Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.041774

ARTICLE

Programmable Logic Controller Block Monitoring System for Memory Attack Defense in Industrial Control Systems

Mingyu Lee[1], Jiho Shin[2] and Jung Taek Seo[3,*]

[1]Department of Information Security, Gachon University, Seongnam, 13120, Korea

[2]Police Science Institute, Korean National Police University, Asan, 31539, Korea

[3]Department of Computer Engineering, Gachon University, Seongnam, 13120, Korea

*Corresponding Author: Jung Taek Seo. Email: seojt@gachon.ac.kr

Received: 05 May 2023 Accepted: 05 September 2023 Published: 29 November 2023

ABSTRACT

Cyberattacks targeting industrial control systems (ICS) are becoming more sophisticated and advanced than in the past. A programmable logic controller (PLC), a core component of ICS, controls and monitors sensors and actuators in the field. However, PLC has memory attack threats such as program injection and manipulation, which has long been a major target for attackers, and it is important to detect these attacks for ICS security. To detect PLC memory attacks, a security system is required to acquire and monitor PLC memory directly. In addition, the performance impact of the security system on the PLC makes it difficult to apply to the ICS. To address these challenges, this paper proposes a system to detect PLC memory attacks by continuously acquiring and monitoring PLC memory. The proposed system detects PLC memory attacks by acquiring the program blocks and block information directly from the same layer as the PLC and then comparing them in bytes with previous data. Experiments with Siemens S7-300 and S7-400 PLC were conducted to evaluate the PLC memory detection performance and performance impact on PLC. The experimental results demonstrate that the proposed system detects all malicious organization block (OB) injection and data block (DB) manipulation, and the increment of PLC cycle time, the impact on PLC performance, was less than 1 ms. The proposed system detects PLC memory attacks with a simpler detection method than earlier studies. Furthermore, the proposed system can be applied to ICS with a small performance impact on PLC.

KEYWORDS

Programmable logic controller; industrial control system; attack detection

1 Introduction

ICS controls and monitors infrastructures such as transportation, power plant, water treatment, factory. These environments are closed operational technology (OT) unlike information technology. Also, OT has high availability and real-time requirements, so it is hard to apply security solutions [1,2]. With the development of the Internet, ICS is under constant threat of cyberattacks. Starting with Stuxnet in 2010 [3], cyberattacks targeting ICS such as TRITON, BlackEnergy, and Havex are increasing day by day, becoming more sophisticated [4]. A programmable logic controller, a core component of ICS, controls and monitors a number of field devices on ICS. Hence, attackers remotely

inject and manipulate the PLC program. These attacks not only cause economic losses but also lead to casualties. Therefore, research is needed on how to detect cyberattacks that inject and manipulate malicious PLC program while meeting the real-time requirements of PLC.

Recently, security systems using PLC program have been proposed to detect attacks on PLC. Reference [5] proposed security blocks of PLC program to detect memory read/write logic attack and malware worm attack. Reference [6] proposed a control program logic change detector to detect control program attack and memory read/write logic attack.

Most studies to detect PLC program injection or manipulation are conducted on the engineering workstation (EWS) or its higher level. These studies are vulnerable to attacks that manipulate the operator view, such as Stuxnet [7]. In addition, manual process like setting detection rules and implementing security blocks is inefficient than automated process. Finally, in order to meet the high availability and real-time requirements of PLC, a performance impact on PLC measurement is required after security system application.

To solve these challenges, the security system should directly acquire and monitor PLC memory. It is also necessary to measure the performance impact of the security system on the PLC to ensure high availability and real-time requirements of the PLC. In this paper, we propose a PLC program block monitoring system to detect PLC memory attacks by continuously acquiring and monitoring program. This system consists of memory acquisition module, monitoring module, and logging module. This system eventually detects PLC program injection or manipulation by monitoring memory. Also, the proposed system has a very low performance impact on PLC.

The contributions of this study, which proposes a PLC program block monitoring system for security in ICS, are as follows:

- We propose a PLC block monitoring system against memory attacks on programmable logic controllers in industrial control systems.
- The proposed security system detects PLC memory attacks simpler and more efficiently than existing studies with monitoring algorithm.
- The proposed security system is located at level 1 but meets the high availability and real-time requirements of the PLC.

The remainder of this paper is structured as follows. Section 2 deals with background knowledge and research related to PLC security. Section 3 describes the overview, system architecture, and monitoring algorithm of the proposed PLC block monitoring system. Section 4 presents the results of the attack detection test using the PLC block monitoring system and the test results of measuring the performance impact. Section 5 offers the limitations of this study, and finally, Section 6 contains a conclusion and future work.

2 Background and Related Work

2.1 Industrial Control Systems

ICS is a system that monitors and controls devices installed in the field across different industries, such as energy, transportation, water treatment facilities, and factories. Because an ICS operates in an OT environment, it is essential not to compromise productivity and availability. Therefore, the violation of availability attributed to security applications is another challenge of ICS security.

