporate cybersecurity solutions in ICS devices and networks. However, considering the legacy nature of ICSs, adding security features (such as encryption, message authentication, and device memory protection) to a PLC is challenging while maintaining the required speed with which a PLC operates and responds to changes in a physical process. The vendors may choose to upgrade a PLC's hardware. However, this will raise the PLC's cost and not cover the PLCs that are already deployed and functional. These PLCs tend to last for decades; replacing them before their end of life involves substantial costs.

User Authentication

PLCs use password-based authentication. Engineering software can set it up while configuring a PLC. A recent study on PLC authentication protocols reveals fundamental issues, including shared passwords, one-way authentication, and weak encryption and encoding.[3] It involves the PLCs of different vendors, including Schneider Electric's Modicon M221, Siemens' S7-300, Allen-Bradley's MicroLogix 1100/1400, and AutomationDirect's CLICK PLC.

Shared Group Password

PLCs authenticate using a single user group that shares a password. They do not require identification data, such as the username, as part of the authentication process. PLCs consider the communicating user as an authorized entity if the user knows the correct password.

One-Way Authentication

PLCs use one-way client authentication. They require authentication from the engineering software (client) but not vice versa. The PLCs as a server do not authenticate with the engineering software.

Weak Password Encryption on Siemens S7-300

Figure 3(b) shows the weak authentication protocol of the Siemens PLC involving a preshared key of one byte, "K," encrypting an 8-byte password. The encryption algorithm consists of the following steps.

1. First, all eight characters of the password are passed through an encoder that encodes each character according to a substitution table. Figure 4 shows the encoded values for a few characters.
2. Next, the first two encoded values are XORed with *K* to produce *E1* and *E2*, while the rest of the characters are XORed with *K* and *E(i-2)* to produce *Ei*, where *i* can be a whole number from 3 to 8. The encryption algorithm is quite weak. It does not have sufficient diffusion and confusion layers. An attacker can recover the one-byte key from just one plaintext/ciphertext pair. The substitution is conducted at a byte-by-byte level (character by character), making it trivial to reverse engineer the substitution table.

The Siemens S7-300 PLC consists of different blocks, such as an organization block, a functional block, a data block, and a system data block (this block contains the encrypted password). An attacker sends a read request for the system data block to capture the encrypted password. Using the predetermined encryption algorithm, the attacker can decrypt the password.

Weak Password Encoding on Modicon M221

The Modicon M221 employs password encoding during the authentication process to hide the password hash in transit. However, the encoding scheme is weak and can reveal the hash to the attacker upon eavesdropping. Figure 3(a) shows the protocol using two masking bytes, each of size one byte. It XORs both masking bytes with each byte of the SHA-256 password hash. Since the masking bytes and encoded password hash are exchanged between a PLC and engineering software, the attacker can eavesdrop and decode the hash.

Denial of Password Authentication Service on Allen-Bradley MicroLogix 1400

The password set/reset protocol of the MicroLogix 1400 is shown in Figure 3(c). The engineering software first sends an authentication request to which the PLC responds with a random 20-byte challenge. The user then sends a 40-byte response, with the first 20 bytes being the old password encrypted with the challenge and the last 20 bytes being the value of the new password encrypted with the challenge. The PLC checks the value of the first 20 bytes received to confirm if the entered password was correct. If it is, the authentication is successful, and the password is updated to a new value. An attacker can intercept this network traffic and update the value of the last 20 bytes with a random value. Since the first 20 bytes were unchanged, the password set/reset operation is successful, and the new password is set to a random value unknown to the legitimate user.

Poor Password Management on CLICK PLC

CLICK's authentication protocol in Figure 3(d) has a number of problems. 1) The password is transmitted in plaintext. 2) The PLC has a global state that indicates if it is authenticated. So if a legitimate user is successfully authenticated, it allows all other devices to communicate with it without authentication required. 3) The PLC stores the last entered password in credential stores that can be accessed by sending a read request to these stores.

Scan Cycle of Control Logic

A scan cycle runs a control logic continuously and repeatedly to ensure that a connected physical process does not halt at any given time. In each cycle, a PLC gathers data from input devices, such as sensors, and runs the control logic program while updating the output data associated with actuators to control a physical process.

