SHADOWPLCS: A Novel Scheme for Remote Detection of Industrial Process Control Attacks

Junjiao Liu, *Student Member, IEEE*, Xiaodong Lin, *Fellow, IEEE*, Xin Chen, *Member, IEEE*, Hui Wen, Hong Li, Yan Hu, Jiawei Sun, Zhiqiang Shi, and Limin Sun

Abstract—Industrial Control System (ICS) security has become increasingly important as attacks targeting ICSs are more prominent. Although many off-the-shelf industrial network intrusion detection mechanisms have been presented in the past, attackers have always found unique disguisable ways to bypass detections and disrupt actual industrial control processes. To mitigate this deficiency, we present a novel scheme for the detection of industrial process control attacks, called SHADOWPLCS. Specifically, the scheme first automatically analyzes the PLC control code, then extracts key parameters of the PLCs including valid register addresses, valid range of values, and control logic rules as a basis for evaluating attacks. The attack behavior is detected in real-time from different perspectives through active communication with PLCs and passive monitoring of the network traffic. We implemented a prototype system with Siemens S7-300 series PLCs as a case study. Our scheme was evaluated using two Siemens S7-300 PLCs deployed on a gas pipeline network platform. Experiments demonstrate that the presented scheme can accurately detect process control attacks in real-time without affecting the normal operations of PLCs. Compared with the other four representative detection models, our scheme has better detection performance with detection accuracy of 97.3 percent.

Index Terms—Industrial control system, process control attacks, PLC, active and passive, attack detection, control logic

1 INTRODUCTION

ICS is a general term for a class of control systems used in industrial production. It includes Supervisory Control and Data Acquisition (SCADA), distributed control systems and others commonly found in industrial sectors and critical small control systems for infrastructure (such as Programmable Logic Controller, PLC), etc. [43]. ICS is the core of critical infrastructure in traditional countries such as electricity, transportation, and water conservancy, etc [24], [35]. With the advancement of Industry 4.0, ICSs have transferred the communication protocol originally operated on the serial link to TCP/IP while improving their level of informatization. Unfortunately, this shift also provides a convenient way for hackers to carry out attacks [1], [3], [5], [9], [12], [18], [25], [39].

In the field of industrial control, the application of PLC control technology has grown to be an indispensable part of the industry. There is no doubt that the use of PLCs is permeating today's industrial automation. Engineers program these PLCs by developing control codes to control the physical process such as assembly line in automotive manufacturing, elevator. However, PLCs have become a new and effective attack surface for attackers because they directly control the target process. PLCs can be injected with malicious data or have their control code be tampered, causing unexpected results and even tragedies. For example, "Stuxnet" [16], [27] worm tampered the control program of the PLCs in the Iranian nuclear power plant, speeding up the centrifuge speed and damaging the devices. To escape detection, it also responds normal data to confuse control center.

Various off-the-shelf industrial network Intrusion Detection Systems(IDS) have been proposed [10], [11], [13], [17], [19], [31], [33], [48]. The essence of state-of-the-art IDSs are to monitor the content and periodicity of ICS network messages and verify if any major changes have been made. The ICS behavior must be predictable in order for IDS to be feasible, and unfortunately, there still remain critical attacks which the existing IDS can neither detect nor prevent. There are three main reasons for this: First, consistent characteristics are not always observed from the target systems, limiting the extraction of detection rules. Second, each ICS is unique, as a result, domain experts must be closely involved in the creation of the detection rules. However, this process is very expensive and time consuming, stemming from expert error so as to miss key details in the detection rules. Finally, attackers usually use various strategies to evade detection. Therefore, it is difficult for IDSs to detect covert attacks from the root cause.

To overcome the aforementioned limitations of existing solutions, we present a novel attack detection scheme called

- *Junjiao Liu, Jiawei Sun, Zhiqiang Shi, and Limin Sun are with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China. E-mail: {liujunjiao, sunjiawei, shizhiqiang, sunlimin}@iie.ac.cn.*
- *Xiaodong Lin is with the School of Computer Science, University of Guelph, Guelph, ON N1G 1Y4, Canada. E-mail: xlin08@uoguelph.ca.*
- *Xin Chen, Hui Wen, and Hong Li are with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100864, China. E-mail: {chenxin, wenhui, lihong}@iie.ac.cn.*
- *Yan Hu is with the School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China. E-mail: b1727813@ustb.edu.cn.*