A typical network architecture for ICS security is the Purdue reference model [8]. The Purdue model is a reference architecture for ICS that helps organizations to design and implement secure

and reliable control networks. The model divides the ICS environment into seven levels with specific functions and security requirements. The Purdue reference model consists of 7 levels, an OT environment from level 0 to level 3 and an IT environment from level 4 to level 6. Level 0 (the field devices zone) contains field devices such as sensors, actuators, and motors operating in the field. It is the lowest layer in the model and is responsible for performing the actual work of the industrial process. Level 1 (the controller zone) includes PLCs, remote terminal units (RTUs), and intelligent electronic devices (IEDs) that control field equipment and collect measurement values such as temperatures and speed. Level 2 (the control system zone) includes HMIs, which the operators use to interact with the controller, and supervisory control and data acquisition (SCADA). Finally, level 3 (the manufacturing operations system zone) includes an EWS, which communicates with level 1 controllers to collect data or download the control logic, and a historian, which stores the collected information. ICS also integrates various hardware and software components, such as sensors, controllers, and communication networks, which work together to facilitate real-time monitoring and control of industrial processes. The PLC block monitoring system proposed in this study is located at level 1 and monitors the PLC program blocks with guaranteed integrity by directly communicating with the PLC.

2.2 PLC Programming

The PLC targets the field devices of level 0 to control their operations or collect data from them. Because of these characteristics, it is considered an important component of the ICS. The PLC operates based on the created PLC program [9], which consists of the following four main blocks (Fig. 1):

- Organization Block (OB)
- Function (FC)
- Function Block (FB)
- Data Block (DB)

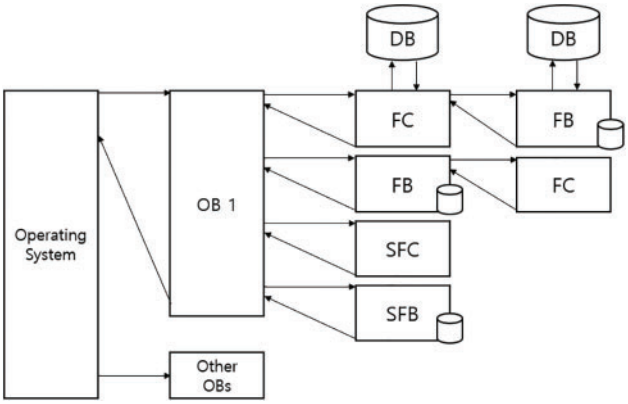

Figure 1: Correlation between PLC program blocks

OB is a block that determines the structure of the PLC program and serves as an interface between the PLC operating system and the user program. In the PLC program, OB1 is a structure that includes other blocks because it communicates with the PLC operating system (OS) and acts as the main function of the PLC program. The remaining OBs have different functions depending on the numbers assigned to them. FC provides certain functions of the program. FB is similar to an FC but has a specific memory area as an instance data block. DB is a block that stores user data, and it is divided

into a global DB shared by all blocks and a local DB that can only be used in a specific block. In this study, OB1, which serves as the main function of the PLC program, and the DB, which contains user data, were collected from the PLC. Subsequently, the block information of OB1 and DB and OB1 byte array are separately monitored in bytes, and tampering is detected by comparing past and present values.

2.3 Related Work

To demonstrate the vulnerability of PLCs, studies have been conducted to attack them by injecting malicious blocks or sending malicious packets. Reference [10] presented a four-step remote attack Control Logic Injection Attack (CLIK) targeting PLCs. This attack remotely infects the PLC and hides the infection with the EWS. Therefore, PLC security techniques operating at the level of the EWS have difficulty detecting such an attack. Reference [11] used fragmentation and noise padding to bypass signature-based detection and deep packet inspection (DPI) for the stealthy transmission of malicious packets. Compared to previous studies that used signature-based detection and DPI, our study detects attacks by comparing the previous and present values of the PLC block, so it cannot be bypassed by fragmentation and noise-padding techniques. Reference [12] discussed attacks that bypass traditional detection methods to target Siemens' S7 PLCs with little traffic and short times. Such an attack can be detected because the byte array of the OB1 or the block information of the DB change in the course of writing to the PLC. Reference [13] proposed a PLC attack using a malicious OB10 (time-of-day interrupt) and implemented an attack on a Siemens' S7-300 PLC. The monitoring system of our study can detect OB10 injection through a byte array change in OB1.

As many papers have been published demonstrating PLC vulnerabilities, several security-related studies have been conducted to detect cyberattacks on PLCs. Reference [5] proposed an approach to adding monitoring and logging mechanisms for security to a PLC. A security block implemented as an OB performs critical information monitoring, system integrity monitoring, event logging, and rule-based detection through an intrusion detection system (IDS)/rule checker. However, the proposed solution costs to implement security blocks and the larger the control logic, the higher the cost and performance impact. Reference [14] reviewed relevant studies on code security, network security, and other PLC security issues. Code security research, formal analysis, is a model detection technique that analyzes the PLC program syntax, converts it into an intermediate language, and uses a detection model to check for code defects. When this technique is applied to an extensive PLC program, it leads to the issue of excessive consumption of time and memory due to the use of symbol execution. Network security includes a security monitoring system and PLC memory change detection technique. The first method uses network mirroring and agents to collect traffic and logs and monitor remote connection services, unauthorized processes/ports, and open platform communications (OPC) read/write commands. The technique is based on the reliability of SCADA systems, such as EWS and HMI, and if the system is compromised, the data can be damaged in the middle. Reference [15] introduced a shade to prevent remote control logic injection attacks on PLCs. This technology detects the write request in ICS network traffic and mirrors it to shadow memory to check the maximum length of the decompiled byte sequence, number of rungs, number of opcodes, number of n-grams, and byte entropy of the scan area. This technology has the limitation of relying on write requests from network traffic to detect PLC attacks. Reference [16] studied the applicability of countermeasures against memory attacks in ICS environments. Based on the secure water treatment plant testbed, they measure the memory safety overhead due to the memory-safe compilation of the PLC and quantify the tolerability of the overhead in terms of the real-time constraints of the ICS. Reference [17] proposed PLCprint, which uses PLC memory artifacts to detect and classify memory

