12-2021

HRPDF: A software-based Heterogeneous Redundant Proactive Defense Framework for Programmable Logic Controller

Ke LIU

Jing-Yi WANG

Qiang WEI

Zhen-Yong ZHANG

Jun SUN
Singapore Management University, junsun@smu.edu.sg

See next page for additional authors

Citation

LIU, Ke; WANG, Jing-Yi; WEI, Qiang; ZHANG, Zhen-Yong; SUN, Jun; MA, Rong-Kuan; and DENG, Rui-Long. HRPDF: A software-based Heterogeneous Redundant Proactive Defense Framework for Programmable Logic Controller. (2021). *Journal of Computer Science and Technology*. 36, (6), 1307-1324.
Available at: https://ink.library.smu.edu.sg/sis_research/6924

Liu K, Wang JY, Wei Q *et al.* HRPDF: A software-based heterogeneous redundant proactive defense framework for programmable logic controller. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 36(6): 1307–1324 Nov. 2021. DOI 10.1007/s11390-021-1647-7

HRPDF: A Software-Based Heterogeneous Redundant Proactive Defense Framework for Programmable Logic Controller

Ke Liu[1], Jing-Yi Wang[2], Qiang Wei[1,*], Zhen-Yong Zhang[2,3], Jun Sun[4], Rong-Kuan Ma[1], and Rui-Long Deng[2]

[1] *State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou 450001, China*

[2] *College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China*

[3] *College of Computer Science and Technology, Guizhou University, Guiyang 550025, China*

[4] *School of Information Systems, Singapore Management University, Singapore 689867, Singapore*

E-mail: {chexin, wangjyee, 12132013, zhangzhenyong}@zju.edu.cn; junsun@smu.edu.sg
{csewmf, dengruilong}@zju.edu.cn

Received June 1, 2021; accepted November 15, 2021.

Abstract Programmable logic controllers (PLCs) play a critical role in many industrial control systems, yet face increasingly serious cyber threats. In this paper, we propose a novel PLC-compatible software-based defense mechanism, called Heterogeneous Redundant Proactive Defense Framework (HRPDF). We propose a heterogeneous PLC architecture in HRPDF, including multiple heterogeneous, equivalent, and synchronous runtimes, which can thwart multiple types of attacks against PLC without the need of external devices. To ensure the availability of PLC, we also design an inter-process communication algorithm that minimizes the overhead of HRPDF. We implement a prototype system of HRPDF and test it in a real-world PLC and an OpenPLC-based device, respectively. The results show that HRPDF can defend against multiple types of attacks with 10.22% additional CPU and 5.56% additional memory overhead, and about 0.6 ms additional time overhead.

Keywords industrial control system, programmable logic controller, proactive defense, heterogeneous redundant architecture

1 Introduction

Industrial control systems (ICSs) are widely used in national critical infrastructures like power grids, oil, natural gas, water conservancy and metallurgy, which are crucial to economy and social stability [1, 2]. Programmable logic controllers (PLCs), as the core of many ICSs, play an essential role to control the industrial processes via the I/O interface. Due to the importance of PLCs, they have become the main target of attackers consequently [3]. For example, in 2011, the "Stuxnet" incident destroyed the nuclear facilities in Natanz, Iran, and eventually delayed the launch of the Bushehr Nuclear Power Plant [4]. Since then, similar cyber attacks against PLC in ICS, such as Duqu [5], BlackEnergy [6], Industroyer [7], and Triton [8], caused huge economic losses and posed a threat even to human lives.

Although being effective in some settings, existing defense approaches against PLC-oriented attacks have the following key limitations. Firstly, defending approaches adopted by intrusion detection systems (IDSs) [9–11], deception defense [12–14] and attestation [15–19] take effect after attacks happened, mainly detecting them but not blocking them, while blocking technologies such as industrial firewalls alone are not able to effectively block growing sophisticated attacks [20–22]. Secondly, most existing approaches are

This work is supported by the National Key Research and Development Program of China under Grant No. 2020YFB2010900 and the Fundamental Research Funds for the Central Universities (Zhejiang University NGICS Platform) of China under Grant No. TC190A449.

*Corresponding Author

designed based on the characteristics of one or several specific kinds of known attacks, such as [9, 13, 15]. Thirdly, most existing approaches are hardware-based and require external devices, such as [11, 13, 17], which are costly, and difficult to deploy and update, especially in a deployed distributed control system (DCS). Specifically, large-scale DCSs often have multiple controllers that are scattered everywhere, and hardware-based approaches may need to monitor the I/O interface of each controller and the network connecting controllers.

To address these issues, in this paper, we present a novel proactive and PLC-compatible defense mechanism, called Heterogeneous Redundant Proactive Defense Framework (HRPDF). Unlike existing approaches[13, 15–17, 23], HRPDF is designed to actively defend against a variety of attacks targeting the core software stack of PLC, including 1) firmware modification attacks (firmware level), 2) control logic tampering attacks (logic application level), and 3) PLC memory attacks (memory level). The success of HRPDF mainly depends on a novel software redundancy framework laying in multiple levels of the PLC's software stack. This idea is inspired by the redundant system architecture to cope with errors and unpredictable failures during normal operation for fault tolerance and stability guarantee, which has been widely used in large-scale control systems like DCSs and modern computer systems[24]. Besides, HRPDF is PLC-compatible, i.e., no external devices are needed while satisfying the real-time and availability requirements of PLC.

Specifically, HRPDF implements heterogeneous redundancy in multiple levels of the PLC's software stack as follows (shown in Section 4). Firstly, in the compiling phase, HRPDF adopts several different compiler enhancement mechanisms to generate different versions of executable binaries. Secondly, in the execution phase, HRPDF uses multiple heterogeneous runtimes with equivalent functionality and executes control logic separately in a PLC's scan cycle. Note that these runtimes are the key to PLC's security enhancement. Thirdly, a runtime manager is built on top of the runtimes to make decisions on the final output and complete a scan cycle. In addition, to minimize the overhead, HRPDF adopts a new communication mechanism to ensure the real-time requirement and data integrity of the communication between the runtimes and runtime manager.

To evaluate HRPDF, we realize HRPDF in a real-world PLC used in distributed energy generation system as well as an OpenPLC-based device①. Note that we focus on ARM-based devices since a large number of PLCs in ICSs use the ARM processors[25]. We first compare HRPDF with several state-of-the-art defense techniques including invariants-based detection mechanism (IDM)[11], ECFI[15], PLCDefender[17], and Shade[26], to validate its defense capability. Besides, we conduct comparative experiments between HRPDF and the original PLC to evaluate its time and performance overhead. Note that this is crucial for HRPDF to be adopted in real-world PLCs. Extensive experiments have shown that the prototype implementation of HRPDF has much stronger defense capabilities compared with the state-of-the-art defending approaches while introducing the low overhead, i.e., with 10.22% additional CPU overhead, 5.56% additional memory overhead, and about 0.6 ms additional time overhead.

In a nutshell, we make the following main contributions.

- We propose a proactive and PLC-compatible defense framework HRPDF, which is a novel software redundancy framework to enhance PLC's security against a wide spectrum of PLC specific attacks.
- We implement HRPDF in both a real-world PLC and an OpenPLC-based device on raspberry. We design compilation enhancement techniques and a lock-free inter-process communication (IPC) mechanism to minimize the overhead of HRPDF to satisfy the real-time and availability requirements.
- We collect 492 control logics from different industries and conduct extensive experiments to validate HRPDF's performance in terms of defending against cyber attacks. The results show that HRPDF is promising in defending against a wide range of attacks with an acceptable cost and does not affect the real-time and availability of PLC.

2 Preliminary

2.1 PLC Programs and PLC Scan Cycle

PLC is the core control unit in ICSs. PLC's software architecture and programming languages are often implemented according to the IEC 61131-3② standard by a third party (e.g., Codesys③). Commonly

①https://www.openplcproject.com/, Oct. 2021.

②https://webstore.iec.ch/publication/62427, Oct. 2021.

③https://www.codesys.com/, Oct. 2021.

used programming languages include ladder diagram (LD), function block diagram (FBD), structured text (ST), instruction list (IL), and sequential function chart (SFC), among which LD, FBD, and SFC are graphic languages, and the rest are textual languages.

A control logic is written with one or multiple languages listed above by the engineer and then compiled on the engineering workstation (EWS). The compiled binary PLC program is subsequently downloaded to the PLC's runtime for execution. During the execution stage, PLC's runtime first scans the input table to read the data from input devices and executes the control logic with the inputs. Then, the execution results of control logic will be flushed and stored in the output table, and finally updated to the output devices. The whole process of reading inputs, executing control logic, and updating outputs is called as a scan cycle [27]. The above work cycle of a PLC is depicted in Fig.1. It is worth mentioning that the memory layout of PLC includes the CODE segment, the CONFIG segment and the OBJECT segment and so on, where the input/output (I/O) table in the OBJECT segment is connected to the field devices.

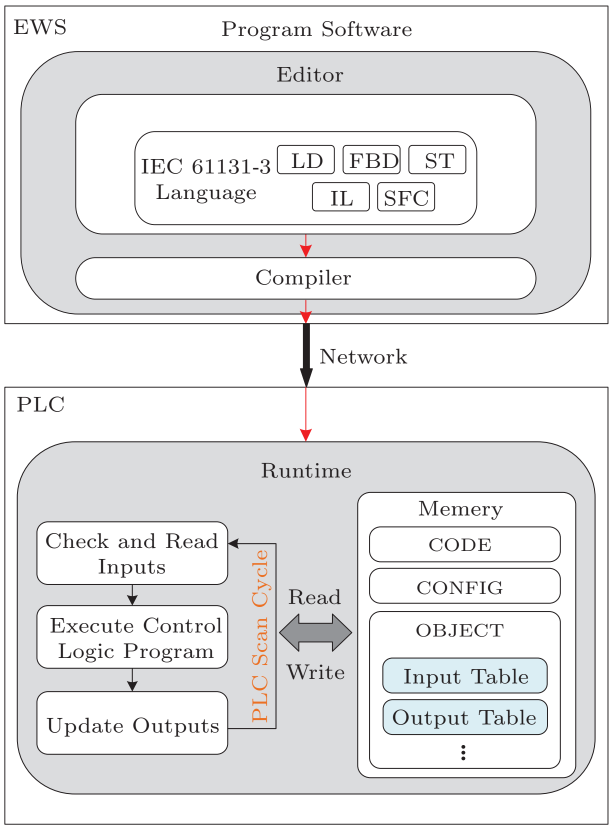


Fig.1. PLC programs downloading process and PLC scan cycle.