Manuscript received 5 June 2020; revised 30 Nov. 2020; accepted 17 Dec. 2020. Date of publication 21 Dec. 2020; date of current version 13 May 2022. (Corresponding author: Hui Wen.) Digital Object Identifier no. 10.1109/TDSC.2020.3046267

SHADOWPLCs. The need of SHADOWPLCs for ICS is motivated through an analysis of five representative industrial process control attacks - *illegal address access attack, malicious data injection attack, configuration tampering attack, control logic infection attack* and *control program replacement attack*. Our analysis shows that the state-of-the-art IDSs are insufficient, especially in *control logic infection attack* and *control program replacement attack*. It is difficult to detect these two attacks using single-point content monitoring and predicting because they are very advanced and covert.

Industrial process runs automatically depending on the setting of the PLC control program, so PLC code carries the most comprehensive and complete information of the industrial control process. SHADOWPLCs automatically analyzes PLC control codes and extracts key PLC parameters, including valid register addresses, valid value ranges, and control logic rules as a baseline for detecting attacks. In other words, SHADOWPLCs can obtain the most complete inspection specifications without prior knowledge and human effort to comprehend physical processes. Based on the rules thus obtained, SHADOWPLCs actively communicates with PLCs, maps their storage space, passively monitors the network traffic, and detects process control attacks in real-time from different perspectives.

For the purpose of the feasibility of our novel scheme, we implemented a prototype system with Siemens S7-300 series PLCs as a case study. Although this system was developed for specific PLC manufacturers and models, the proposed scheme can be modified and expanded to accommodate other PLCs and protocols. In the end, our scheme was evaluated using two Siemens S7-300 PLCs deployed on a gas pipeline network platform, and pressure on the processing power of the PLC network has been collected. Experiments show that our scheme can accurately detect process control attacks in real time without affecting the normal operation of the PLC. Compared with the other four representative detection approaches, our scheme has better detection performance. In particular, the detection accuracy of *control logic infection attack* and *control program replacement attack* can reach 94.7 and 97.3 percent, respectively, while the best detection results in other approaches are 78.7 and 87.3 percent, respectively.

Overall, the key *contributions* of our work are summarized as follows:

- A novel scheme for extracting detection rules leveraging PLC code is presented, which can understand the actual industrial control process without prior knowledge and human effort.
- Active and passive intrusion detection methods are used together to detect the process control attack in real time from different perspectives. To the best of our knowledge, this work takes the lead in applying a combination of active and passive approaches for ICS intrusion detection.
- Based on the Siemens S7-300 series PLCs, we developed an intrusion detection prototype. Although this system was developed for specific PLC manufacturers and models, based on this idea, it can be modified to expand and adapt to other PLCs and protocols.

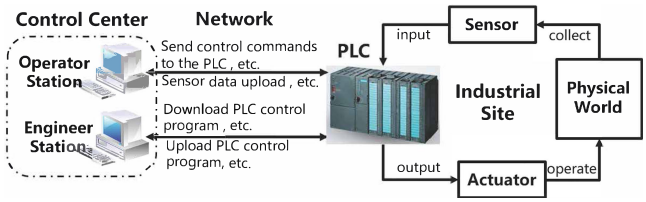


Fig. 1. The basic composition of the environment of an ICS.

- We evaluated the scheme using two Siemens S7-300 PLCs deployed on a gas pipeline network platform. Experiments show that this scheme can accurately detect process control attacks in real-time without affecting the normal operation of the PLCs. Compared with the other four representative detection approaches, our scheme outperforms the other approaches in terms of detection accuracy.

Roadmap. We structure the remainder of this paper as follows. Section 2 briefly introduces the related technical background of PLC, and in Section 3 we summarize process control attacks against PLC. Section 4 details our proposed detection scheme. Section 5 mainly describes the specific implementation. The experimental evaluation is detailed in Section 6. Section 7 discusses the advantages and limitations of our detection scheme. Section 9 is the conclusion. Section 8 introduces the related work of current ICS intrusion detection methods. Finally, Section 9 concludes our work.

2 TECHNICAL BACKGROUND

As shown in Fig. 1, an ICS environment consists of a control center and industrial sites. The bridges that link them are PLCs. A PLC is a digital logic controller with microprocessors that are programmed to define control logic to maintain the desired state of the physical process [4]. PLCs are connected to control center via monitoring network. The engineering station downloads the process control program to PLCs, and then PLCs automatically control the industrial production according to the control logic instructed by the program. An operator station monitors the industrial process by requesting the measurement data of the industrial site from PLCs. When necessary, operators can send control commands to PLCs to carry out corresponding operations on production process. In industrial field, PLC is connected to sensors and actuators, and control programs use sensor data as input and set outputs to activate actuators.