attacks. The proposed methodology conducted detection experiments on Siemens S7-300 PLC and Allen-Bradley ControlLogix PLC at Glasgow University liquid purification testbed, and evaluated attack detection performance through one class support vector machine and k-nearest neighborhood detector. However, this study relies on existing PLC artifacts in detecting memory attacks, and securing storage capacity is important because numerous PLC artifacts are stored in the database. Reference [18] provided a programmable logic controller variable block scanner, a tool for identifying vulnerabilities in PLCs. The PLC-VBS scanner is tested to be able to identify vulnerable bits and bytes within PLC memory and to allow attackers to interact with the device. However, PLC-VBS requires a predefined set of signatures for vulnerable bits and bytes. Therefore, it is difficult to detect memory attacks caused by weak bits and bytes that are not in the signature set. Reference [6] proposed a control program logic change detector to detect abnormal operations of PLC. This system detected the two types of attacks against PLC based on detection rules (DRs). This study took an approach similar to the present one. Still, there is a problem: the DRs must be defined for rungs existing in multiple PLC programs and individually updated when an operator intentionally changes the PLC program.

- Control program attack: An attack that replaces the PLC program in an EWS and then downloads it to the PLC.
- Memory read and write logic attack: An attack that changes the value of a memory address by sending a read/write command directly to a PLC.

Therefore, this study proposes and implements a monitoring system that monitors block information as byte array data to detect attacks by reading OB1 and DB. The system is simple because it only detects changes in the previous and present values, not whitelist-based changes. When any change in values occurs, the system logs in and simultaneously sends an alarm to an operator so that they may determine whether the manipulation is intentional or malicious.

3 Proposed Method for Detecting and Monitoring Program Blocks on PLC

3.1 Overview

This study proposes a monitoring system that monitors PLC program blocks and detects manipulation. The monitoring system is connected to the level 1 switch (L1 switch), and the data to be monitored is as follows. Block information includes block type, block number, block size, load memory size, local data, checksum, version, and code date (Table 1).

Table 1: PLC block information

Block information	Description
Block type	The type of the block, such as “OB” for an organization block or “DB” for a data block
Block number	The number of the block within its type
Block size	The size of the block in bytes
Load memory size	The amount of memory required to load the block into the PLC’s memory
Local data	The amount of data that is stored locally in the block
Checksum	A value used to ensure the integrity of the block data
Version	The version of the block
Code date	The date that the block was last modified

The PLC program block has many blocks, including OB1. The proposed method monitors byte array and block information for OB1 such as main function. OB1 includes all information except dynamic data that changes in real-time. Next, block information from DBs is utilized to monitor dynamic information. Because an attacker modulates the structure of the program or specific data when attacking the PLC program, efficient monitoring is possible by monitoring both OB1 representing the entire structure and DBs storing data.

The architecture of the overall system for PLC program block monitoring is illustrated in Fig. 2. The system is divided into three modules: a PLC memory acquisition module for collecting PLC memory, a monitoring module for monitoring blocks through memory comparison, and a logging module for storing time, block information, and data when tampering is detected. Fig. 3 shows the process flow of the system. The system is connected to a level 1 switch of the Purdue model of PLC memory acquisition; the PLC performance problems that can arise from this situation are discussed in Section 4. The system is directly connected to the PLC, not the EWS. The shorter the network distance between the monitoring system and the PLC becomes, the more data integrity is guaranteed. For example, in the case of Stuxnet, an attack was carried out in which a normal screen was output to the EWS by infecting s7otbxdx.dll in an EWS located at level 3. If PLC data is collected through the EWS, normal PLC data before the attack is included. Therefore, this system is connected to the level 1 switch to collect the actual OB and DB from the PLC to monitor and detect the attack.

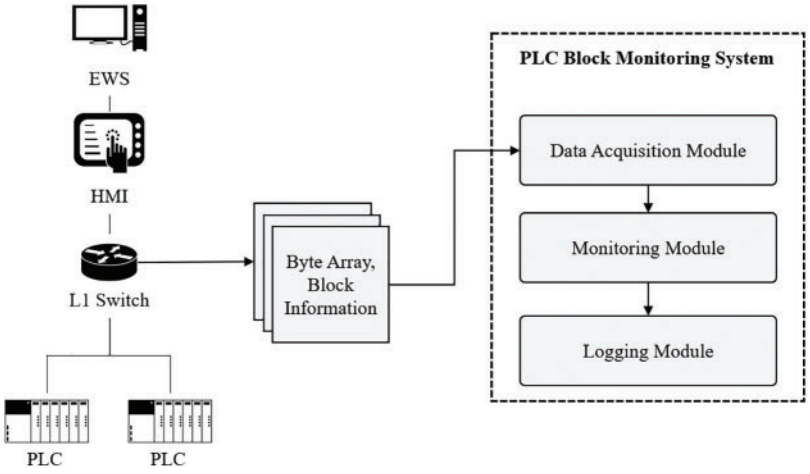

Figure 2: PLC block monitoring system architecture

3.2 Memory Acquisition

The memory acquisition module of the proposed system collects data to be monitored from the PLC. The proposed method collects byte array and block information in OB1 and DBs. The byte array is all data, including the block header, body, and footer. The byte array collected by the memory acquisition module is the byte array of OB1, which represents the overall structure of the PLC program. The block information includes a block type, number, size, and checksum. There are three PLC memory collection methods: built-in support, the debug port, and the ICS communication protocol [19]. The proposed system adopts the ICS communication protocol method because it collects data from many PLCs rather than forensics which collects data from a single PLC.