2.2 Redundant Architecture

In modern ICS, redundant architecture, which has been widely used in DCS, SIS, etc., is an important fault tolerance technology to cope with errors and unpredictable failures during normal operation and guarantee the stability of the system [24]. We show an example of redundant controller architecture in Fig.2. The controllers A and B are standby to each other and connected by bus. When the system is working, the two controllers will execute the same control logic but only one of them is actively responsible for controlling the field devices while the other serves as a standby controller. Once the active controller fails (e.g., being attacked) and cannot operate normally, the standby controller will take over the system. Note that such traditional redundancy architecture is isomorphic, which means that the hardware and software structures of the active and standby devices are often the same. Furthermore, such redundancy solutions are rather expensive [24]. Moreover, isomorphic redundancy has been proven to be vulnerable against cyber attacks and even introduces new risks [28].

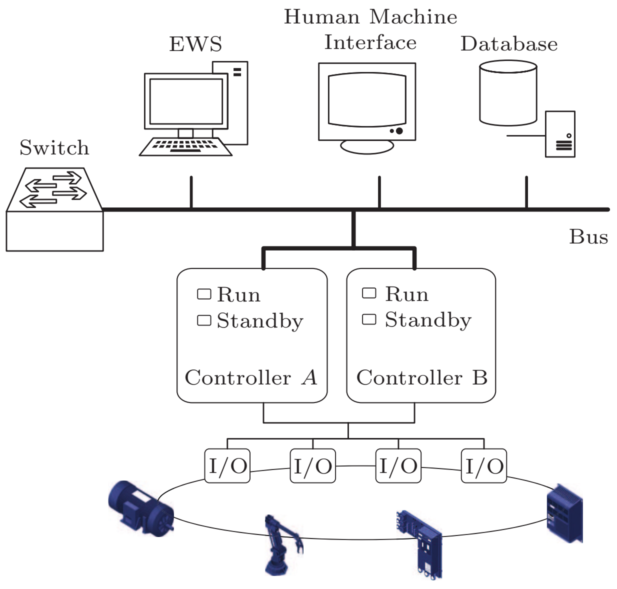


Fig.2. Redundant controller architecture.

3 Threat Model

We first formulate our threat model and then give the problem statement. We assume that the attackers are able to access PLCs in ICS either remotely via the network or physically. Note that this is a reasonable assumption adopted by multiple recent researches [10, 11, 17, 28]. Consequently, the attackers could read and modify the PLC memory, control logic application and firmware, etc. to launch a range of attacks targeting PLC's software stack as follows.

1) *Firmware Modification Attack*[28–31]. This type of attacks exploits the vulnerabilities in the firmware update validation mechanism of a PLC's runtime. The attackers would create a counterfeit firmware sample and download it to a target PLC. The firmware samples are often carefully constructed to execute privileged instructions or pave the way for stealthy attacks.

2) *Control Logic Tampering Attack*[32–36]. This type of attacks could happen in multiple stages: development, transmission and execution. In this work, we focus on the transmission and execution stage, which mainly exploits the vulnerabilities of the authentication mechanism of PLC's runtime. The attackers can manipulate the status of the industrial procedure via the counterfeit control logic.

3) *PLC Memory Attack*[37–39]. There are three types of attacks against PLC's memory in the scope of this paper: attacks on the PLC working memory, attacks on the input table and attacks on the output table. This type of attacks exploits the vulnerabilities of the memory protection mechanism of PLC's runtime. We assume that the attacker has the ability to modify the value of the PLC's virtual memory address and protocol-oriented memory address via memory corruption vulnerabilities or industrial protocol vulnerabilities. However, we assume that the attackers will not bluntly overwrite a large range of memory addresses with jungle bytes to cause unpredictable abnormality, but precisely overwrite the value of one or several specific memory addresses to affect the industrial procedure.

These attacks are common to PLCs which could potentially damage the status of a PLC[28,32,36,37]. In particular, we assume that the attackers have limited knowledge of the defense framework. We do not consider the situation where the attackers can compromise the EWS like Stuxnet, and the insider attacks such as supply chain attacks[40] are also not in the scope of the attacker model.

The objectives of HRPDF is to defend the above attacks proactively with an acceptable cost. The basic research objectives (ROs) are listed as follows.

• *RO*1. The mechanism needs to be able to defend and block such attacks before they take effect.

• *RO*2. The defense mechanism should not occupy too many PLC resources and compromise the availability of the PLC.

• *RO*3. The real-time requirements of the PLC cannot be compromised by the defense mechanism.

4 HRPDF Design

4.1 PLC Internal Design

The overall HRPDF architecture is shown in Fig.3. Inside the redundantly designed PLC, HRPDF consists of a runtime manager and several runtimes. The runtimes are heterogeneous and functionally equivalent to each other, as the core of this framework. Heterogeneity means that the control logic and the memory layout are different. Functional equivalence means that given the same input to the PLC programs in runtimes, they will return the same execution results. This framework not only preserves the original function, but also improves the defense capabilities of the PLC. The runtime manager, as a hypervisor, has the highest priority and is responsible for multiple tasks including runtime management, controlling the PLC scan cycle, recording security event logs, and raising alarms. The communication between the runtimes and the runtime manager will inevitably bring time delay; therefore, it is necessary to design and implement an efficient and practical IPC mechanism to reduce the impact on the real-time performance of the PLC as much as possible.

In addition, the PLC internal design changes the original PLC scan cycle as shown in Fig.4. The detailed workflow of the new PLC scan cycle consists of six phases. At the beginning, the runtime manager reads inputs from the input table and sends them to runtimes. Then, different runtimes execute control logic separately according to the inputs and then send the results back to the runtime manager. Finally, the runtime manager decides the proper results according to the results of these runtimes and flushes them to the storage area and output table. Fig.4 shows that the new PLC scan cycle of HRPDF has three new phases, which are IPC in the input phase, IPC in the output phase, and decision making of the runtime manager. The time overhead analysis experiment shows that the additional time overhead caused by the decision making is negligible for the millisecond scan cycle. Therefore, the main time overhead of HRPDF is IPC in the input phase and the output phase, which need to be evaluated. Note that, due to the compilation enhancement strategy (such as code obfuscation), the control logic in HRPDF will also be more complex than the original. Therefore, the additional execution time should not be ignored.

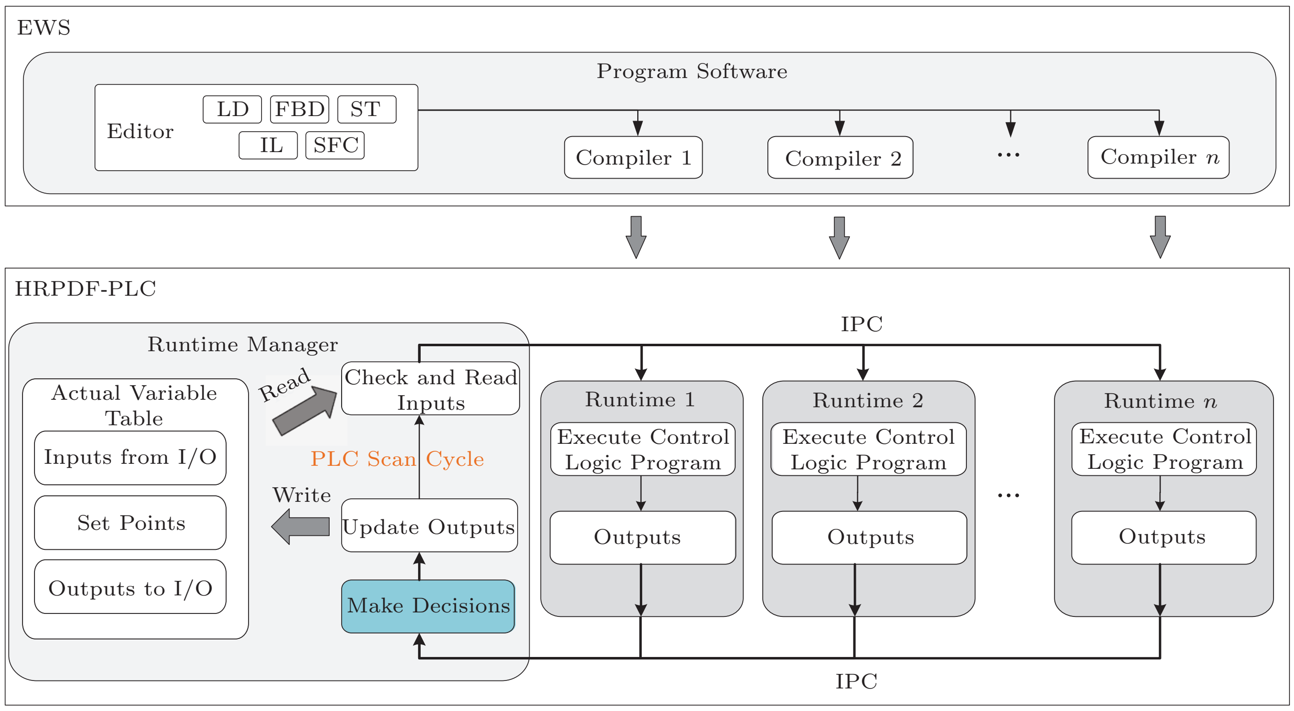


Fig.3. Overall architecture of HRPDF.