2.1 PLC Working Process

The working process of a PLC is carried out in a cycle scan. When a PLC is in the running time, its operating cycle can be divided into three basic phases(see Fig. 2). 1)Input Sampling Phase. PLC scans each input port one by one, saves the current states of all input devices to the corresponding memory area, and refers to the memory area dedicated to store the states of input devices as the input image registration; 2)Program Execution Phase. CPU reads user instructions from the user program storage area, and after performing corresponding actions, generates corresponding results, and refreshes corresponding output image registers. The corresponding states of input image registers, output image registers, middle registers, etc. are required during

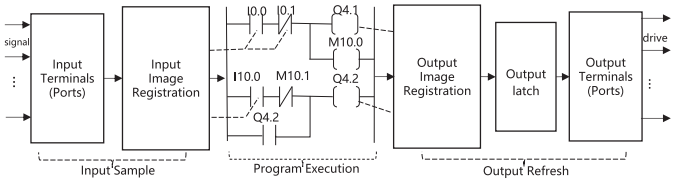


Fig. 2. Simplified sequence of a PLC working process.

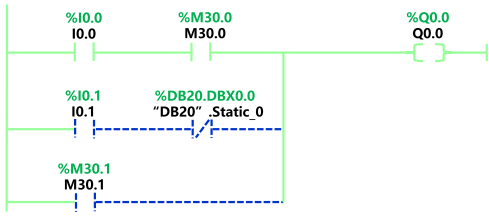


Fig. 3. TIA online monitoring PLC program running status.

this period; and 3)Output Refresh Phase. System program transfers the contents of output image register to output latch, and passes outputs through output terminal to drive external load. Output latches keep their states until next cycle, while states of output registers are dynamic during the program execution phase.

2.2 PLC Languages

The International Electrotechnical Commission standard IEC 61131-3 [14] is an international standard for PLC languages, which stipulates the syntax, semantics and display in control logic programming, and partially modifies the previous programming languages to form the current common five languages : Ladder Diagram (LD), Sequential Function Charts (SFC), Function Block Diagram (FBD), Structured Text (ST) and Instruction List (IL).

Many PLC manufacturers have introduced PLC products that satisfy the IEC 61131-3 standard. For example, most of Rockwell's PLC products have software options that comply with the standard structure text. Schneider's Modicon TSX Quantum PLC products can use the Concept software package that is consistent with this standard. It supports the Modicon 984 ladder diagram and also follows the five programming languages of the IEC 61131-3 standard. Siemens SIMATIC S7-200, S7-300, S7-400, C7-620 and other series of PLCs use the SIMATIC software package, in which the ladder diagram and functional block diagram are in conformity with this standard. Although different manufacturers or models of PLCs have different representations in control code programming, they are invariable and have commonality with each other. The difference between grammar and programming concepts is basically small. Therefore, based on the above facts, combined with our presented detection scheme, we can customize the development of separate ICS environments.

2.3 PLC Control Logic Monitoring

Almost all PLC programming software provides a program monitoring mode that allows remote and intuitive monitoring of the operating status of PLC program. As shown in Fig. 3, the TIA Portal V13 monitors the ladder program of the Siemens S7-300 PLC online. It actively communicates with the PLC to obtain the current operating status and display it graphically. The ladder diagram logic in Fig. 3 can be transformed into mathematical expression as $(I0.0 \ \& \ M30.0) \mid (I0.1 \ \& \ \neg DB20.DBX0.0) \mid M30.0 = Q0.0$. In the PLC at this time, the input registers *I0.0* and *I0.1*, the middle registers *M30.0* and *M30.1*, the data block *B20.DBX0.0* and the output register *Q0.0* have values of 1,0,1,0,1,1, respectively. Bring them into mathematical logic operations, we can get the same result as Fig. 3. This is a low-interference method that does not affect normal operation of the PLC. Inspired by this, our intrusion detection scheme extracts detection rules from PLC code, and also uses a low-interference method to communicate with PLCs, maps their current information in memory, performs real-time remote monitoring of industrial control processes, and discovers process control attacks behavior.

3 Attack Model

For attackers, the ultimate goal of attacking an ICS is to operate the physical process without being detected by advanced IDSs or factory operators [38]. In this section, we first discuss the adversary model under consideration, and then present the five representative attack scenarios.