Engineering workstations can update a control logic in a PLC remotely. This feature inherently enables remote code injection, allowing attackers to append the control logic with malicious code. The scan cycle takes care of the execution of the attacker's code to target the PLC's memory regions that are inaccessible through

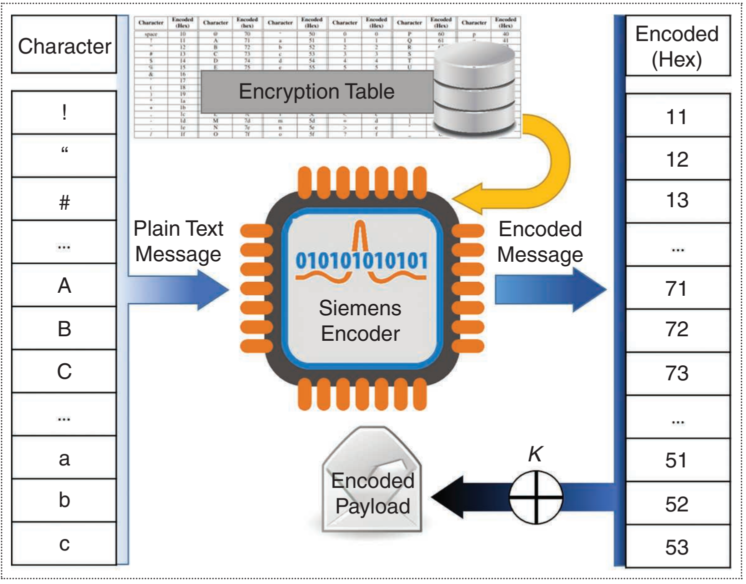


Figure 4. The encryption algorithm of the Siemens S7-300 has a scheme that performs XOR with key (*K*) after encoding. Because of the weak scheme of the encryption algorithm, the security design of cheap PLCs is insufficient, so that even the encryption table can be identified through reverse engineering.

ICS protocols. Following are two examples; one targets the firmware jump table, while the other manipulates a PLC stack for return-oriented programming (ROP).

Direct Firmware Object Manipulation

Direct firmware object manipulation (DFOM) targets a firmware data structure to perform an attack. For instance, the M221 PLC maintains a jump table containing the addresses of the firmware's built-in functions, such as the timer and counter used by the control logic.[9] DFOM compromises a control logic by targeting a function in the jump table. Specifically, it modifies the jump table to hijack the timer function of a control logic that turns on a light after a delay of 3 s. DFOM first injects a malicious payload for the timer (which disables the delay in turning on the light) in the on-chip RAM region of the PLC and then modifies the jump table with the address of the malicious timer code.

ROP

ROP is an exploitation technique that uses gadgets in a device's memory to run malicious code.[10] A gadget is a block of machine instructions that ends with a 'return' instruction. An attacker populates the stack with the memory addresses of gadgets to execute them in a sequence forming a malicious code.

ROP on a PLC is a possible ICS attack and is different from ROP attacks in the IT world. It does not require taking administrative privileges (such as getting a shell) or complete control of the target device. Instead, the attacker needs a set of gadgets that ultimately manipulate the output ports of a PLC to sabotage the connected physical process. For example, in the case of PLCs that use the RX Renesas architecture, register R0 is the stack pointer, R12 maps to input ports, and R13 maps to output ports of a PLC. Figure 5 shows a few gadgets that manipulate the value of the R13 register. An attacker can use these gadgets to update the value of a PLC's output ports, affecting the correct functioning of the connected physical process.

The other differences include installing gadgets in the stack without user input or buffer overflow and ensuring that the attack runs in each scan cycle and not once when an attacker installs the gadgets. Our successful ROP attack requires an initial attack vector involving appending a small stack modification code (which modifies the contents of the stack with the attacker's gadgets) to the existing control logic. As the gadgets execute, they are removed from the stack. However, the stack modification code repopulates the stack in each scan cycle when it runs along with the original control logic.