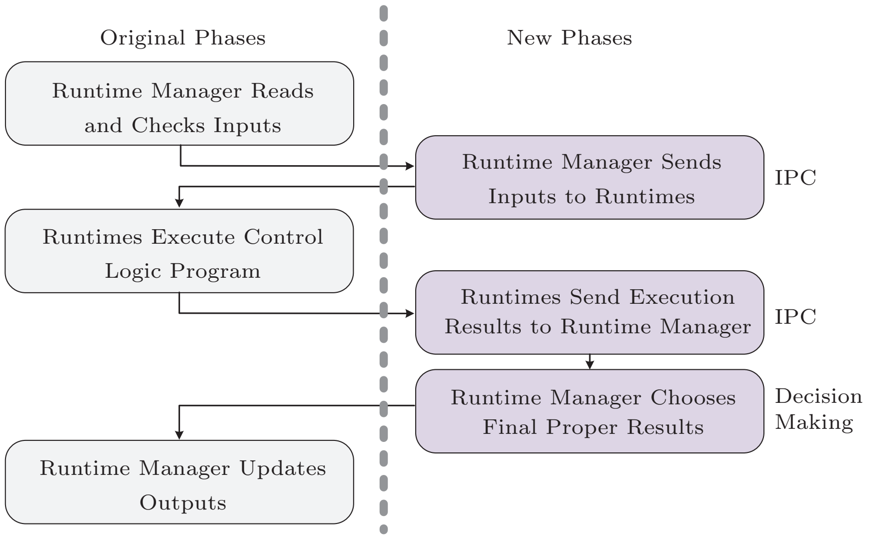

Original Phases
New Phases
Runtime Manager Reads and Checks Inputs
Runtime Manager Sends Inputs to Runtimes
IPC
Runtimes Execute Control Logic Program
Runtimes Send Execution Results to Runtime Manager
IPC
Runtime Manager Chooses Final Proper Results
Decision Making
Runtime Manager Updates Outputs

Fig.4. New PLC scan cycle.

4.2 Engineer Workstation Design (EWS)

EWS is an important part of an ICS for designing, configuring and diagnosing the software and hardware in the system. Logic applications are written, debugged, and downloaded to PLC via EWS. The redundant EWS architecture in HRPDF is shown in Fig.3, responsible for generating multiple functionality-equivalent control logic programs. The principle of the EWS design is to use different compilation enhancement strategies, such as obfuscation and randomization, to ensure that the logic application and memory layout of runtimes are also different. Specifically, the virtual address and the protocol-oriented address of the main memory segment of each runtime are different. The address of any memory segment of the runtime should be useless memory address in other runtimes. Therefore, the attacks on a specific memory (such as

I/O tables) can only affect the normal operation of runtime at most.

5 HRPDF Implementation Details

5.1 Platform

We implement the HRPDF on a real-world PLC, which belongs to the SF-auto CSD-826 PLC family[4]. CSD-826 is an integrated controller mainly used in the energy market, which adopts an 866 MHz dual-core 32 bit ARM Cortex-A7 CPU and runs on a real-time Linux operating system with a PREEMPT Kernel (version 3.14.1). We collaborate with SF-auto and implement the HRPDF based on the original runtime source code. In addition, we also implement HRPDF on an OpenPLC-based device, which is deployed on Raspberry Pi 3 Model B[5]. However, because of the differences between OpenPLC and the real-world PLC, we only implement the PLC's internal design of HRPDF on OpenPLC.

5.2 Compiler Enhancement

5.2.1 Address Randomization

We use address randomization to make sure that the control logic application will be executed in runtimes differently. In particular, we improve the compilation strategy of the CSD-826's original program software by adopting address space layout randomization (ASLR), which is done by randomly placing main memory segments of runtime, such as the CODE segment, the CONFIG segment and the OBJECT segment. To guarantee the success of such address randomization, the compiler further checks that 1) different address segments in the same runtime are not overlapping, and 2) the address of any main memory segment in one runtime should not overlap with other runtimes. Fig.5 shows how address randomization can ensure that the distribution of the main memory segments such as the CODE segment, the CONFIG segment and the OBJECT segment in the virtual memory is different.

Note that we also adopt randomization on address segments handling protocols (separated from the virtual address executing logic applications) to counter attacks initialized from the protocol side. Fig.6 shows how we uniformly randomize the protocol-oriented address, where the specific memory in each segment can further be randomized as shown in Fig.7.

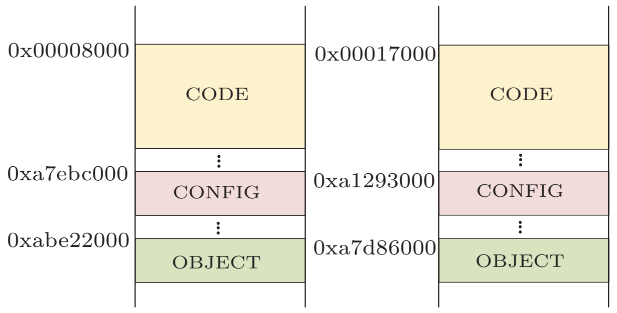


Fig.5. Randomizing the virtual address of the segments.

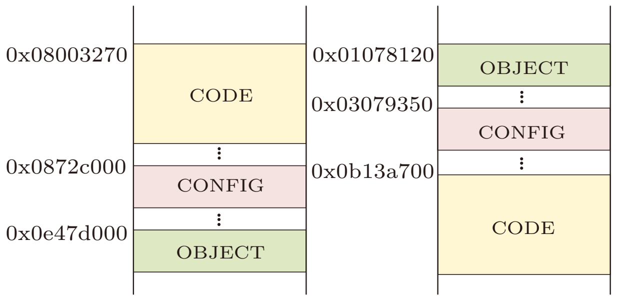


Fig.6. Randomizing the protocol-oriented address of the segments.

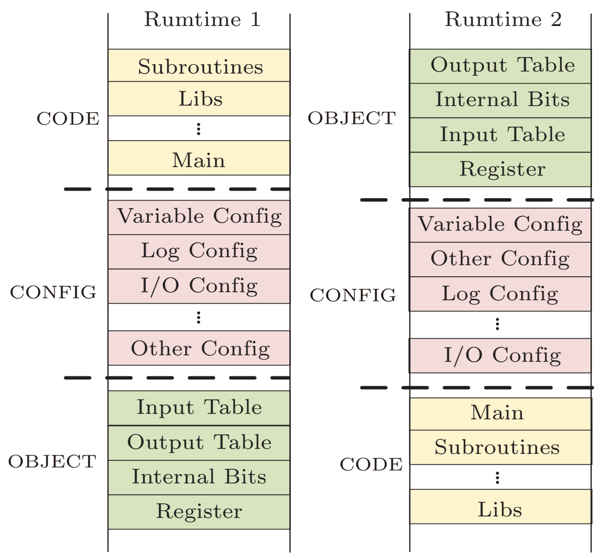


Fig.7. Randomizing the distribution of submodules in segments.

5.2.2 Obfuscation

Obfuscation of logic applications could significantly increase the cost of an attacker to analyze the binary control logic[41]. To implement effective obfuscation, HRPDF will randomly insert some junk instructions in the compiling phase, which could be categorized

[4]https://www.sifang-electric.com/about, Oct. 2021.

[5]https://www.raspberrypi.org/, Oct. 2021.

into executable and non-executable instructions. Executable junk instructions will take up CPU time slices and be able to perform normal memory read/write operations, while not affecting the original control logic functions. A variety of executable junk instructions are designed in HRPDF for different compilation strategies, including reversible operations, invalid assignment instructions, etc.

HRPDF also adopts additional strategies to hinder static reverse engineering. For instance, in the compiling time, HRPDF replaces regular procedure calls with jump tables and branch functions, which prevents an attacker from obtaining the logic applications directly from the PLC for further analysis. Besides, HRPDF also improves ultimate packer for executables[6] to pack the compiled binaries with proprietary strategies, which mainly counters an attacker attempting to obtain the compiled binaries such as the project file of CODESYS V2[42].

Note that in general, the obfuscation technology will incur additional time overhead inevitably, especially the executable junk instructions. Considering the real-time requirements, HRPDF is limited to insert a small number of executable junk instructions (executed in each PLC scan cycle). Meanwhile, since unpacking will only happen once at the beginning, the caused additional time overhead is negligible.

5.3 Runtime Manager and Runtime

5.3.1 Runtime Manager

The runtime manager is the most critical part of HRPDF, which has the highest privilege and priority in the PLC. There are only CONFIG and OBJECT segments in the memory layout of the runtime manager because it is not intended to execute control logic. Note that the I/O table in the OBJECT segment of the runtime manager corresponds to the field devices. In each PLC scan cycle, the runtime manager scans and reads the inputs from the input devices and sends them to runtimes. After executing the control logic, the runtimes send the results, which include execution results and part of the internal storage data, to the runtime manager for decision making. The runtime manager compares the results of each runtime byte by byte and selects the most common values as the final results. Then, the runtime manager flushes the results to the corresponding memory area. If the result of a runtime is abnormal (different from the most common result), the runtime manager will immediately raise an alarm and mark the corresponding runtime as an exception. Note that the runtime manager will only wait for the results for a limited time. If one runtime does not return results beyond a time threshold, the runtime manager will give up waiting and mark the runtime as an exception as well.

When a runtime is marked as an exception, the runtime manager will start the runtime checking mechanism (RCM). RCM mainly scans the existence of the flags in the OBJECT, CONFIG, and CODE segments to judge whether the memory is corrupted, which are random numbers generated and inserted by the compilers. In addition, the runtime manager records the accuracy of each runtime to deal with some corner cases. For example, when all runtimes return different results, the runtime manager will select the results according to the most reliable runtime with the highest historical accuracy.

5.3.2 Runtime

In the heterogeneous runtime, the I/O table does not communicate with the field devices directly. Alternatively, a runtime obtains inputs from and send the results to the runtime manager. In addition, some new functions have been added to the runtime to ensure that the runtime can parse and execute the randomized and obfuscated binaries. Note that it is important to implement runtime isolation, such as assigning different privileges to different runtimes, which can potentially prevent attackers from using a compromised runtime to attack the others.

While generating multiple heterogeneous runtimes, the memory layout, logic application execution function and communication mechanism are taken into consideration. Specifically, we set different compilation configurations for different runtimes, in which the memory layout and strategy for adding junk instructions are different. Then the compiler can compile and generate multiple heterogeneous but functionally equivalent runtimes according to different compilation configurations.

5.4 Communication Mechanism

In HRPDF, the new PLC scan cycle includes two IPCs. To guarantee the normal operation of the new PLC scan cycle and reduce the additional time overhead, the IPC mechanism should be designed to meet

[6]https://upx.github.io/, Oct. 2021.

the following requirements.

• The mechanism should guarantee data integrity, which means that the runtime should not disrupt the runtime manager.

• The mechanism should be lock-free, and will not block the normal operation of the runtime manager and runtimes. The priority of the runtime manager is higher than that of the runtime, and the IPC algorithm or the data structure needs to be able to avoid the priority inversion problem. Furthermore, the mechanism should avoid being exploited by attackers to block the PLC scan cycle.

• The mechanism should not impose any assumptions on the time and speed of reading or writing operations, i.e., the worst-case under attack needs to be considered.