3.1 Adversary Model

Adversaries can conduct remote cyber attacks on PLCs, such as obtaining control authority of control center or attacking PLCs as a middleman. We consider that the goal of adversaries is to manipulate or disrupt the actual industrial control process. In this work, we have investigated the cyber attacks on PLCs in recent years, divided the main attack methods into two categories: 1) False data injection. 2) Control logic injection.

False data injection refers to inject malicious data that does not conform to the industrial control process to PLCs, and eventually lead to the destruction of process control. For example, write inversion or min/max [8], modifying key setpoint variables and process values [16]. Stephen McLaughlin *et al.* [29] introduced a tool SABOT that automatically maps control commands in a PLC to the target control system behavior specification. This mapping restores sufficient semantics of the PLC's internal layout and can instantiate any malicious controller code. Meixell *et al.* [32] pointed out that simple serial protocols (such as Modbus and DNP3) have been included in IP datagrams. Attackers can just construct an IP-based control packet and send it to the PLCs, which can have serious consequences. Tzokatziou *et al.* [44] pointed out that since PLC communication protocol is a plaintext transmission, there is no authentication process for the communicating object. Therefore, attackers can use the CoDeSys system to directly connect with PLCs, capture network messages, and then directly send the tampering control command to PLCs to achieve the purpose of manipulating the PLC, such as starting and stopping.

Control logic injection refer to tamper or replacement of the original control logic running on target PLCs [31], [41]. Attackers inject malicious control logic to target PLCs by interfering with the normal download/upload control logic engineering operation. "Stuxnet" [27] worm tampered with PLC control program, speeding up the centrifuge speed of

the Iranian nuclear power plant, and responding historical data to confuse control center. Yoo and Ahmed [50] proposed two covert control logic injection attacks, namely *Data Execution Attack* and *Fragmentation and Noise Padding Attack*, to hide the transmission of control logic on the network, preventing packet header signature and deep packet inspection. Sushma Kalle *et al*. [21] proposed a new remote attack method, *CLIK*, which automatically modifies the control logic running in the remote target PLC to interrupt the physical process. *CLIK* also uses a new virtual PLC approach to hide malicious modifications by using engineering software to capture network traffic from raw (uninfected) control logic. McLaughlin [30] designed a malware for PLC that generates dynamic loads from process observations acquired inside control system. Saranyan Senthivel [40] makes the PLC successfully execute malicious programs by manipulating PLC's ladder logic to spoof engineering software. Control logic injection demands high technical skill from adversaries. They must master relevant knowledge of computer and control science in order to make necessary control program modifications or replacements so as to achieve any attack purpose relatively covertly, and state-of-the-art network IDSs have difficulty detecting these types of intrusions.

3.2 Attack Scenarios

Based on the adversary model discussed above, we consider the following attack scenarios that could severely disrupt the industrial control process: *illegal address access attack, malicious data injection attack, configuration tampering attack, control logic infection attack* and *control program replacement attack*. In this paper, we only focus on these five industrial process control attack scenarios, and do not consider physical attacks, firmware attacks, side channel attacks, etc., because they may be less feasible or harmful, or be detectable by existing IDSs.

Illegal Address Access Attack. In this attack scenario, assume that PLCs are black boxes to adversaries, but adversaries have the ability to remotely access the PLC's register space. In order to effectively attack PLCs, adversaries will first probe the PLC's register space to find the key attack execution points. Such as exploring the input and output interfaces, timers, counters, key parameters that PLC is using. This attack is easy to launch, and adversaries do not need to know the actual industrial process.

Malicious Data Injection Attack. It is assumed that adversaries already know some key control points in PLCs, such as register *Q0.0* to control the opening or closing of a valve. Adversaries focus on these key control points, resulting in disordered control. A common attack method is to force a write operation to a register address, overwriting the current value of the register. For example, before the sewage purification is completed, the valve is closed. At this time, the register *Q0.0* is "0". Adversaries forcedly injected "1" into *Q0.0*, covering the actual state of *Q0.0*, causing the valve to open and sewage to leak. This kind of attack is simple and rude, and the effect is significant, but it is not easy to hide. Representative attacks such as [32], [44].