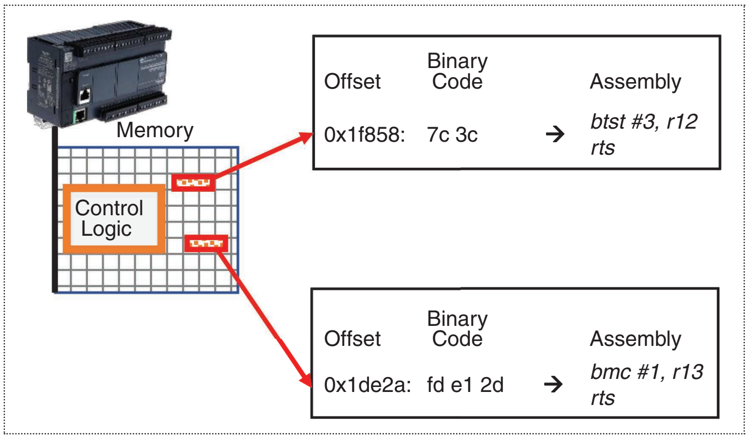


Figure 5. Unlike typical IT, where data execution attacks, such as ROP attacks, are widely known, most PLCs are not designed to defend against them. All gadgets are indexed from memory to exploit the 'return' instruction. This figure shows two gadgets in the memory that can be used to construct a control logic that turns on a light if a certain push button is pressed.

Remote PLC Maintenance

A control engineer uses the engineering workstation to configure and update a PLC configuration remotely, including network and protocol configurations and control logic. Similarly, the attacker can target a control logic and inject malicious code, as discussed in the last section. However, since most PLCs run different control logics, targeting a large number of PLCs requires an automated infection process that takes into account the current control logic in a target PLC. Further, instead of modifying the original control logic, the attacker can be stealthy and download a malicious control logic to a separate PLC memory region often accessible through ICS protocols, such as the I/O data region.

Automated Control Logic Infection

CLIK automates the infection process in four phases,[5] as shown in Figure 6. The first step bypasses PLC authentication to retrieve an original control logic from a PLC. PLCs employ password-based authentication mechanisms that are often weak and prone to subversion.[3] The second step decompiles the control logic (in machine instructions) to a high-level source code in the instruction list, ladder logic, or other IEC 61131-3 languages. In the third step, CLIK uses a rule-based malicious logic generator to add malicious functionality to the original control logic, which is then compiled and transferred to the PLC.

The fourth and final step is to conceal the infection from the control engineer. CLIK utilizes a prebuilt virtual PLC that intercepts the network traffic between the engineering workstation and a target PLC. When a control engineer requests an upload operation from the PLC to read the infected program, the virtual PLC sends the original control logic initially stolen from the PLC.

Data Execution Attack

PLCs do not employ data execution prevention (DEP), allowing attackers to execute any memory block.[4] The attacker exploits it to inject malicious control logic into memory blocks accessible through an ICS protocol. Specifically, the attacker targets the memory regions commonly read and written by ICS services to avoid firewall rules. Considering that the HMI reads I/O data periodically, the attacker transfers a malicious control logic in small chunks to the PLC's data blocks and then redirects the system control flow to execute it (as shown in Figure 7). For instance, the M221 PLC stores the base address of the control logic in the configuration block (shown in Figure 8), which can be modified to run the code in the data block.

Control Logic Decompilation

Engineering (programming) software has a built-in decompiler integrated with the upload functionality to retrieve a control logic from a PLC and then decompile it to one of the IEC 61131-3 languages, such as ladder logic or structured text. The *upload* functionality supports an essential engineering operation to examine and update a PLC's control logic. When an attacker modifies the control logic in a PLC, it also disables the decompilation capability to avoid exposing the malicious control logic. The attacker achieves this by changing the header values and machine instructions of the control logic program.

Denial of Engineering Operations Attack

Denial of engineering operations (DEO) attacks[7] subvert the engineering software from acquiring or uploading an infected control logic in a PLC. The control logic runs successfully in the PLC, but it fails to perform the *upload* function, as shown in Figure 9. For example, in the case of Allen-Bradley's MicroLogix 1400, the attacker can craft and download a malicious control logic on the PLC. This can be done by simply adding an additional rung before the last rung in a ladder logic program at the byte-code level. The addition of one rung changes the content and size of control logic files including the configuration, data files, etc. The addition of one rung at byte code makes a discrepancy between the actual control logic and its metadata (configuration file). Since engineering software relies on the configuration file/metadata to recompile the binary control logic into a higher level representation, it is unable to acquire the control logic from the PLC memory, but it can still run on the PLC. In this way, the attacker can develop a malicious control logic that works on the actual PLC but crashes the engineering software on upload.

Control Logic Obfuscation Attack

This attack[6] obfuscates a control logic code to cause a discrepancy in the engineering software's decompilation process. Specifically, the attacker rewrites a control logic program using machine instructions that are not aligned with the engineering software decompilation. However, since the PLC executes machine instructions, it runs the obfuscated control logic successfully. When a control engineer tries to acquire the obfuscated code using engineering software, the software fails to decompile it. This attack was demonstrated on two PLCs: the Modicon M221 and Siemens S7-300.