IPC in the input stage is that the runtime manager sends the inputs to runtimes; hence we design a lock-free single-writer/multi-reader mechanism between the runtime manager and runtimes, which have a fixed-size shared memory queue and are based on the atomic read-modify-write primitives. In this paper, we use compare-and-swap (CAS) which is a synchronization primitive widely used in lock-free algorithm implementation[43]. The completed design of the IPC mechanism in HRPDF is shown in Algorithm 1.

The array $Buffer$ stores the data transferred between the runtime manager and runtimes. Assuming that value a_i held by $Reading[i]$ indicates whether $Buffer[a_i]$ is being used. The variable $Latest$ indicates the latest data written by the runtime manager. The function $GetBuff()$ is responsible for selecting a free buffer slot for the runtime manager for the next PLC scan cycle. The array $InUse$ is a local variable used to mark the buffer slot that is currently occupied and is initialized in lines 8–11. During lines 12–18, the runtimes' status and the buffer slot status are acquired and recorded in $InUse$, and the corresponding elements are marked as true. Note that if there is a buffer slot occupied by a runtime, it means that the runtime is abnormal. The runtime manager finds a free buffer slot at the beginning of a new PLC scan cycle, and all runtimes should be idle and waiting for inputs. Therefore, the runtimes that still occupy the buffer slot must be abnormal or under attack. Finally, the first buffer slot with a false value in $InUse$ will be returned to $RuntimeManagerWriter()$. After finding a free buffer slot, the runtime manager writes data and updates $Latest$ in lines 27 and 28. In lines 29–31, the runtime manager checks the status of all runtimes by scanning $Reading$ and then updates all non-zero elements in $Reading$ to inform all runtimes in which buffer slot the input is stored. The operation uses CAS primitives to guarantee atomicity. In function $RuntimeReader()$, the runtime uses the CAS command to assign $Latest$ to $Readind[RtmID]$ in lines 37 and 38. If it succeeds, it means that the runtime manager has written the input data of this PLC scan cycle into $Buffer$, and then the runtime obtains the corresponding buffer slot index from the array $Reading$ and reads the corresponding input data from the proper buffer slot in lines 39 and 40.

Algorithm 1. Communication Mechanism in the Input Phase

```
1: int $N$;
2: Data $Buffer[N+2]$;
3: int $Reading[N]$;
4: int $Latest$;
5:
6: function $GetBuff()$
7:     boolean $InUse[N]$;
8:     int $i, k$;
9:     for $i \leftarrow 0$ to $N$ do
10:        $InUse[i] \leftarrow$ False;
11:    end for
12:    $InUse[Latest] \leftarrow$ True;
13:    for $i \leftarrow 0$ to $N$ do
14:        $j \leftarrow Reading[i]$;
15:        if $j != 0$ then
16:            $InUse[Latest] \leftarrow$ True;
17:        end if
18:        mark runtime $i$ as exception;
19:    end for
20:    for $i \leftarrow (Latest+1)\%(N+2)$ To $N+2$ do
21:        if $InUse[i] ==$ False then return $i$;
22:        end if
23:    end for
24: end function
25:
26: function $RuntimeManagerWriter()$
27:    $widx \leftarrow GetBuff()$;
28:    Write input data to $Buffer[widx]$;
29:    $Latest \leftarrow widx$;
30:    for $i \leftarrow 0$ to $N$ do
31:        $CAS(Reading[i], N+2, widx)$
32:    end for
33: end function
34:
35: function $RuntimeReader()$
36:    $RtmID \leftarrow RuntimeID$;
37:    $Reading[RtmID] \leftarrow N+2$;
38:    $ridx \leftarrow Latest$;
39:    $CAS(Reading[RtmID], N+2, ridx)$;
40:    $ridx \leftarrow Reading[RtmID]$;
41:    Read input data from $Buffer[ridx]$;
42: end function
```

While in the output phase, the runtimes send the results to the runtime manager respectively. We could use multiple fixed-size lock-free single-writer/single-reader queues, and different runtimes use the different queues to communicate with the runtime manager. This problem can be solved by a fixed size circular buffer[7]. The head and tail pointers of the circular buffer are both in the shared memory. The head pointer is written by the writer and the tail pointer is written by the reader. Each runtime uses an independent circular buffer to communicate with the runtime manager. In HRPDF, we implement the IPC mechanism with Michael and Scott's approach [44].

6 Evaluation

In this section, we conduct a series of experiments comparing the original devices and security-enhanced (implemented on both CSD-826 and OpenPLC-based) devices to evaluate the effectiveness of HRPDF mainly from the following two aspects. Firstly, we evaluate the defense capabilities of HRPDF against different kinds of software-oriented attacks. Secondly, we evaluate the real-time and availability performance of HRPDF considering the incurred time and the resource overhead. For each experiment, we test 100 PLC scan cycles.

6.1 Effectiveness in RO1: Defense Capabilities

To evaluate the defense capabilities of the framework, we design and implement the firmware modification attack, the control logic tampering attack, and the PLC memory attack on original CSD-826 PLC and OpenPLC-based device. The details of the attack are as follows.

• *Firmware Modification Attack.* The CSD-826 updates the firmware through the network protocol and lacks verification. Using this vulnerability, we can inject malicious firmware. We could modify the configuration of the output table in the firmware, causing its firmware to fail to execute normally. In CSD-826, we launch an attack from the network as attackers, and then inject malicious firmware, which can stealthily modify the value of the output table to affect the industrial procedure. In the OpenPLC-based device, we manually replace its firmware (an executable program named OpenPLC)

• *Control Logic Tampering Attack.* In CSD-826, we generate the counterfeit downloading packets including malicious control logic, and download them to CSD-826, which can affect the normal operation of the original control logic and tamper with the execution results of the control logic. However in the OpenPLC-based device, we exploit the HTTP service it provides for normal control logic downloading to implement the attack. Note that the normal workflow of the OpenPLC-based device is different from that of real-world PLC. In the OpenPLC-based device, we write the control logic in programming software, from which we can export the structured text (ST) file, and then download it to the PLC through the HTTP service of the OpenPLC-based device runtime. Then, the OpenPLC-based device runtime compiles the ST file and runs the new runtime so that we can exploit the HTTP service to finish the control logic tampering attack.

• *PLC Memory Attack.* We design three types of attacks against the PLC memory. The first is the attack on the CONFIG segment of CSD-826, which overwrites the pointer of the output table to a certain address, and then manipulates the value of virtual memory corresponding to the output table via controlling the value of the address. The second is the attack on the PLC input table, which exploits the authentication variabilities of the industrial protocol and replays the packets to tamper with the value of the protocol-oriented addresses of the input table. The third is the attack on the PLC output table, which is similar to the second attack. In the OpenPLC-based device, we compromise its working memory via the Modbus server.

We conduct attack experiments on the original devices and HRPDF-devices at the same time. The experimental results are shown in Table 1. It can be seen that HRPDF can effectively defend current common various attacks against PLC. Fig.8 shows the basic architecture of our experimental environment.

Case Study. Taking the control logic tampering attack as an example, the adversary illegally tampers with a simple control logic, which can manipulate a value of output table. As shown in Fig.9, this type of attacks can tamper with the original control logic into a forged control logic and then influence the result of its execution, and can easily compromise the original PLC, and then manipulate the field device via affecting the output table. However, it can only tamper one of the runtimes in HRPDF at most because the memory layout and protocol-oriented addresses of the runtimes are different. In the next, the runtime manager could find the compromised runtime because of the abnormal results. Note that we deliberately fix the address of one of the runtimes (runtime 1 in Fig.9) to be consistent with the original PLC runtime for better comparison experiments.

[7]https://en.wikipedia.org/wiki/Circular_buffer, Oct. 2021.

Table 1. Defense Capabilities of HRPDF

Attack Type		CSD-826 (HRPDF)	CSD-826 (Original)	OpenPLC (HRPDF)	OpenPLC (Original)
Firmware modification attack		√	×	√	×
Control logic tampering attack		√	×	√	×
PLC memory attack	CONFIG segment attack	√	×	√	×
	Input table attack	√	×	√	×
	Output table attack	√	×	√	×

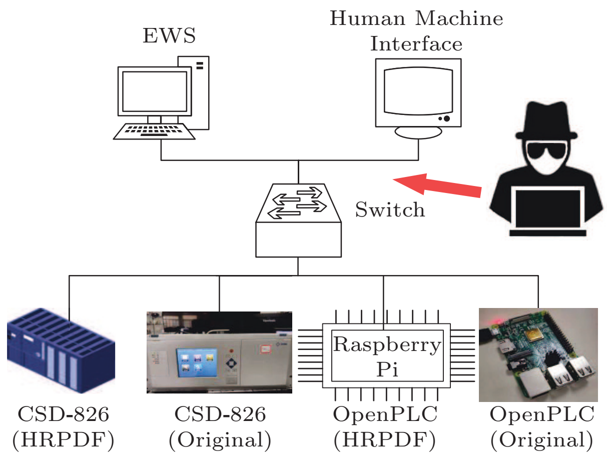


Fig.8. Basic architecture of experimental environment.

We compare defensive capabilities with different types of previous work. IDM is an invariants-based anomaly detection system [11]. ECFI is the first control-flow verification system for PLC and belongs to the category of local attestation [15]. PLCDefender is a remote attestation framework with a physics-based model to preserve the control behavior integrity of PLC [17]. Shade is a novel shadow memory technology against control logic tamper attacks [26]. Table 2 shows the comparison results of the defensive capabilities of these approaches. It can be seen that these defense mechanisms can only be used for specific attacks and cannot comprehensively improve the security of PLCs.