Configuration Tampering Attack. Suppose adversaries get some knowledge of the industrial process, they indirectly damage the industrial process by maliciously tampering with key program configuration information. For example, tampering with timer and counter settings, etc., launch delay attacks. Tampering with PID control parameters causes system oscillation. Tamper with the range of key variables in the physical process, such as the frequency conversion range of a particular centrifuge variable frequency drive is between 807 Hz and 1210 Hz. After a PLC runs the program, the configuration information is not normally monitored by a control center in real time. Therefore, once critical configuration information is tampered with, it will be hard to detect and accurately locate it only by monitoring network traffic. Representative attacks such as [8], [16].

Control Logic Infection Attack. Suppose adversaries have the ability to steal the PLC's current control program. They interfere with the actual industrial process through infection control programs. For example, replace the value in the I/O register with a controllable value somewhere in memory to control the state of the actuator. Insert/delete instructions/rungs. Replace operators in equations. Modify set points. Modify control flow determinants (variables that influence decisions on conditional branches of control logic),etc. Representative attacks such as [21], [27], [30], [40].

Control Program Replacement Attack. In this attack scenario, adversaries do not infect the original PLC control program, but implant malicious control code and try to let PLC execute them. Because this attack is completely dominated by adversaries to destroy any industrial process that they want to accomplish, it has a strong concealment and is destructive and is ready to launch. Representative attacks such as *Data execution attack* and *Fragmentation and Noise padding attack* [50].

3.3 Addressing Attacks

The essence of state-of-the-art ICS IDSs are essentially monitoring the content or periodicity of ICS network messages and verify whether there get some significant changes in them. Since they are constant or predictable in ICS networks, these IDSs have certain detection effects for *illegal address access attacks* and *malicious data injection attacks*.

However, ICS network traffic lacks real-time monitoring of program configuration, control logic running status, etc. Therefore, IDSs generally have poor detection effects on *control logic infection attack* and *control program replacement attack*. Comparison and evaluation with other four representative detection methods prove that our scheme has better detection performance. In particular, the detection accuracy of *control logic infection attack* and *control program replacement attack* can reach 94.7 and 97.3 percent, while the best detection results in other approaches are 78.7 and 87.3 percent, respectively.

4 Approach

In this work, we present a novel scheme for detecting process control attacks, called ShadowPLCs, which mainly includes two phases: rules generation and attack detection. In this section, we will introduce leveraging PLC code for signature based detection rule generation and low-disturbance active and passive intrusion detection core algorithms.

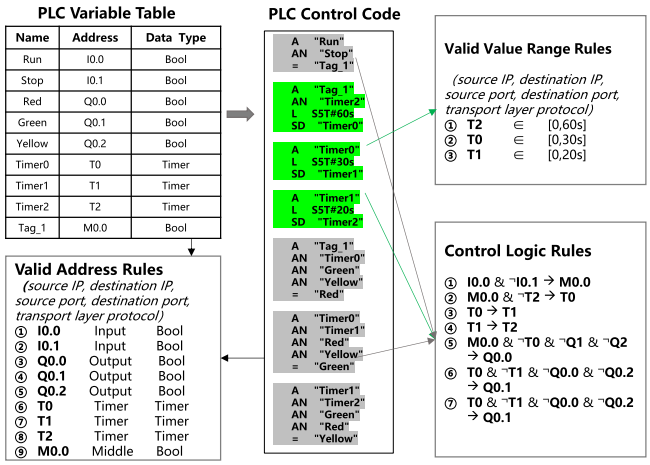


Fig. 4. Generating detection rules based on one-way traffic light control code.

4.1 Leveraging PLC Code for Signature-Based Detection Rules Generation

Control logic is a program that is repeatedly executed in a PLC that controls the actual industrial process. PLC code carries the most comprehensive and complete information of control process, such as contacts, coils, instructions, etc., which are constant values or register addresses that act as variables. These information are not fully available only by mirroring ICS network traffic. By analyzing PLC code, the IDS signature set that can be identified mainly has three types of whitelist rules: the valid address rules, the valid value range rules, and the control logic rules.

1) *Valid addresses rules*. The valid address space inside a PLC is fixed and it is related to the control program. Under normal circumstances, the control center only accesses some valid addresses and most of the operations are read, because the valid address stores the current device status information. Due to lack of knowledge of the ICS, many attackers use brute force methods to infect vulnerable registers. If there is access beyond the valid address ranges, this abnormal behavior can be quickly identified.