Support for ICS Network Protocols

PLCs communicate with the engineering software through request/response messages. The engineering software (the client) sends a request, while the PLC (the server) generates a response. PLCs are required

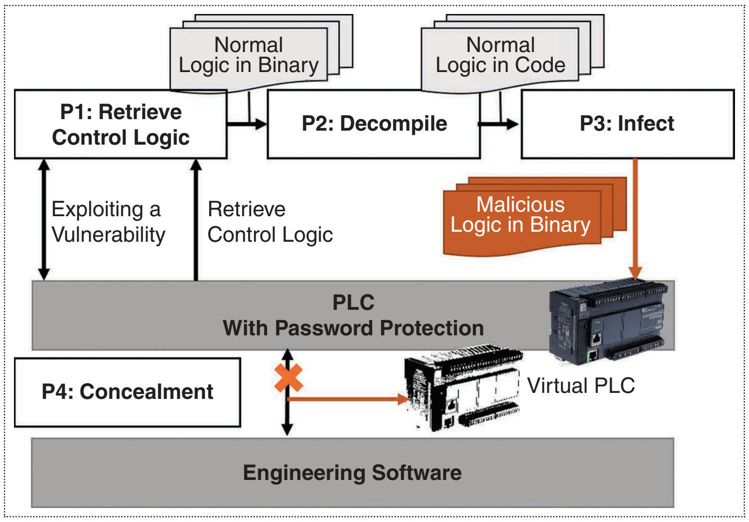


Figure 6. A control logic infection attack consists of four phases. Phase 1 (P1) compromises the PLC security measures and retrieves the control logic. In P2, the stolen (compiled) binaries are decompiled and transferred to P3. P3 injects malicious logic and then transmits the infected binary back to the PLC. P4 hides the PLC's malicious logic from the engineering software using a virtual PLC.

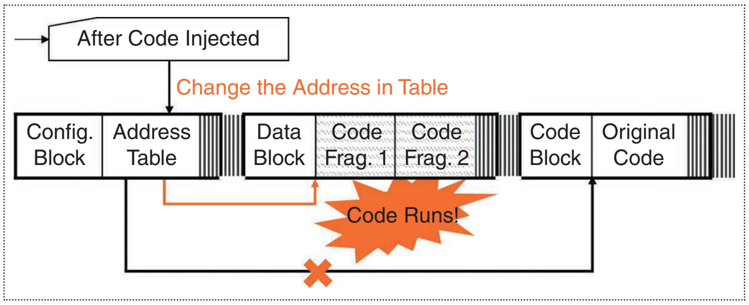
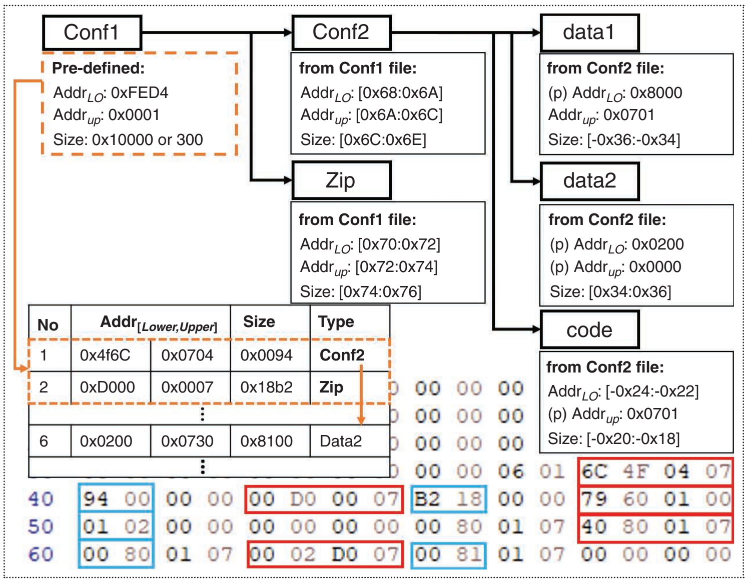