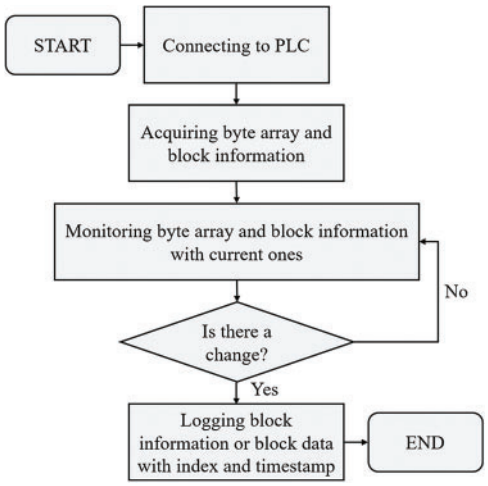

Figure 3: Process flow of PLC block monitoring system

3.3 Monitoring

The monitoring module of the proposed system monitors the byte array and block information of OB1 and block information of DBs obtained from the memory acquisition module. There are many monitoring methods, and they are determined by the characteristics of the data. The byte array of OB1 and block information of OB1 and DBs collected in the memory acquisition module are static data, not data that changes dynamically under the influence of field devices. Therefore, this monitoring system uses a simple and robust method of detecting changes by comparing previous and current values for periodically collected data.

Algorithm 1 describes the monitoring algorithm to detect memory manipulation of the PLC block. $\mathcal{I}$ is the identifier of the monitored PLC block, and in the case of OB1, $\mathcal{I}$ is 1. $\mathcal{L}$ is the byte array and block information of the PLC block. When $\mathcal{L}$ is acquired through the memory acquisition module, the length of $\mathcal{L}$ is stored as l. $\mathcal{L}'$ is the current byte array and block information that is continuously collected by the memory acquisition module. The monitoring module compares $\mathcal{L}[i]$ with $\mathcal{L}'[i]$, which repeats l times. If $\mathcal{L}[i]$ and $\mathcal{L}'[i]$ are different due to PLC memory attack, the comparison is over and an index is recorded or a security alarm occurs. Algorithm 1 is repeated until a PLC memory attack is detected or monitoring is stopped. As l increases, Algorithm 1 takes the same amount of time to compare. Therefore, expressing the time complexity of Algorithm 1 in Big O notation is O(n).

Algorithm 1: Monitoring algorithm to detect manipulation of the PLC block.

Input:
$\mathcal{I}$: the identifier of the PLC block to be monitored.
$\mathcal{L}$: the byte array or block information of the PLC block($\mathcal{I}$).

1. $l \leftarrow$ get the length of $\mathcal{L}$;
2. **Repeat**
3. $\mathcal{L}' \leftarrow$ get a current byte array or block information of the PLC block($\mathcal{I}$);
4. **if** $\mathcal{L} \neq \mathcal{L}'$ **then**
5. **for** $i \leftarrow 1$ to l **do**

(Continued)

Algorithm 1 (continued)

```
6.          if L[i] ≠ L'[i] then
7.              record logs or alert the security alarm with index i;
8.              break;
9.          end if
10.      end for
11.   end if
12.  until stopping monitoring
```

3.4 Logging

The logging module of the proposed system operates if modulation to the PLC byte array or block information is detected during monitoring. The logging module helps the operator take action by recording timestamps, changed bytes, and indexes for changed bytes, or by alerting security alarms. The operator is provided with logs or security alarms that can be used to distinguish between normal behavior and PLC memory attacks.

4 Experiments

The experimental environment shown in Table 2 was established to test the PLC block monitoring system proposed and implemented in this study. EWS is Siemens' engineering workstation, the TIA Portal V16, for PLC programming, virtual PLC, PLC operating mode setting, and cycle time measurements. PLCs are Siemens' S7-300 and S7-400 PLC, which account for the largest percentage of the ICS market implemented through the PLCSIM functionality of the TIA Portal V16 [20]. Since two PLCs use the S7Comm protocol [21], it implements communication using the python-snap7 library. A program in Fig. 4 for performing a timer function is downloaded to the S7-300 and S7-400 PLC. Network extension is a NetToPLCsim program that supports TCP/IP interfaces for communication between virtual PLC and monitoring systems implemented through PLCSIM. The monitoring system implements the PLC program block monitoring method proposed in this study and is a script implemented in Python. A representative library of the implemented system is python-snap7. The library is a platform that provides an Ethernet communication suite for interfacing with Siemens' S7 PLCs. To implement the acquisition module of the monitoring system, the list_blocks(), get_block_info(), and full_upload() functions of the python-snap7 library were used, and the description of each function is in Table 3. Finally, the PC executes all of the above components. In the above experimental environment (Table 2), we conducted two experiments. The first experiment was a PLC program manipulation detection experiment, which verified whether the proposed system could detect attacks when modulating the PLC program using the python-snap7 library. The second experiment measured the performance impact of PLC due to the monitoring system.