2) *Valid value range rules*. PLC code contains important configuration information. Once they are tampered with, the steady state of the physical world will be destroyed. For example, set-point information such as timers and counters, tampering with them may cause early or delayed attacks. The value range of some key variables, such as the frequency conversion range of the centrifugal frequency converter driver is between 807 Hz and 1210 Hz. Some special instructions will also affect the value range, such as the "div" instruction cannot be divided by 0, otherwise it will cause equipment failure. Some control algorithms have key parameters, such as Proportional-Integral-Derivative (PID) algorithm, by modifying them, adversaries aim at causing a controllable deviation from the targeted system's steady state. We analyze PLC code and generate valid value range rules to monitor key configuration information in PLCs.

3) *Control logic rules*. ICS guides industrial process through control logic of PLC programs. Common control methods include on-off logic control, analog control, motion control, process control, and so on. Various register variables (such as input registers, output registers, middle registers, timers, counters, etc.) in a PLC cooperate with each other to complete automation control goal. Therefore, we parse the PLC code, extract the logical relationships between register variables, and then generate control logic rules. Using these rules as a baseline, detect malicious attacks such as PLC control logic infection or program replacement.

As shown in Fig. 4, we take one-way traffic light control as an example to demonstrate the use of PLC code to generate signature-based detection rules. The one-way traffic light control program contains two files. One file is the program source code, and it is the automatic control program of the signal light. The other file is the PLC variable table, which records the variable name, actual PLC address, and data type. We first query the variable table and replace the variable names in the source code. As shown in the first two blocks(rungs) of Fig. 4, Such as *"Run"* $\leftrightarrow$ *I0.0*, *"Stop"* $\leftrightarrow$ *I0.1* and *"Tag_1"* $\leftrightarrow$ *M0.0*.

```
A  "Run"    <->  A   I0.0
AN "Stop"   <->  AN  I0.1
=  "Tag_1"  <->  =   M0.0

A  "Tag_1"   <->  A   M0.0
AN "Timer2"  <->  AN  T2
L  S5T#60s   <->  L   S5T#60s
SD "Timer0"  <->  SD  T0
```

Then perform lexical, grammatical, and semantic analysis on the code instructions to generate valid address rules, such as valid addresses *I0.0, I0.1, M0.0, T2, T0*; Valid value range rules, such as $T2 \in [0,60s]$, and control logic rules, such as $I0.0 \ \& \ \neg I0.1 \rightarrow M0.0$, $M0.0 \ \& \ \neg T2 \rightarrow T0$.

4.2 Active and Passive Combined Attack Detection Model

We presented an intrusion detection scheme combining active and passive approaches. As shown in Fig. 5, this IDS includes two parts: a passive detection engine and an active detection engine. The passive detection engine detects abnormal behaviors such as illegal address operations and illegal value violations by passively monitoring network traffic. The active detection engine actively communicates with PLCs, maps the memory space of PLCs with low interruption and polling, and monitors the industrial control process for abnormalities in real time. It is difficult and costly for attackers to avoid the intrusion detection scheme combining active and passive, and to stealthy themselves. Because they need to constantly simulate the normal industrial field state and conform to the control logic, and even need to tamper with the firmware or the network communication module of devices, rather than cheat IDS and control center by simple ARP spoofing and replay packets and other traditional means.

4.2.1 Passive Detection

The passive detection engine first mirrors the network traffic flowing through industrial switch, and extracts the *five-tuple information (source IP, destination IP, source port, destination port, transport layer protocol), operation address, operation instruction* and *specific value* through in-depth parse of industrial control protocol. Then it checks whether the above information conforms to the valid addresses whitelist and valid value range

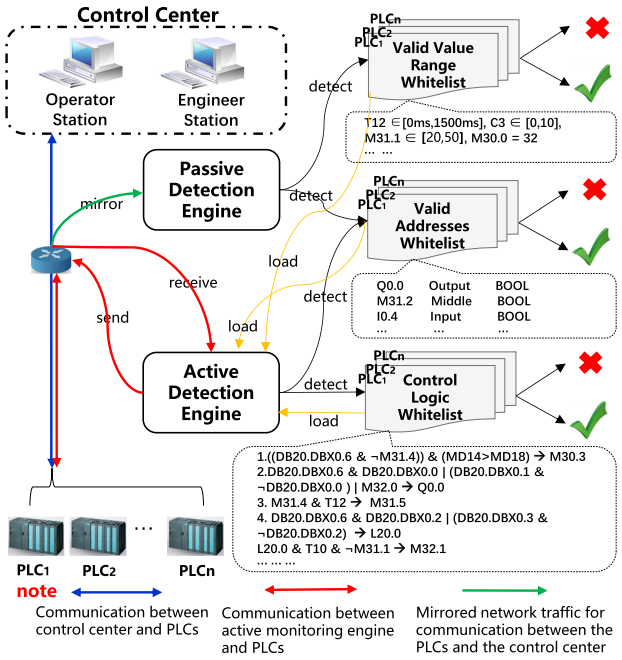


Fig. 5. Process control attack detection.

whitelist, and detect abnormal behaviors such as illegal links, illegal address operations and illegal values.