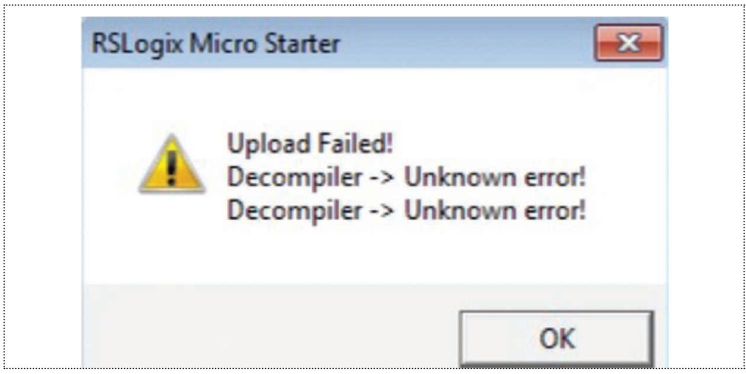
Figure 7. Data execution attack subverts signatures based on packet header fields by redirecting the address table in configuration (metadata) block from the original control logic code to data blocks. Config.: configuration; Frag.: fragment.



No	Addr$_{[Lower,Upper]}$		Size	Type
1	0x4f6C	0x0704	0x0094	Conf2
2	0xD000	0x0007	0x18b2	Zip
⋮				
6	0x0200	0x0730	0x8100	Data2
⋮				

```
                          0 00 00 00
                          0 00 00 00
                          0 00 00 00
                          0 00 00 06 01 6C 4F 04 07
40  94 00 00 00 00 D0 00 07 B2 18 00 00 79 60 01 00
50  01 02 00 00 00 00 00 00 00 80 01 07 40 80 01 07
60  00 80 01 07 00 02 D0 07 00 81 01 07 00 00 00 00
```

Figure 8. Metadata of Schneider Electric's Modicon M221. The upper diagram illustrates the metadata structure of the M221 PLC, while the table in the middle summarizes this information. The binary data (in hexadecimal form) of Configuration 1 (Conf1) is displayed at the bottom, which contains the outlined address (red) and size (blue) information for most metadata areas.


RSLogix Micro Starter

Upload Failed!
Decompiler -> Unknown error!
Decompiler -> Unknown error!

Figure 9. Rockwell Automation's RSLogix Micro Starter pops up the "Upload Failed!" message with an unknown error from the decompiler as a result of a DEO attack.

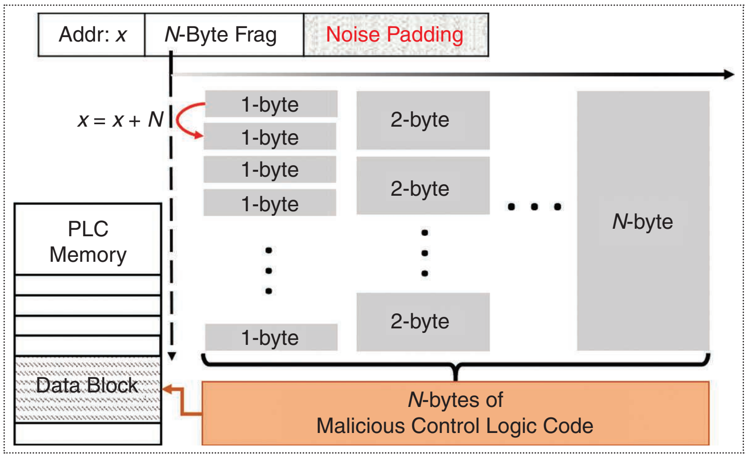


Figure 10. Fragmentation and noise padding attack.

to support various ICS protocols to provide heterogeneous communication and functions.

For example, Allen-Bradley's MicroLogix 1400 supports protocols such as ENIP, DNP3, and Programmable Controller Communication Commands, which is a proprietary protocol. If these protocols are not encrypted, they can be targeted by attackers to initiate communication with the PLC and launch an attack with reverse engineering.[13] The recent attack targets on ICSs (Industroyer 2.0) are also based on a deep understanding of ICS protocols. The following examples further illustrate the manipulation of ICS protocols.

Fragmentation and Noise Padding Attack

In a fragmentation and noise padding attack, an attacker subverts network detection for malicious code by transferring control logic in multiple packets byte by byte. This attack exploits the fact that deep packet inspection techniques cannot detect attack packets that contain significantly small-sized attack payloads because these packets tend to blend in with normal packets.