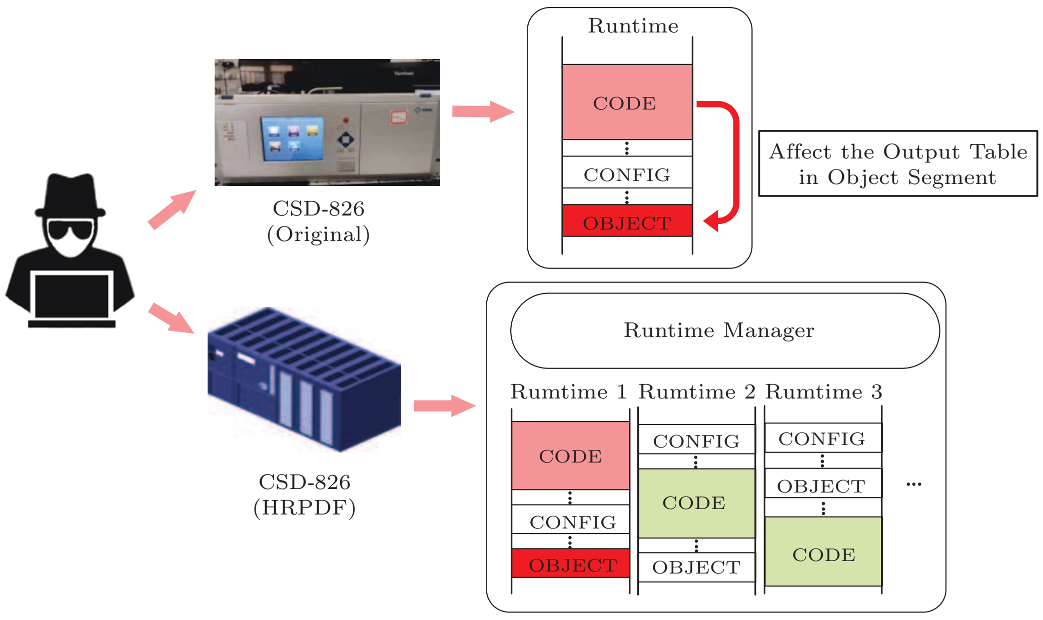


Fig.9. Case study of the control logic tampering attack.

Table 2. Comparison of Defense Capabilities Between HRPDF and Other Mechanisms

Attack Type		HRPDF	IDM	ECFI	PLCDefender	Shade
Firmware modification attack		√	×	√	×	×
Control logic tampering attack		√	×	√	×	√
PLC memory attack	CONFIG segment attack	√	×	×	×	√
	Input table attack	√	√	×	√	×
	Output table attack	√	√	×	√	×

6.2 Effectiveness in RO2: Performance Overhead

To evaluate the performance overhead of the several techniques presented in the paper on real-world cyber physical system (CPS) control logic applications, we widely collect representative control logics on Github. A total of 4 833 control logics in the ST format are collected, in which 492 of them can perform a real-world CPS process. According to the functionalities of the 492 control logics, they can be divided into five categories of CPS processes, including manufacturing, energy, chemical industry, water, and others. We use them to evaluate the performance overhead of HRPDF. The performance overhead is mainly in terms of CPU and memory utilization in this paper. We use the perf tool suite[8] and the top tool[9] that comes with the target system to obtain an accurate measurement value.

Real-World Impact. We execute these control logics on both CSD-826 with HRPDF and the original CSD-826. The comparison results are shown in Fig.10. HRPDF does bring additional resource overhead, but

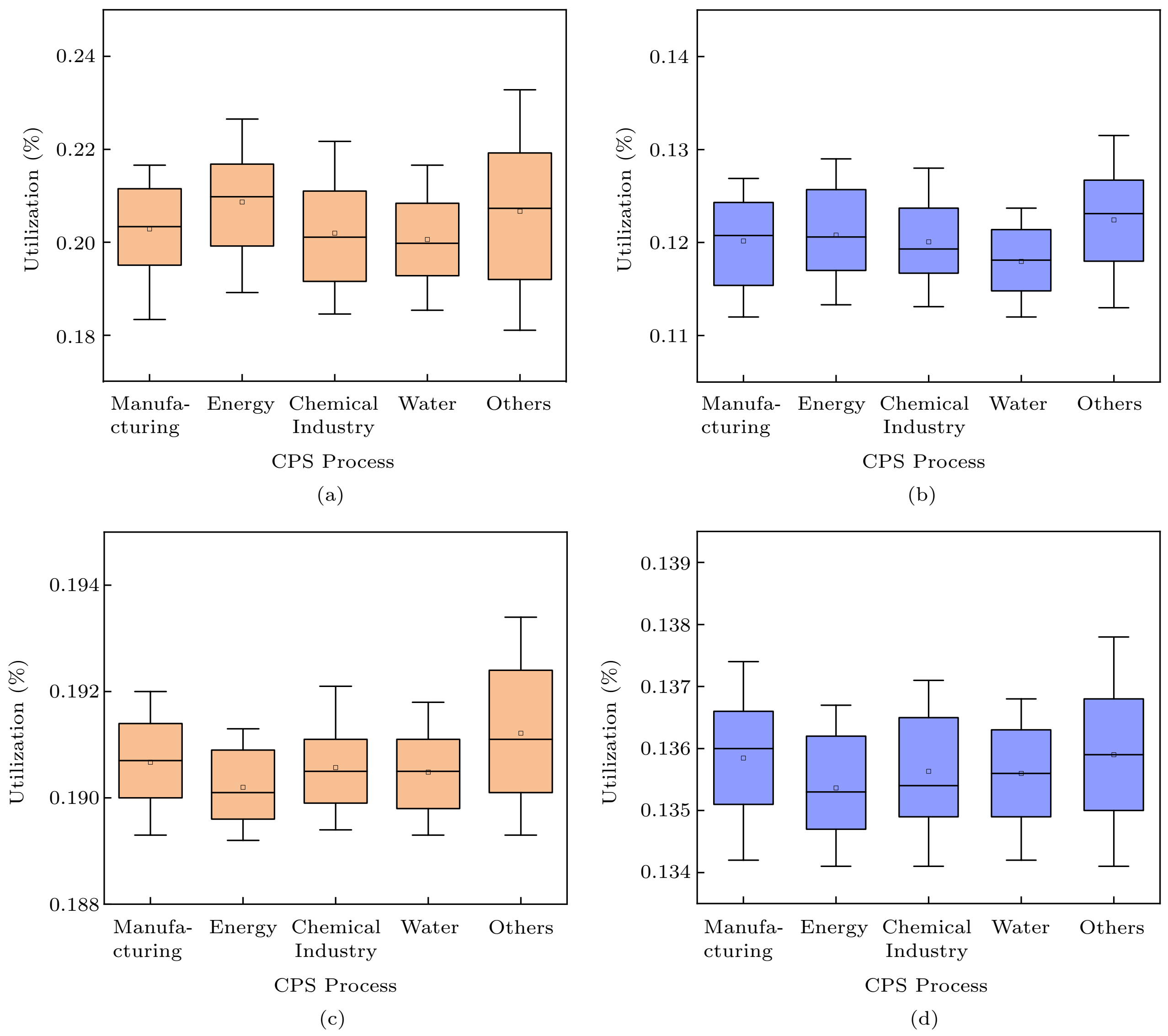


Fig.10. Impact of different control logics on resource overhead. (a) CPU utilization on CSD-826 (HRPDF). (b) CPU utilization on CSD-826 (original). (c) Memory utilization on CSD-826 (HRPDF). (d) Memory utilization on CSD-826 (original).

[8] https://man7.org/linux/man-pages/man1/perf.1.html, Oct. 2021.

[9] https://man7.org/linux/man-pages/man1/top.1.html, Oct. 2021.

the addition resource overhead is in an acceptable range. In addition, the difference of CPS processes has little impact on the additional resource overhead. These experimental results show that our proposed techniques have a limited impact on the performance overhead when executing diverse real-world control logics.

Worst Impact. In order to further evaluate the impact of other indicators on performance overhead, we select two worst cases to evaluate CPU and memory overhead on different implementations of runtimes. The number of runtimes of implementations of HRPDF is set to 3, and the number of I/O is set to 16. Compared with other real-world control logic applications, the PID algorithm [45] and the SHA-2 control logic have the highest average CPU and memory utilization on CSD-826 with HRPDF; therefore, we choose the two cases for the next experiment.

Table 3 shows the CPU and memory utilization of the worst case in different devices. Note that the number of runtimes in HRPDF is set to 3, and the number of I/O is set to 16. The results show that the implementation of the HRPDF framework in CSD-826 causes about 10.22% the additional CPU overhead and 5.56% the additional memory overhead. Even in the worst case, the performance overhead is limited to an acceptable range. An additional 4.18% CPU and 2.09% memory are required in an OpenPLC-based device, which is far from reaching 70% resource utilization.

Runtimes Number. We conduct experiments to observe the impact of the number of runtimes on performance. Fig.11 and Fig.12 show the changes in performance overhead caused by the number of runtimes when the PLC executes SHA-2 control logic. With the increase in the number of runtimes, the increase in the utilization of CPU and memory is acceptable. In CSD-826, each additional runtime will increase about 2.64% CPU and 1.72% memory utilization, while in the OpenPLC-based device it is 0.65% CPU and 0.86% memory utilization.

To prove that such additional overhead is acceptable for PLCs, we investigate the CPU and memory utilization of some off-the-shelf PLCs in normal operations. Because in some PLCs, the telnet or secure shell server is enabled by default, we can get the default username and password from the PLC manual or customer service staffs. Then we can remote login the device and measure the CPU and memory utilization. The results are shown in Table 4. From the results, we can see that the performance of many existing PLCs has been greatly improved, and more than 60% of the CPU and memory resources of the PLC listed in the table are idle. For example, the Wago 750-8212 PLC even leaves more than 90% of the CPU and memory unused. In addition, many modern conventional PLCs use Cortex-A series CPUs, because not all the industrial environments have extremely strict real-time requirements. Our solution inevitably brings some time overhead, but this is acceptable especially in PCS environments.

6.3 Effectiveness in RO3: Time Overhead

HRPDF adds three additional steps to the original PLC cycle, which is the reason for the additional time overhead. Experimental results show that the time overhead for the runtime manager to choose the correct results is negligible. Therefore, we focus on the additional time overhead of a whole PLC scan cycle caused by the IPC between the runtime manager and runtimes. To evaluate the time overhead, we design different configurations to generate the runtime (having different numbers of I/O or different numbers of runtimes).

According to the communication mechanism in Section 5, the time overhead of IPC depends on the amount of the data transferred. Fig.13 shows the influence of the number of I/O on time delay. In this experiment, we set three runtimes, and calculate how much more time that HRPDF spends in the PLC scan cycle than the original device. The results show that as the number of I/O increases, the time delay caused by HRPDF is gradually increasing. Fig.14 shows the influence of the number of runtimes on time delay while the number of I/O is set to 16. The increase in the number of runtimes will also lead to an increase in time delay. The experimental results also show that the delay can be controlled at about 0.6 ms less than 1 ms, which is an acceptable cost for most industrial scenarios.