Table 2: Experimental environment

Component	Program	Description
EWS	TIA Portal V16	Siemens' Engineering Workstation with features such as PLC programming, virtual PLC, PLC mode settings, and cycle time measurements
PLC	PLCSIM (S7-300, S7-400)	TIA Portal's PLC simulator, which implements a virtual S7 PLC
Network extension	NetToPLCsim	Supporting TCP/IP communications between PCs when PLCSIM and a simulator are running
Monitoring system	Python script	A system that implements the PLC program block monitoring method presented in this study with Python and monitors S7-300 and S7-400 PLC
PC	CPU: 11th i7-11700 RAM: 32 G GPU: GeForce RTX 3060 OS: Windows 11	A PC that runs the EWS, PLC, network extension, and monitoring system

4.1 Detecting Memory Manipulation in the PLC

Cyberattacks on PLC include code injection in which an attacker injects malicious code, memory manipulation that partially modifies or changes the original program, DoS attack that depletes the system resources, firmware manipulation that modulates the PLC firmware, and Zero-Exploit. Code injection and memory manipulation appear in advanced persistent threat (APT) attacks in ICS such as Stuxnet and studies [10–13, 22–26]. In this paper, to evaluate the detection capability of the proposed system, we experimented with whether malicious OB inflection and DB manipulation can be detected.

In this experiment, we evaluated whether the PLC block monitoring system that monitors the OB1 byte array, OB1 information, and DB information of the PLC can detect PLC memory manipulation. The attacker PC performed a PLC memory attack (Table 4) that inserted and manipulated the PLC program using the download() and db_fill() functions of the python-snap7 library (Table 5). The monitoring system executed Algorithm 1 every 0.15 s.

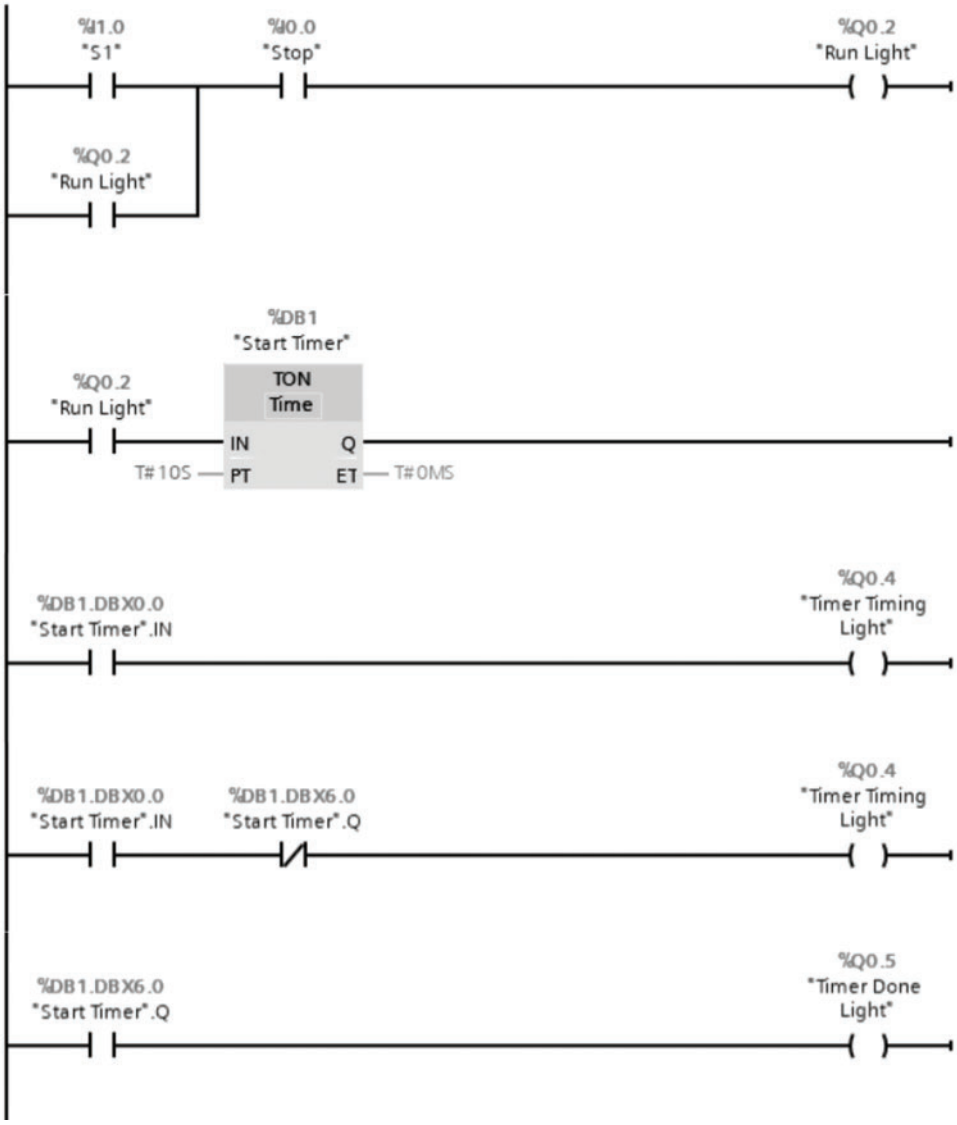

Figure 4: PLC program on the S7-300/400 PLC

Table 3: Functions of python-snap7 used in the PLC block monitoring system

Function	Description
list_blocks()	Returns the number by the program block.
get_block_info()	Returns the block information for the specified block.
full_upload()	Uploads an entire block body. The whole block (including the header and footer) is copied into the user buffer.