As shown in Fig. 6, the red part of the network information list indicates an abnormality. The valid range of the counter *C1.0* is between 0 and 20, but the value of *C1.0* is 23 in network, so the alarm is *C1.0* illegally out of bounds. In ②, *M30.0* is not in the valid address rules whitelist, so *M30.0* is an illegal address. In ③, *IP 192.168.20.4* and *port 35555* are new connection and are not in the whitelists, so an illegal link is reported. Algorithm 1 shows the process of passive detection. Line 1 indicates a deep parsing protocol and extracts *five-tuples*, *addresses*, *operations* and *values*. Line 2 is illegal link detection. When unknown *IP*, unknown *port*, and unknown communication appear in network traffic, the illegal link is reported. Line 4 detects illegal addresses. When there are addresses that are not actually used by the PLC in the network traffic, it will alarm illegal address; line 6 indicates detection of illegal value range. When values are out of range, etc., it will alarm illegal value range.

4.2.2 Active Detection

The active detection engine mainly detects control logic anomalies, preventing important memory addresses from

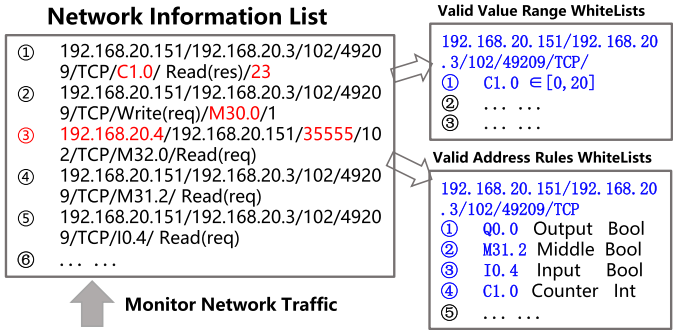


Fig. 6. An example of passive detection.

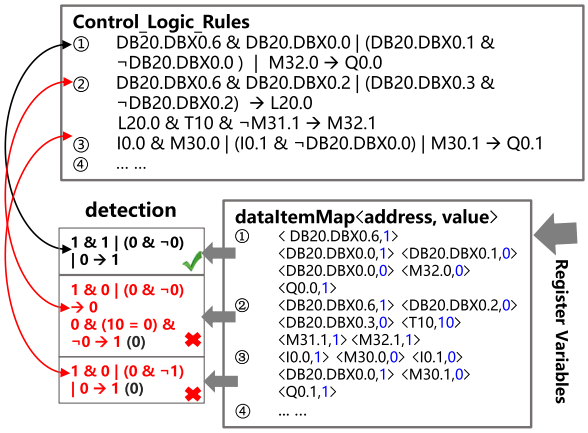


Fig. 7. A case of control logic detection.

being injected with malicious data, key configuration tampering, and control programs being tampered with or replacement. A control logic rule usually consists of multiple register variables. These variables at one time can detect the current program running status of PLC. Because the communication network between PLCs and control center is static, the monitoring of industrial field status adopts a request-response method, which cannot meet the requirements for simultaneous monitoring of associated multi-register variables. We actively establish a connection with a PLC, construct a network data packet according to the communication protocol format, read multiple associated register variables at one time, and map the storage space of the PLC without affecting the normal operations of the PLC. Specifically, the active detection module first actively connects to PLCs and maps corresponding memory space, and then verifies whether they meet the valid value range whitelist and the control logic whitelist.