Figure 10 shows a fragmentation and noise padding attack. In each packet, the attacker sends one byte of control logic followed by noise, which can easily be ignored by network detection tools. To ensure that the target PLC does not use noise and only executes the control logic code, the start location of each write request increments by one instead of incrementing by the size of the last write request. This way, in each write request the previously written noise is overwritten with the next control logic byte. This process goes on until the entire control logic is written on the PLC memory.

Man-in-the-Middle Attack

Man in the middle (MITM) is a well-known threat in the IT as well as the ICS industries. It manipulates the messages in transit by intercepting the traffic between two legitimate parties. The following two attacks make use of the MITM approach to hide an infected control logic from a control engineer who tries to retrieve it via engineering software. An attacker, however, needs to reverse engineer the communication protocol to know the semantics and change the messages in transit so they are accepted by the PLC as well as the engineering software.

1. *DEO 1*: This attack intercepts the communication between the engineering software and the PLC.[7] The attacker first replaces part of the ladder logic being downloaded with infected logic. Then, upon an upload request from the engineering software, the attacker intercepts and manipulates the packets in transit and changes the infected control logic with the legitimate one to make sure the infection remains hidden from the control engineer.

2. *DEO 2*: In this attack, the attacker intercepts the network traffic and replaces part of the ladder logic instructions with noise as it is being transferred from a PLC to programming software in an engineering workstation.[7] This infection basically causes the engineering software to crash.

Device Operation Modes

Most PLCs have different modes, such as program, run, remote, and test modes. These modes represent different states in the PLC, and each mode allows certain PLC operations. For example, in *run* mode, the PLC disables remote memory read/write operations and focuses only on executing the control logic. Similarly, if the control engineer wants to make any changes to the PLC memory, that is, upload or download control logic, the PLC has to be in *program* mode. In *program* mode, the PLC halts the execution of control logic and allows the user to make read/write operations on the PLC memory. Some PLCs also have a *test* mode. This mode is used to test the program execution before allowing the PLC to operate real-world outputs.

"The limitation arising from the PLC design, which prioritizes availability above all else, instills fear of change among users and vendors."

The user can change the mode of a PLC remotely from the engineering software. Upon the user's action, the engineering software sends a command message to the PLC to go to the desired mode. Since these modes can change the state of the PLC and control the execution of control logic as well as the access to PLC memory, they can be exploited by the attacker to sabotage the working of an ICS.

Control Logic Engine Attack

A control logic engine attack[8] is different from typical control logic injection attacks in that it targets the logic engine (responsible for running the control logic) of a PLC by exploiting the PLC modes instead of transferring a malicious control logic. In this attack, the attacker tries to put the PLC in a state where it is unable to execute the control logic written on it. The attacker can capture the network communication between the PLC and the engineering software, identify the messages responsible for the mode change, and replay these messages to the targeted PLC. This is possible because the engineering software sends the command message to change the PLC mode. Network monitoring to notice a mode change is now a well-known path to interrupt communication between PLCs and engineering workstations.[11]

Connection to I/O Devices

As shown in Figure 2, PLC I/O is discrete I/O that in most cases is connected through I/O modules on the PLC. Since I/O is an analog device that generates signals, an ICS uses I/O modules to collect data and transmit them to the PLC based on serial communication protocols for industrial systems. For example, Allen Bradley's Flex I/O and Siemens' ET 200 series are representative I/O modules. The I/O modules use industrial serial protocols to transfer data to the PLC module. These design features make a PLC vulnerable to attacks that attempt to corrupt the I/O module by manipulating its data. For instance, it is possible to insert false (manipulated) I/O data into an injection attack in a fully blind situation.[14] Since this attack manipulates data representing I/O controllers and I/O devices, it impairs the operation of I/O modules and PLCs. Manipulating I/O data successfully shows that these I/O data are not being properly validated.

Discussion

Traditionally, ICSs operated in isolation, providing a secure environment for critical infrastructure. However, with the rise of connectivity to corporate networks and the Internet, these systems have become vulnerable to cyberthreats.[12] To address this, organizations like the Cybersecurity and Infrastructure Security Agency and the ICS Cyber Emergency Response Team are making continuous efforts, such as releasing new advisories.[15]

This section will discuss security requirements for PLCs in ICSs. Some may act independently as countermeasures against specific attacks, but the mutual benefit is more significant when they are applied together. Completely redesigning the PLCs would be one of the solutions, but this approach entails a high cost and potential backward compatibility issues. The limitation arising from the PLC design, which prioritizes availability above all else, instills fear of change among users and vendors.[3]

Although PLC manufacturers somewhat fulfill these security requirements, it is important for them to highlight these measures to end users. For instance, Rockwell Automation's Logix 5000 satisfies some of the requirements outlined next, but it is still necessary to emphasize them as the user may overlook them.