Table 4: PLC memory attack

Attack type	Description
Malicious OB injection	Malicious OB injection using python-snap7's download()
DB manipulation	DB manipulation using python-snap7's db_fill()

Table 5: Functions of python-snap7 used in PLC memory attacks

Function	Description
download()	Download byte array of OB to PLC
db_fill()	Fills a DB in PLC with a given byte

4.1.1 Malicious OB Injection

Malicious OB injection attacks are attacks that insert malicious OB produced by attackers into normal PLCs, which were used in Stuxnet and are also used in various studies attacking PLCs. A byte array of normal OB1 was collected, and some bytes were manipulated to valid values and then inserted into the PLC using download() of python-snap7. The PLC block monitoring system detected malicious OB1 immediately after insertion and returned the timestamp of the detected time and the index of the manipulated byte.

4.1.2 DB Manipulation

DB manipulation attacks are attacks that manipulate the values of variables present in the DB of PLCs and are used in various studies attacking PLCs, as is malicious OB injection. The value of the variable was modified based on the byte array of normal DB1 and manipulated using db_fill(). The PLC block monitoring system immediately detected DB1 manipulation and returned the timestamp.

4.2 Measuring the Performance Impact of PLC

PLC cycle time refers to the time it takes for the PLC to fully execute the program and update all I/O and is an important measure of the performance of the PLC. The PLC also has real-time constraints that ensure each cycle time does not exceed a predefined period. Consequently, earlier studies have measured the impact of security systems on PLC performance like cycle time [2,5,16,27,28]. The PLC block monitoring system may also affect PLC performance because it communicates directly with PLC at level 1 and continuously sends requests using the functions of python-snap7. The shortest, current, and longest cycle times were measured through the 'online and diagnostic' feature of TIA Portal V16, an EWS software. The shortest cycle time represents the shortest amount of time the PLC has taken to execute the program. It is often an indicator of the maximum processing capability of the PLC. The current cycle time represents the amount of time the PLC took to execute the program in the last cycle. It is an indicator of the current performance state of the PLC. The longest cycle time represents the longest amount of time the PLC has taken to execute the program. It is useful for identifying bottlenecks in the system or detecting abnormal situations.

In this experiment, we evaluate the impact by the application of the PLC block monitoring system and the impact by the request interval.

4.2.1 Measuring PLC Performance Impact by Application of Proposed System

The PLC block monitoring system may cause overhead because it is connected to level 1 where PLCs are located and communicates directly with PLCs. Table 6 shows how much the introduction of the PLC block monitoring system affects the performance of the PLC. Table 6 suggests that the PLC performance impact from the proposed system application is less than 1 ms because the 'online and diagnostic' feature of TIA Portal V16 measures cycle time in ms.

- Test case 1: This test case shows PLC cycle times before applying the PLC block monitoring system.
- Test case 2: This test case shows PLC cycle times after applying the PLC block monitoring system.

Table 6: PLC cycle time introduced by the PLC block monitoring system

Test case	Shortest cycle time	Current cycle time	Longest cycle time
1	10 ms	10 ms	54 ms
2	10 ms	10 ms	54 ms

4.2.2 Measuring PLC Performance Impact by Request Interval

The PLC block monitoring system continuously requests the PLC to collect data such as block information and byte arrays. Requests to the PLC, such as functions in Table 3, can cause additional data processing by the PLC, which can affect the cycle time of the PLC. Table 7 shows how much the request interval affects the performance of the PLC.

- Test case 1: This test case shows PLC cycle times when the request interval is set to 0.5 s.
- Test case 2: This test case shows PLC cycle times when the request interval is set to 0.3 s.
- Test case 3: This test case shows PLC cycle times when the request interval is set to 0.15 s.

Table 7: PLC cycle time by request interval

Test case	Shortest cycle time	Current cycle time	Longest cycle time
1	10 ms	10 ms	54 ms
2	10 ms	10 ms	54 ms
3	10 ms	10 ms	54 ms

4.3 Performance Comparison with Conventional Research

Many studies have been conducted to detect and protect against PLC memory attacks. We consider three aspects of the performance of the proposed PLC block monitoring system and its operating position, whether the PLC memory attack can be protected, and its performance impact for comparison with the system presented in previous studies (Table 8). The operating position is the level of the Purdue model in which the system is installed for PLC protection. It mainly protects the PLC directly at level 1, where the PLC is located, or at level 3, where the EWS that manages the PLC is located. Whether PLC memory attacks can be protected indicates whether the proposed system can protect memory attacks against PLCs. The performance impact is derived from cycle

time measurement experiments as a performance impact on PLC by applying a system for protecting PLC. Our proposed method directly protects the PLC and is located at level 1, which is immune to manipulation of view attacks. In addition, PLC memory attacks can be protected based on the first experiment, and significant results in performance impact are based on the second experiment. In conclusion, it detects PLC memory attacks in a simple and powerful manner compared to existing studies and has less impact on performance.