Table 3. Performance Overhead of the Worst Case in Different Devices

Control Logic	CSD-826 (Original)		CSD-826 (HRPDF)		OpenPLC (Original)		OpenPLC (HRPDF)	
	Average (%)	Worst (%)	Average (%)	Worst (%)	Average (%)	Worst (%)	Average (%)	Worst (%)
PID algorithm	13.27	14.81	23.32	25.03	0.84	0.97	4.69	5.15
SHA-2 control logic	13.78	N/A	19.34	N/A	1.19	N/A	3.28	N/A

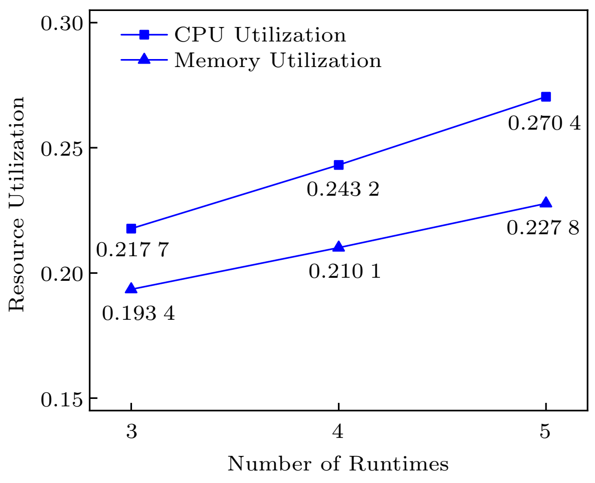


Fig. 11. Impact of number of runtimes on resource overhead (CSD-826).

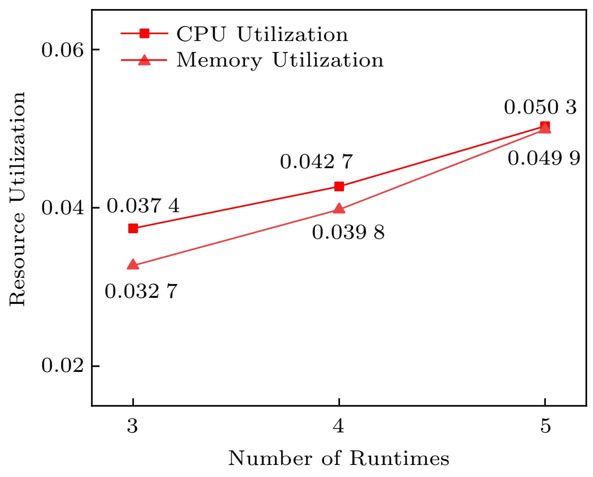


Fig. 12. Impact of number of runtimes on resource overhead (OpenPLC-based device).

Table 4. Average Resource Usage in Different PLCs

Vendor & Controller	CPU	Core	Main Memory (MB)	Runtime	OS	CPU Utilization (%)	Memory Utilization (%)
Wago 750-8212	Cortex A8 1 GHz	1	512	Codesys V3	RTlinux	6.50	8.38
Wago 750-8202	Cortex A8 600 MHz	1	256	Codesys V2	RTlinux	31.41	13.56
Atekon NA-300	ARM9 400 MHz	1	128	N/A	RTlinux	12.02	25.28
Atekon NA-400	Cortex A8 1 GHz	1	512	N/A	VxWorks	3.49	11.51
Hollysys LK-210	Intel Xscale IXP42X 533 MHz	1	512	Codesys V2	Linux	41.43	32.81
SF-auto CSC-850	Cortex A8 1 GHz	2	512	ProConOS	VxWorks	20.96	27.22
SF-auto CSC-830	Cortex A7 866 MHz	2	256	N/A	VxWorks	17.73	21.18
SF-auto CSD-826	Cortex A7 866 MHz	2	512	N/A	RTlinux	23.34	26.73

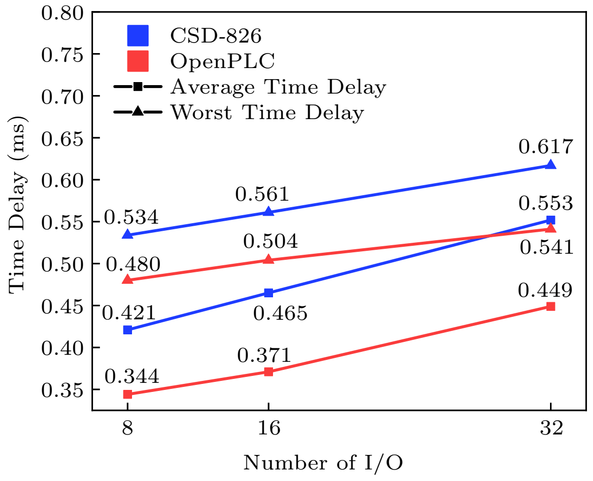


Fig.13. Impact of the number of I/O on time delay.

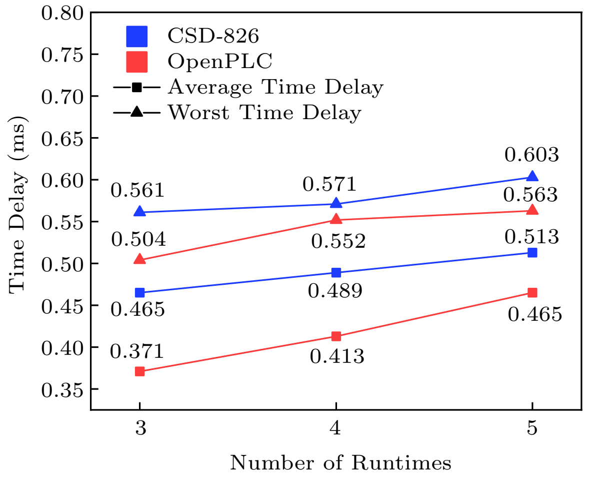


Fig.14. Impact of the number of runtimes on time delay.

In addition, due to the obfuscation technologies, the control logic execution time of runtime also needs to be evaluated. We conduct experiments to compare the execution time of 492 different control logics in our framework with the original device runtime. Fig.15 shows the experimental results, from which we could see that the obfuscation technologies only cause less than 0.015 ms average additional time overhead. This effect is small enough to be negligible for the millisecond scan cycle. Note that the time overhead does not include time for unpacking, which will only happen once after a successful download.

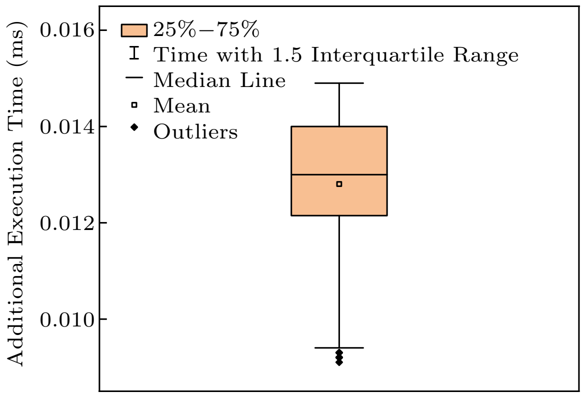


Fig.15. Additional execution time of the control logics.

Discussion. The execution time of each heterogeneous but functionally equivalent runtime is different, which means that the execution time will be subordinated to the slowest runtime. However, HRPDF uses the shared memory for synchronization; therefore, the runtime manager will not be blocked by runtimes and we could set a waiting time threshold for the runtime manager. Once the execution time of a runtime exceeds the threshold, the runtime manager will mark it as an exception and no longer wait. The value of the threshold can be determined according to the actual normal execution time and the real-time requirements of the system. For some extreme time-sensitive scenarios, such time overhead may not be desirable, but for process control systems (PCS), e.g., chemical industry and thermal power, this is completely acceptable.

7 Discussion

In HRPDF, the increase in the number of runtimes will also increase the overhead of EWS, because the number of corresponding compilers will also increase. And to ensure the difference of runtimes, the cost of randomization and obfuscation validation of EWS will also increase. The increase in the number of runtimes has little impact on the performance of the host computer. We do not evaluate the increment of EWS's additional overhead, because we think the resources of EWS are sufficient to cover these overheads.

The runtime manager is the most important part of PLC. The approach we adopt is setting high privilege and closing other additional services. In fact, the runtime manager process can be protected separately. For example, we can use TrustZone to protect it[10]. However, virtually each PLC relies on different architectures and OSs. How to implement the process protection technologies on these diversified architectures and OSs is another practical issue that needs to be considered.

Junk instructions introduced as a part of the obfuscation strategy will cause additional time overhead. In addition, if carelessly selected, these instructions may cause CPU-level scheduler conflicts, resulting in pipeline stalls and even cache coherency issues, which may have a considerable impact on the normal execution of the system. However, we do not find the impact of this problem in actual experiments, and the time overhead generated by junk instructions is completely acceptable. This problem needs to be further analyzed, and it is also one of the next directions for improvement.

8 Related Work

IDS. IDS is a defense mechanism that has been widely studied and applied[46–50]. It usually uses the data, such as network traffic and logs, to build a system model and monitors the operation of the system in real time. Once the behavior against the preset model is detected, it will record and alarm. The IDS in ICS can be divided into misuse-based intrusion detection[46, 47] and anomaly-based intrusion detection[9–11]. Among them, anomaly-based intrusion detection is more widely used in ICS. Graveto *et al.*[48] proposed an intrusion detection mechanism based on shadow security unit (SSU), which can monitor the I/O of the controller in a novel and fine-grained way to detect many attacks against the controller. Due to the working principle of SSU, its deployment difficulty and cost are high, especially in DCSs. In addition, this approach cannot detect stealthy attacks[28, 37] at the firmware level in the sophisticated ICS cyber kill chain[49]. Caselli *et al.*[50] also proposed interesting approaches for PLC runtime monitoring and proposed the S-IDS, which have great detection capabilities for sequence attacks in semantic attacks. However, they focused on the detection of sequential attacks and therefore lacked the ability to detect other attacks, and they did not evaluated the impact on the real-time and availability of the original ICS. A common problem with existing intrusion detection technologies is that they can only detect the attack after it takes effect, but cannot block it before it takes effect. Therefore, when IDS detects an attack, it means that the attack may have occurred and even it has achieved its purpose of destroying ICS.

[10]https://developer.arm.com/ip-products/security-ip/trustzone, Oct. 2021.