User Management

Typically, there are multiple users who manage PLCs in the field. In modern IT, it's a standard practice to independently *identify and authorize* each user for security purposes. However, PLCs entrust the control center with these security essentials, making them vulnerable if an unauthorized individual gains access to the control center.

Source Verification

The PLC accepts messages from what appears to be a legitimate source without verifying the source. *Source verification* is not applied to both the node and the code, even if the message contains the control logic source code. Engineering software often supports a code verification function, but the PLC rarely verifies the transmitted source code. Suppose all nodes go through an appropriate authentication and the transmitted control logic can be verified; then it is challenging for an attacker to induce malicious actions in the PLC.

Standard Cryptographic Algorithms

Authentication without confidentiality is vulnerable; thus, widely known authentication methods employ encryption. However, PLCs cannot easily apply the application of standard data cryptographic algorithms, such as public-key infrastructure, which requires a trusted certificate authority. Although there may be an initial financial burden for PLC vendors, it is a better investment in the long run compared to the potential damages and risks that may occur from weak authentication methods.

Memory Access Control

Since PLCs are configured and programmed using engineering software, they have the privilege to remotely read and write data on the PLC memory. However, access to the sensitive area in a nonvolatile memory should be verified with access control to secure the control logic of the PLC. Some PLCs even provide access rights to registers corresponding to stack pointers, making it essential to have memory access control that requires a separate procedure to access sensitive memory regions.

DEP

To protect against attacks like CLIK and control logic injection, PLCs must enforce protection schemes like DEP that prevent an attacker from running executable code in the memory. With protection schemes like DEP, an attacker will not be able to execute an injected malicious code. DEP can be implemented in both software and hardware. Software-based DEP is typically a feature of the operating system, while hardware-based DEP requires support from the processor. Implementing DEP in PLCs can be challenging because of resource constraints, limited operating system support, compatibility issues, and increased costs.

Key Management

In the field of key management, it is widely recognized that all keys should be updated periodically as a security measure. Even if users do not utilize the password change function, there should be minimal effort required to convert the internally encoded result. It is imperative that vendors improve the default key management systems to ensure an appropriate level of security.

"It is crucial to carefully consider the tradeoffs between security and system performance when designing and implementing security features in PLCs."

PLCs provide real-time control of changes in a connected physical process and directly control critical infrastructure, making them attractive targets for attackers aiming to sabotage physical processes. Implementing cybersecurity features in PLCs, such as user management, source verification, and cryptographic algorithms, often involves tradeoffs between the primary requirement of real-time control and the need for security. For example, introducing cryptographic methods can potentially impact the real-time performance of the system as both the PLC and the communicating entity will need to encrypt and decrypt messages. The degree of impact will depend on the specific encryption method utilized. Additionally, implementing memory access control and DEP measures can further enhance the security of PLCs. However, these measures can also introduce new challenges to vendors, such as bypassing authentication, which renders the security feature ineffective. The security features in current PLCs are insufficient and often contain many exploitable vulnerabilities due to bad design and poor implementation choices.

Therefore, it is crucial to carefully consider the tradeoffs between security and system performance when designing and implementing security features in PLCs. This article explored the different operational features of various PLCs and further discussed how attackers can exploit them to launch an attack on ICS. By understanding these tradeoffs and potential vulnerabilities/solutions, we can develop more effective security measures to protect critical infrastructure while maintaining the indispensable design requirements. ■

Acknowledgment

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award 17STCIN00001-05-00.