Table 8: Comparative analysis of the proposed method with conventional research

Reference	Operating position (Purdue)	Protecting 'PLC memory attack'	Performance impact
Chan et al. [5]	Level 1	Partial	2 ms≤
Chekole et al. [16]	Level 1	Yes	≤4.6 ms
Cook et al. [17]	Level 1	Partial	200 ms≤
Maesschalck et al. [18]	Level 2	Partial	–
Proposed system	Level 1	Yes	≤1 ms

5 Limitations

This study proposed and implemented a PLC block monitoring system for the security of ICS. The system uses the simple but powerful detection method using OB and DB of PLC. However, there are some challenges to overcome:

- Limited PLC types. This system uses the python-snap7 library for memory acquisition. This library implements Siemens S7Comm protocol and cannot apply the system to other vendors' PLCs.
- Lack of PLC programs. It is necessary to experiment with numerous different PLC programs to measure the precise performance impact.
- Lack of decompile functions. Providing decompiled results for manipulated byte array values would improve visibility. Currently, only tampered values and indices can be identified.
- Limitations of data block monitoring. In the case of the DB, since various values continuously change while the PLC is operating, tampering cannot be detected through previous–present value comparison. Therefore, the system monitors only block information, not the block data of DBs.

6 Conclusion

In this study, we proposed a system to detect memory manipulation by monitoring OB1 and DB in PLC program blocks to protect ICS that have become more vulnerable with the introduction of industrial IoT. The monitoring system was applied to level 1 (the controller zone) to monitor a reliable PLC program by directly communicating with the PLC. This makes it possible to detect attacks on the PLC efficiently. In addition, OB1, including the program's main function and DB that stores user data, were monitored for efficient detection. It was found that PLC memory manipulation could be detected, and the very minute performance impact caused by the application of the monitoring system was confirmed.

In follow-up research, we intend to perform experiments inquiring whether tampering detection can be achieved by applying a monitoring system to various vendor products and multiple PLC programs. In addition, by adding the decompile functions [29], we hope to provide decompiled values instead of returning only an index or byte array for the tampered part. Finally, we desire to study how the ever-changing data block can be monitored.

Acknowledgement: Thanks to our researchers and the reviewers which have improved this paper.

Funding Statement: This research was supported by the Korea WESTERN POWER (KOWEPO) (2022-Commissioned Research-11, Development of Cyberattack Detection Technology for New and Renewable Energy Control System Using AI (Artificial Intelligence), 50%) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-01806, Development of Security by Design and Security Management Technology in Smart Factory, 40%) and the Gachon University Research Fund of 2023 (GCU-202110280001, 10%).

Author Contributions: Study conception and design: M. Lee, J. Seo, J. Shin; method: M. Lee; implementation and experiments: M. Lee; draft manuscript preparation: M. Lee, J. Shin; draft review: J. Seo. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and materials in the experiments can be produced from the corresponding author on request.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] Z. Drias, A. Serhrouchni and O. Vogel, "Analysis of cyber security for industrial control systems," in *Proc. of 2015 Int. Conf. on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)*, Shanghai, China, pp. 1–8, 2015.

[2] Y. Chang, T. Kim and W. Kim, "Impact analysis of PLC performance when applying cyber security solutions using active information gathering," in *Proc. of 16th Int. Conf. on Critical Information Infrastructures Security (CRITIS 2021)*, Lausanne, Switzerland, pp. 131–151, 2021.

[3] M. A. Z. Raja, H. Naz, M. Shoaib and A. Mehmood, "Design of backpropagated neurocomputing paradigm for Stuxnet virus dynamics in control infrastructure," *Neural Computing and Applications*, vol. 34, pp. 5771–5790, 2022.

[4] R. Kumar, R. Kela, S. Singh and R. Trujillo-Rasua, "APT attacks on industrial control systems: A tale of three incidents," *International Journal of Critical Infrastructure Protection*, vol. 37, no. 1, pp. 100521, 2022.

[5] C. Chan, K. Chow, S. You and K. Yau, "Enhancing the security and forensic capabilities of programmable logic controllers," in *Proc. of 14th IFIP Int. Conf. on Digital Forensics*, New Delhi, India, pp. 351–367, 2018.

[6] K. Yau and K. Chow, "PLC forensics based on control program logic change detection," *Journal of Digital Forensics, Security and Law*, vol. 10, no. 4, pp. 5, 2015.

[7] S. Al-Rabiaah, "The "Stuxnet" virus of 2010 as an example of a "APT" and its "recent" variances," in *Proc. of 2018 21st Saudi Computer Society National Computer Conf. (NCC)*, Riyadh, Saudi Arabia, pp. 1–5, 2018.

[8] B. Siemers, L. Fischer and S. Lehnhoff, "A trust model in control systems to enhance and support cybersecurity," in *Proc. of IEEE 7th Int. Energy Conf. (ENERGYCON)*, Riga, Latvia, pp. 1–6, 2022.