Attestation. Attestation needs to pre-set behavior rules or system models and uses the collected PLC data to determine whether there is an abnormality. Previous work on the PLC attestation can be divided into two categories: local attestation [15, 16] and remote attestation [17–19, 23, 51]. The biggest difference between the two is whether the verification is performed on PLC or the external hardware device. However, most of the proposed attestation technologies usually aim at a certain type of attacks and cannot be combined together to improve defensive capabilities. In addition, few of them evaluated the impact of defense mechanisms on the real-time and availability of the original PLC.

Deception Defense. The deception defense is to manipulate the information learned by the attacker during the reconnaissance phase to prevent the attack. DefRec is such a mechanism to disrupt reconnaissance of cyber-physical infrastructures via the physical function virtualization [13]. Honeypot is also one of the effective deception defense technologies and always keeps progressing. López-Morales *et al.* [14] proposed a high-interaction, extensible, and malware-collecting honeypot called HoneyPLC, which significantly pushes the state-of-the-art field forward. However, deception defense does not improve the security of PLC; hence it needs to be combined with other technologies in actual deployment.

Formal Methods. The application of formal methods in ICS security is still in continuous development. Existing work is mainly to perform model checking on PLC programs, most of which analyzes and models the control logic statically, such as [52–55]. It is worth mentioning that Janicke *et al.* [56] proposed a runtime-monitoring technology to ensure the normal operation of ICS. However, this approach mainly focuses on control logic attacks. In addition, this approach is a remote attestation approach so that external devices are required during deployment.

9 Conclusions

We presented a novel PLC-compatible proactive defense framework called HRPDF with redundant architecture in multiple layers. HRPDF includes a runtime manager, multiple heterogeneous runtimes, and compiling security enhancement mechanisms. These heterogeneous but functionally equivalent runtimes can defend against a variety of attacks against PLC. In particular, we proposed an IPC mechanism to meet the real-time and availability requirements of PLC. We implemented a prototype of HRPDF on both real-world PLC and OpenPLC-based device and evaluated it with 492 control logics from different industries. The results showed that the proposed framework has satisfiable defense capabilities against a wide spectrum of software-oriented attacks with 10.22% additional CPU overhead, 5.56% additional memory overhead, and about 0.6 ms additional time overhead. Compared with previous work, HRPDF has stronger defense capabilities and can block the attacks before they take effect within an acceptable cost.

In addition, the HRPDF will be further optimized by considering the unpredictable low-level impact caused by the improvement of the compilation strategies, such as pipeline stalls and even cache coherency issues.

References

[1] McLaughlin S, Konstantinou C, Wang X, Davi L, Sadeghi A, Maniatakos M, Karri R. The cybersecurity landscape in industrial control systems. *Proceedings of the IEEE*, 2016, 104(5): 1039-1057. DOI: 10.1109/JPROC.2015.2512235.

[2] Knowles W, Prince D, Hutchison D, Disso J F P, Jones K. A survey of cyber security management in industrial control systems. *International Journal of Critical Infrastructure Protection*, 2015, 9: 52-80. DOI: 10.1016/j.ijcip.2015.02.002.

[3] Zonouz S, Rrushi J, McLaughlin S. Detecting industrial control malware using automated PLC code analytics. *IEEE Security* & *Privacy*, 2014, 12(6): 40-47. DOI: 10.1109/MSP.2014.113.

[4] Farwell J P, Rohozinski R. Stuxnet and the future of cyber war. *Survival*, 2011, 53(1): 23-40. DOI: 10.1080/00396338.2011.555586.

[5] Bencsáth B, Ács-Kurucz G, Molnár G, Vaspöri G, Buttyán L, Kamarás R. Duqu 2.0: A comparison to Duqu. Technical Report, CrySyS Lab, 2015. https://www.crysys.hu/publications/files/duqu2.pdf, Nov. 2021.

[6] Lee R M, Assante M, Conway T. Analysis of the cyber attack on the Ukrainian power grid. Technical Report, Electricity-Information Sharing and Analysis Center, 2016. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/05/20081514/E-ISAC_SANS_Ukraine_DUC_5.pdf, Nov. 2021.

[7] Lee R, Slowik J, Miller B, Cherepanov A, Lipovsky R. Industroyer/Crashoverride: Zero things cool about a threat group targeting the power grid. Technical Report, Black Hat, 2017. https://www.blackhat.com/docs/us-17/wednesday/us-17-Lee-Industroyer-Crashoverride-Zero-Things-Cool-About-A-Threat-Group-Targeting-The-Power-Grid.pdf, Nov. 2021.

[8] Di Pinto A, Dragoni Y, Carcano A. TRITON: The first ICS cyber attack on safety instrument systems. Technical Report, Nozomi Networks, 2018. https://www.nozominetworks.com/downloads/US/Nozomi-Networks-TRITON-The-First-SIS-Cyberattack.pdf, Nov. 2021.

[9] Ponomarev S, Atkison T. Industrial control system network intrusion detection by telemetry analysis. *IEEE Transactions on Dependable and Secure Computing*, 2016, 13(2): 252-260. DOI: 10.1109/TDSC.2015.2443793.

[10] Zhang F, Kodituwakku H A D E, Hines W, Coble J B. Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system and process data. *IEEE Transactions on Industrial Informatics*, 2019, 15(7): 4362-4369. DOI: 10.1109/TII.2019.2891261.

[11] Feng C, Palleti V R, Mathur A, Chana D. A systematic framework to generate invariants for anomaly detection in industrial control systems. In *Proc. the 2019 Network and Distributed System Security Symposium*, February 2019. DOI: 10.14722/ndss.2019.23265.

[12] Cifranic N, Hallman R, Romero-Mariona J, Souza B, Calton T, Coca G. Decepti-SCADA: A cyber deception framework for active defense of networked critical infrastructures. *Internet of Things*, 2020, 12: Article No. 100320. DOI: 10.1016/j.iot.2020.100320.

[13] Lin H, Zhuang J, Hu Y C, Zhou H. DefRec: Establishing physical function virtualization to disrupt reconnaissance of power grids' cyber-physical infrastructures. In *Proc. the 27th Network and Distributed System Security Symposium*, February 2020. DOI: ndss.2020.24365.

[14] López-Morales E, Rubio-Medrano C, Doupé A, Shoshitaishvili Y, Wang R, Bao T, Ahn G J. HoneyPLC: A next-generation honeypot for industrial control systems. In *Proc. the 2020 ACM SIGSAC Conference on Computer and Communications Security*, November 2020, pp.279-291. DOI: 10.1145/3372297.3423356.

[15] Abbasi A, Holz T, Zambon E, Etalle S. ECFI: Asynchronous control flow integrity for programmable logic controllers. In *Proc. the 33rd Annual Computer Security Applications Conference*, December 2017, pp.437-448. DOI: 10.1145/3134600.3134618.

[16] Garcia L, Zonouz S, Wei D, De Aguiar L P. Detecting PLC control corruption via on-device runtime verification. In *Proc. the 2016 Resilience Week*, August 2016, pp.67-72. DOI: 10.1109/RWEEK.2016.7573309.

[17] Salehi M, Bayat-Sarmadi S. PLCDefender: Improving remote attestation techniques for PLCs using physical model. *IEEE Internet of Things Journal*, 2021, 8(9): 7372-7379. DOI: 10.1109/JIOT.2020.3040237.

[18] McCune J M, Li Y, Qu N, Zhou Z, Datta A, Gligor V, Perrig A. TrustVisor: Efficient TCB reduction and attestation. In *Proc. the 2010 IEEE Symposium on Security and Privacy*, May 2010, pp.143-158. DOI: 10.1109/SP.2010.17.

[19] Dessouky G, Zeitouni S, Nyman T, Paverd A J, Davi L, Koeberl P, Asokan N, Sadeghi A. LO-FAT: Low-overhead control flow attestation in hardware. In *Proc. the 54th Annual Design Automation Conference*, June 2017, Article No. 24. DOI: 10.1145/3061639.3062276.

[20] Cheminod M, Durante L, Seno L, Valenzano A. Performance evaluation and modeling of an industrial application-layer firewall. *IEEE Transactions on Industrial Informatics*, 2018, 14(5): 2159-2170. DOI: 10.1109/TII.2018.2802903.

[21] Li D, Guo H, Zhou J, Zhou L, Wong J W. SCADAWall: A CPI-enabled firewall model for SCADA security. *Computers & Security*, 2019, 80: 134-154. DOI: 10.1016/j.cose.2018.10.002.

[22] Jiang N, Lin H, Yin Z, Xi C. Research of paired industrial firewalls in defense-in-depth architecture of integrated manufacturing or production system. In *Proc. the 2017 IEEE International Conference on Information and Automation*, July 2017, pp.523–526. DOI: 10.1109/ICInfA.2017.8078963.

[23] Zeitouni S, Dessouky G, Arias O, Sullivan D, Ibrahim A, Jin Y, Sadeghi A R. ATRIUM: Runtime attestation resilient under memory attacks. In *Proc. the 2017 IEEE/ACM International Conference on Computer-Aided Design*, November 2017, pp.384-391. DOI: 10.1109/ICCAD.2017.8203803.

[24] Stój J. Cost-effective hot-standby redundancy with synchronization using EtherCAT and real-time ethernet protocols. *IEEE Transactions on Automation Science and Engineering*, 2021, 18(4): 2035-2047. DOI: 10.1109/TASE.2020.3031128.

[25] Schwartz M D, Mulder J, Trent J, Atkins W D. Control system devices: Architectures and supply channels overview. Technical Report, Sandia National Laboratories, 2010. https://energy.sandia.gov/wp-content/gallery/uploads/JCSW_Report_Final.pdf, Nov. 2021.

[26] Yoo H, Kalle S, Smith J, Ahmed I. Overshadow PLC to detect remote control-logic injection attacks. In *Proc. the 16th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, June 2019, pp.109-132. DOI: 10.1007/978–3–030–22038–9_6.

[27] Bryan L A, Bryan E A. Programmable Controllers: Theory and Implementation (2nd edition). Industrial Text Company, 1997.

[28] Ma R, Cheng P, Zhang Z, Liu W, Wang Q, Wei Q. Stealthy attack against redundant controller architecture of industrial cyber-physical system. *IEEE Internet of Things*, 2019, 6(6): 9783-9793. DOI: 10.1109/JIOT.2019.2931349.

[29] Basnight Z, Butts J, Lopez J, Dube T. Firmware modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection*, 2013, 6(2): 76-84. DOI: 10.1016/j.ijcip.2013.04.004.