Disclaimer

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

References

1. I. Ahmed et al., "SCADA systems: Challenges for forensic investigators," *Computer*, vol. 45, no. 12, pp. 44–51, Dec. 2012, doi: 10.1109/MC.2012.325.
2. I. Ahmed et al., "Programmable logic controller forensics," *IEEE Security Privacy*, vol. 15, no. 6, pp. 18–24, Nov./Dec. 2017, doi: 10.1109/MSP.2017.4251102.
3. A. Ayub et al., "Empirical study of PLC authentication protocols in industrial control systems," in *Proc. IEEE Secur. Privacy Workshops*, 2021, pp. 383–397, doi: 10.1109/SPW53761.2021.00058.
4. H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *Proc. IFIP Int. Conf. ICT Syst. Secur. Privacy Protection*, Cham, Switzerland: Springer, 2019, pp. 33–48.
5. S. Kalle et al., "CLIK on PLCs! Attacking control logic with decompilation and virtual PLC," in *Proc. Binary Anal. Res. Workshop, Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–12, doi: 10.14722/bar.2019.23074.
6. N. Zubair et al., "Control logic obfuscation attack in industrial control systems," in *Proc. IEEE Int. Conf. Cyber Secur. Resilience*, 2022, pp. 227–232, doi: 10.1109/CSR54599.2022.9850326.
7. S. Senthivel et al., "Denial of engineering operations attacks in industrial control systems," in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy*, 2018, pp. 319–329, doi: 10.1145/3176258.3176319.
8. S. Qasim et al., "Attacking the IEC 61131 logic engine in programmable logic controllers," in *Proc. Int. Conf. Crit. Infrastructure Protection*, Cham, Switzerland: Springer, 2021, pp. 73–95.
9. N. Zubair et al., "PEM: Remote forensic acquisition of PLC memory in industrial control systems," *Forensic Sci. Int., Digit. Investigation*, vol. 40, Apr. 2022, Art. no. 301336, doi: 10.1016/j.fsidi.2022.301336.
10. A. Adeen, Z. Nauman, Y. Hyunguk, J. Wooyeon, and A. Irfan, "Gadgets of gadgets in industrial control systems: Return oriented programming attacks on PLCs," in *Proc. 16th IEEE Int. Symp. Hardware Oriented Secur. Trust*, San Jose, CA, USA, May 2023, pp. 215–226.
11. A. Ghaleb, S. Zhioua, and A. Almulhem, "On PLC network security," *Int. J. Crit. Infrastructure Protection*, vol. 22, pp. 62–69, Sep. 2018, doi: 10.1016/j.ijcip.2018.05.004.
12. X. Qin, K. Mai, N. Ortiz, K. Koneru, and A. A. Cardenas, "Cybersecurity and resilience for the power grid," in *Resilient Control Architectures and Power Systems*, C. Rieger, R. Boring, B. Johnson, and T. McJunkin, Eds. New York, NY, USA: Wiley, 2022, pp. 201–214.
13. S. Senthivel et al., "SCADA network forensics of the PCCC protocol," *Digit. Investigation*, vol. 22, pp. S57–S65, Aug. 2017, doi: 10.1016/j.diin.2017.06.012.
14. W. Alsabbagh and P. Langendörfer, "A fully-blind false data injection on PROFINET I/O systems," in *Proc. IEEE 30th Int. Symp. Ind. Electron.*, 2021, pp. 1–8, doi: 10.1109/ISIE45552.2021.9576496.
15. "CISA releases forty-one industrial control systems advisories," Cybersecurity and Infrastructure Security Agency, Arlington, VA, USA, Dec. 2022. [Online]. Available: https://www.cisa.gov/uscert/ncas/current-activity/2022/12/15/cisa-releases-forty-one-industrial-control-systems-advisories

Adeen Ayub is a Ph.D. student at Virginia Commonwealth University, Richmond, VA 23284-2512 USA. Her research interests include vulnerability discovery and exploit development in industrial control systems. Ayub received a M.S. in cybersecurity from New York University. Contact her at ayuba2@vcu.edu.

Wooyeon Jo is a postdoctoral researcher in computer science at Virginia Commonwealth University, Richmond, VA 23284-2512 USA. His research interests include digital forensics, network security, and cyberphysical systems. Jo received a Ph.D. in computer engineering from Ajou University. He is a Member of IEEE. Contact him at jow@vcu.edu.

Syed Ali Qasim is a Ph.D. candidate at Virginia Commonwealth University, Richmond, VA 23284-2512 USA. His research interests include digital forensics and industrial control systems. Qasim received a B.S. in computer science from Lahore University of Management Sciences, Pakistan. Contact him at qasimsa@vcu.edu.

Irfan Ahmed is an associate professor of computer science at Virginia Commonwealth University, Richmond, VA 23284-2512 USA. His research interests include digital forensics, malware, and cyberphysical systems. Ahmed received a Ph.D. in computer science from Ajou University, South Korea. He is a Senior Member of IEEE. Contact him at iahmed3@vcu.edu.