[9] W. Alsabbagh and P. Langendörfer, "A mew injection threat on S7-1500 PLCs-disrupting the physical process offline," *IEEE Open Journal of the Industrial Electronics Society*, vol. 3, pp. 146–162, 2022.
[10] S. Kalle, N. Ameen, H. Yoo and I. Ahmed, "CLIK on PLCs! attacking control logic with decompilation and virtual PLC," in *Proc. of 2019 Workshop on Binary Analysis Research (BAR)*, San Diego, CA, USA, 2019.
[11] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *Proc. of 34th Int. Conf. on Information Security and Privacy Protection (IFIP SEC 2019)*, Lisbon, Portugal, pp. 33–48, 2019.
[12] Y. Wang, J. Liu, C. Yang, L. Zhou, S. Li *et al.,* "Access control attacks on PLC vulnerabilities," *Journal of Computer and Communications*, vol. 6, no. 11, pp. 311–325, 2018.
[13] W. Alsabbagh and P. Langendörfer, "Patch now and attack later—Exploiting S7 PLCs by time-of-day block," in *Proc. of 4th IEEE Int. Conf. on Industrial Cyber-Physical Systems (ICPS)*, Victoria, BC, Canada, pp. 144–151, 2021.
[14] X. Pan, Z. Wang and Y. Sun, "Review of PLC security issues in industrial control system," *Journal of Cyber Security*, vol. 2, no. 2, pp. 69–83, 2020.
[15] H. Yoo, S. Kalle, J. Smith and I. Ahmed, "Overshadow PLC to detect remote control-logic injection attacks," in *Proc. of 16th Int. Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, Gothenburg, Sweden, pp. 109–132, 2019.
[16] E. G. Chekole, J. H. Castellanos, M. Ochoa and D. K. Y. Yau, "Enforcing memory safety in cyber-physical systems," in *Proc. of ESORICS 2017 Int. Workshops, CyberICPS 2017 and SECPRE 2017*, Oslo, Norway, pp. 127–144, 2017.
[17] M. M. Cook, A. K. Marnerides and D. Pezaros, "PLCPrint: Fingerprinting memory attacks in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3376–3387, 2023.
[18] S. Maesschalck, A. Staves, R. Derbyshire, B. Green and D. Hutchison, "Walking under the ladder logic: PLC-VBS: A PLC control logic vulnerability scanning tool," *Computers & Security*, vol. 127, no. 1, pp. 103116, 2023.
[19] N. Zubair, A. Ayub, H. Yoo and I. Ahmed, "PEM: Remote forensic acquisition of PLC memory in industrial control systems," *Forensic Science International: Digital Investigation*, vol. 40, pp. 1–10, 2022.
[20] A. Robles-Durazno, N. Moradpoor, J. McWhinnie, G. Russell and I. Maneru-Marin, "PLC memory attack detection and response in a clean water supply system," *International Journal of Critical Infrastructure Protection*, vol. 26, no. 1, pp. 100300, 2019.
[21] Q. Shen, L. Wang, L. Zhang, B. Wang, C. Liu *et al.,* "Security analysis of industrial control S7 protocol based on peach," in *Proc. of 9th Int. Conf. on Computing and Data Engineering (ICCDE'23)*, New York, USA, pp. 72–77, 2023.
[22] A. Qasim, A. Ayub, J. Johnson and I. Ahmed, "Attacking the IEC 61131 logic engine in programmable logic controllers," in *Proc. of Int. Conf. on Critical Infrastructure Protection (ICCIP 2021)*, pp. 73–95, 2021.
[23] Y. Geng, K. Liu, R. Ma and Q. Wei, "Research on memory attacks and defenses for programmable logic controllers," in *Proc. of 2022 4th Int. Conf. on Communications, Information System and Computer Engineering (CISCE)*, Shenzhen, China, pp. 256–260, 2022.
[24] Y. Zhang, M. Li, X. Zhang, Y. He and Z. Li, "Defeat magic with magic: A novel ransomware attack method to dynamically generate malicious payloads based on PLC control logic," *Applied Sciences*, vol. 12, no. 17, pp. 8408, 2022.
[25] W. Alsabbagh and P. Langendörfer, "A flashback on control logic injection attacks against programmable logic controllers," *Automation*, vol. 3, no. 4, pp. 596–621, 2022.
[26] N. Zubair, A. Ayub, H. Yoo and I. Ahmed, "Control logic obfuscation attack in industrial control systems," in *Proc. of 2022 IEEE Int. Conf. on Cyber Security and Resilience (CSR)*, Rhodes, Greece, pp. 227–232, 2022.
[27] A. Mochizuki, K. Sawada, S. Shin and S. Hosokawa, "On experimental verification of model based white list for PLC anomaly detection," in *Proc. of 11th Asian Control Conf. (ASCC)*, Gold Coast, QLD, Australia, pp. 1766–1771, 2017.

[28] A. Abbasi, T. Holz, E. Zambon and S. Etalle, "ECFI: Asynchronous control flow integrity for programmable logic controllers," in *Proc. of 33rd Annual Computer Security Applications Conf. (ACSAC)*, Orlando, Florida, USA, pp. 437–448, 2017.
[29] W. Alsabbagh and P. Langendörfer, "A stealth program injection attack against S7-300 PLCs," in *Proc. of 2021 22nd IEEE Int. Conf. on Industrial Technology (ICIT)*, Valencia, Spain, pp. 986–993, 2021.