[30] Schuett C, Butts J, Dunlap S. An evaluation of modification attacks on programmable logic controllers. *International Journal of Critical Infrastructure Protection*, 2014, 7(1): 61-68. DOI: 10.1016/j.ijcip.2014.01.004.

[31] Garcia L, Brasser F, Cintuglu M, Sadeghi A, Mohammed O, Zonouz S. Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit. In *Proc. the 26th Network and Distributed System Security Symposium*, February 26–March 1, 2017. DOI: 10.14722/ndss.2017.23313.

[32] Govil N, Agrawal A, Tippenhauer N O. On ladder logic bombs in industrial control systems. In *Proc. the 2017 International Workshop on Security and Privacy Requirements Engineering and the 2017 International Workshop on the Security of Industrial Control Systems and Cyber-Physical Systems*, September 2017, pp.110-126. DOI: 10.1007/978–3–319–72817–9_8.

[33] Senthivel S, Dhungana S, Yoo H, Ahmed I, Roussev V. Denial of engineering operations attacks in industrial control systems. In *Proc. the 8th ACM Conference on Data and Application Security and Privacy*, March 2018, pp.319-329. DOI: 10.1145/3176258.3176319.

[34] Yoo H, Ahmed I. Control logic injection attacks on industrial control systems. In *Proc. the 34th IFIP TC 11 International Conference on ICT Systems Security and Privacy Protection*, June 2019, pp.33-48. DOI: 10.1007/978–3–030–22312–0_3.

[35] Kalle S, Ameen N, Yoo H, Ahmed I. CLIK on PLCs! attacking control logic with decompilation and virtual PLC. In *Proc. the Workshop on Binary Analysis Research*, February 2019. DOI: 10.14722/bar.2019.23074.

[36] Sun R, Mera A, Lu L, Choffnes D. SoK: Attacks on industrial control logic and formal verification-based defenses. In *Proc. the 2021 IEEE European Symposium on Security and Privacy*, September 2021, pp.385-402. DOI: 10.1109/EuroSP51992.2021.00034.

[37] Abbasi A, Hashemi M. Ghost in the PLC designing an undetectable programmable logic controller rootkit via pin control attack. In *Proc. the 2016 Black Hat Europe*, November 2016.

[38] Robles-Durazno A, Moradpoor N, McWhinnie J, Russell G, Maneru-Marin I. Implementation and detection of novel attacks to the PLC memory of a clean water supply system. In *Proc. the 4th International Conference on Technology Trends*, August 2019, pp.91-103. DOI: 10.1007/978–3–030–05532–5_7.

[39] Robles-Durazno A, Moradpoor N, McWhinnie J, Russell G, Maneru-Marin I. PLC memory attack detection and response in a clean water supply system. *International Journal of Critical Infrastructure Protection*, 2019, 26: Article No. 100300. DOI: 10.1016/j.ijcip.2019.05.003.

[40] Hou Y, Such J, Rashid A. Understanding security requirements for industrial control system supply chains. In *Proc. the 5th IEEE/ACM International Workshop on Software Engineering for Smart Cyber-Physical Systems*, May 2019, pp.50-53. DOI: 10.1109/SEsCPS.2019.00016.

[41] Behera C K, Bhaskari D L. Different obfuscation techniques for code protection. *Procedia Computer Science*, 2015, 70: 757-763. DOI: 10.1016/j.procs.2015.10.114.

[42] Keliris A, Maniatakos M. ICSREF: A framework for automated reverse engineering of industrial control systems binaries. In *Proc. the 26th Annual Network and Distributed System Security Symposium*, February 2019. DOI: 10.14722/ndss.2019.23271.

[43] Valois J D. Lock-free linked lists using compare-and-swap. In *Proc. the 14th Annual ACM Symposium on Principles of Distributed Computing*, August 1995, pp.214-222. DOI: 10.1145/224964.224988.

[44] Michael M M, Scott M L. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *Proc. the 15th Annual ACM Symposium on Principles of Distributed Computing*, May 1996, pp.267-275. DOI: 10.1145/248052.248106.

[45] Ang K H, Chong G, Li Y. PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 2005, 13(4): 559-576. DOI: 10.1109/TCST.2005.847331.

[46] Vollmer T, Alves-Foss J, Manic M. Autonomous rule creation for intrusion detection. In *Proc. the 2011 IEEE Symposium on Computational Intelligence in Cyber Security*, April 2011, pp.1-8. DOI: 10.1109/CICYBS.2011.5949394.

[47] Lin H, Slagell A, Di Martino C, Kalbarczyk Z, Iyer R K. Adapting bro into SCADA: Building a specification-based intrusion detection system for the DNP3 protocol. In *Proc. the 8th Annual Cyber Security and Information Intelligence Research Workshop*, January 2013, Article No. 5. DOI: 10.1145/2459976.2459982.

[48] Graveto V, Rosa L, Cruz T, Simões P. A stealth monitoring mechanism for cyber-physical systems. *International Journal of Critical Infrastructure Protection*, 2019, 24: 126-143. DOI: 10.1016/j.ijcip.2018.10.006.

[49] Assante M J, Lee R M. The industrial control system cyber kill chain. Technical Report, SANS Institute, 2015. https://sansorg.egnyte.com/dl/HHa9fCekmc, Nov. 2021.

[50] Caselli M, Zambon E, Kargl F. Sequence-aware intrusion detection in industrial control systems. In *Proc. the 1st ACM Workshop on Cyber-Physical System Security*, April 2015, pp.13-24. DOI: 10.1145/2732198.2732200.

[51] Kovah X, Kallenberg C, Weathers C, Herzog A, Albin M, Butterworth J. New results for timing-based attestation. In *Proc. the 2012 IEEE Symposium on Security and Privacy*, May 2012, pp.239-253. DOI: 10.1109/SP.2012.45.

[52] Frey G, Litz L. Formal methods in PLC programming. In *Proc. the 2000 IEEE International Conference on Systems, Man and Cybernetics*, October 2000, pp.2431-2436. DOI: 10.1109/ICSMC.2000.884356.

[53] Adiego B F, Darvas D, Vinuela E B, Tournier J C, Bliudze S, Blech J O, Suarez V G. Applying model checking to industrial-sized PLC programs. *IEEE Transactions on Industrial Informatics*, 2015, 11(6): 1400-1410. DOI: 10.1109/TII.2015.2489184.

[54] Kuzmin E, Sokolov V A, Ryabukhin D. Construction and verification of PLC-programs by LTL-specification. *Automatic Control and Computer Sciences*, 2015, 49(7): 453-465. DOI: 10.3103/S014641161407013X.

[55] Ryabukhin D, Kuzmin E. LTL-specification, verification and construction of PLC programs. In *Proc. the Spring/Summer Young Researchers' Colloquium on Software Engineering*, May 2014, pp.19-26. DOI: 10.15514/SYRCOSE-2014-8-3.

[56] Janicke H, Nicholson A, Webber S, Cau A. Runtime-monitoring for industrial control systems. *Electronics*, 2015, 4: 995-1017. DOI: 10.3390/electronics4040995.

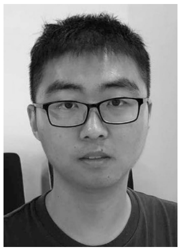
Ke Liu is a Ph.D. candidate in the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou. He received his B.S. degree in computer science from the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, in 2017. His research interests include ICS (industrial control system) security, web security and program analysis of binary code.

Jing-Yi Wang is currently a tenure-track assistant professor at the College of Control Science and Engineering, Zhejiang University, Hangzhou. He received his Ph.D. degree in information system technology and design from Singapore University of Technology and Design, Singapore, in 2018, and his B.S. degree in information engineering from Xi'an Jiaotong University, Xi'an, in 2013. He was a research fellow at the School of Computing, National University of Singapore, Singapore, during 2019–2020, and at Information Systems Technology and Design Pillar, Singapore University of Technology and Design during 2018–2019. His research interests include formal methods, software engineering, cyber-security and machine learning.

Qiang Wei received his Ph.D. degree in computer science and technology from China National Digital Switching System Engineering and Technological Research Center, Zhengzhou. He is currently a professor with the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou. His research interests include network security, industrial internet security and vulnerability discovery.

Zhen-Yong Zhang received his Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, in 2020, and B.S. degree in automation from Central South University, Changsha, in 2015. He was a visiting scholar in Singapore University of Technology and Design, Singapore, from 2018 to 2019. Currently, he is a professor in the College of Computer Science and Technology, Guizhou University, Guiyang, and a research fellow in the School of Control Science and Engineering, Zhejiang University, Hangzhou. His research interests include cyber-physical system security, applied cryptography and machine learning security.

Jun Sun is currently a tenured associate professor at the School of Information Systems, Singapore Management University, Singapore. He received his B.S. and Ph.D. degrees in computing science from the National University of Singapore (NUS), Singapore, in 2002 and 2006, respectively. From 2010 to 2019, he was an assistant/associate professor at the Singapore University of Technology and Design, Singapore. He was a visiting scholar at MIT, Cambridge, from 2011 to 2012. His research focuses on software engineering, formal methods, program analysis, and cyber-security. He is the co-founder of the PAT model checker.



Rong-Kuan Ma received his Ph.D. degree in the State Key Laboratory of Mathematical Engineering and Advanced Computing, Zhengzhou, in 2021. He is a system security researcher. His research interests include program analysis and embedded system security, iCPS security, and Web security.

Rui-Long Deng received his B.Sc. and Ph.D. degrees both in control science and engineering from Zhejiang University, Hangzhou, in 2009 and 2014, respectively. He was a research fellow with Nanyang Technological University, Singapore, from 2014 to 2015, an AITF Postdoctoral Fellow with the University of Alberta, Edmonton, from 2015 to 2018, and an assistant professor with Nanyang Technological University, Singapore, from 2018 to 2019. Currently, he is a professor with the College of Control Science and Engineering, Zhejiang University, Hangzhou, where he is also affiliated with the School of Cyber Science and Technology, Zhejiang University. His research interests include cyber security, smart grid, and communication networks. Dr. Deng serves/served as an associate editor for IEEE Transactions on Smart Grid and IEEE/KICS Journal of Communications and Networks, and a guest editor for IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Cloud Computing, and IET Cyber-Physical Systems: Theory & Applications. He also serves/served as a symposium chair for IEEE SmartGridComm'19 and IEEE GLOBECOM'21.