Algorithm 1. Passive Detection Algorithm

Input:

1. Network_Packets_Queue
2. Valid_Address_WhiteList
3. Valid_Value_Range_WhiteList
4. Five_Tuple

Output: Abnormal Alarm

1: *five_tuples,address,operation,value* = protocolParsing (Network_Packets_Queue);
2: **if** *five_tuples not in Five_Tuple* **then**
3: Alarm("Illegal link: *five_tuples.srcIP, five_tuples.desIP, five_tuples.srcPort, five_tuples.desPort, five_tuples.transferProtocol*");
4: **if** *address not in Valid_Address_WhiteList* **then**
5: Alarm("Illegal Address: *address*, Command: *operation*");
6: **if** *(address, value) not in Valid_Value_Range_WhiteList* **then**
7: Alarm("Illegal Range of Value: *address, value*, 'Value_Range'");

Fig. 7 shows a case of control logic attack detection. *DataItemMap < address, value >* represents PLC memory space mapped by the active detection module. Register addresses and variable value (blue in Fig. 7) form key-value pairs. Control logic rules are the detection benchmark, such as ① *DB20.DBX0.6 & DB20.DBX0.0 | (DB20.DBX0.1 & ¬DB20.*

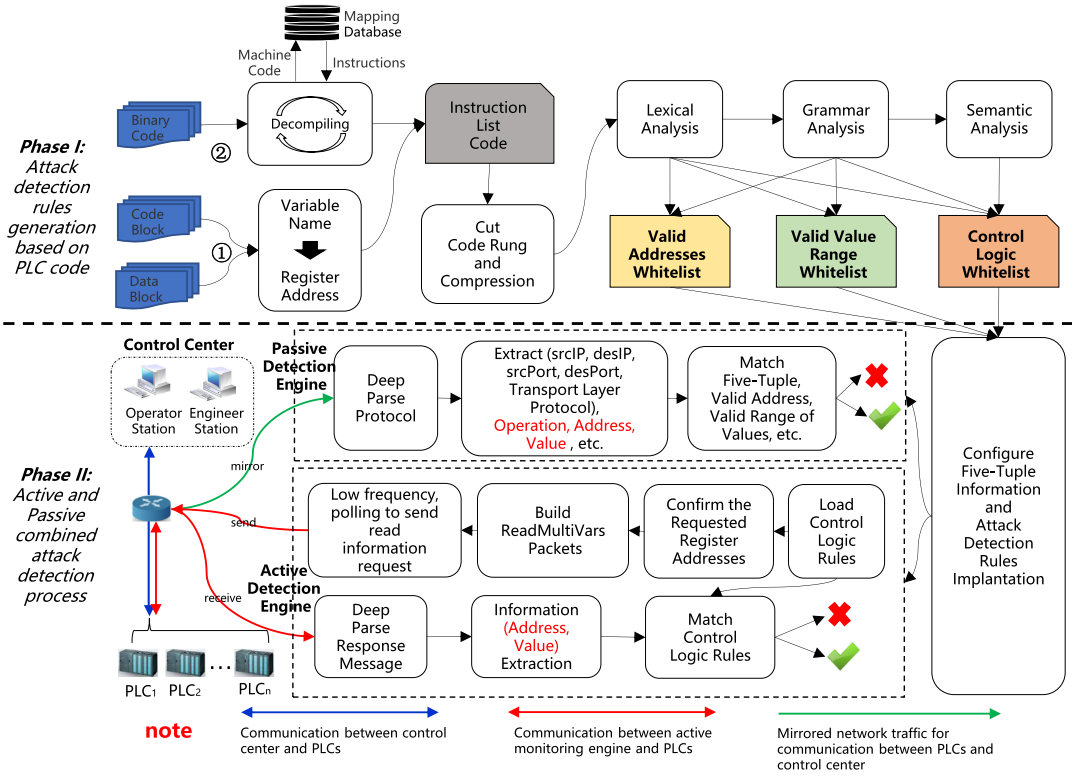


Fig. 8. A systematic approach to build ShadowPLCs.

DBX0.0) | *M32.0* → *Q0.0*, if *DB20.DBX0.6 & DB20.DBX0.0* or *DB20.DBX0.1 & ¬DB20.DBX0.0* or *M32.0* of any one of the results is 1, then *Q0.0* outputs 1, otherwise outputs 0. Therefore, *1 & 1* | *(0 & ¬0)* | *0* → *1* matches the control logic rule ①, which means normal. Similarly, in red, *1 & 0* | *(0 & ¬0)* → 0, 0 & (10 = 0) & *¬0* → *1* and *1 & 0* | *(0 & ¬1)* | *0* → *1* do not match the control logic rules ② and ③, which will alarm the PLC for possible control logic attacks, and can locate suspicious registers and code segments.

The active detection engine detects whether PLC is attacked in an active, low-disturbance and polling manner, and its core part is shown in Algorithm 2. First, establish communication connection with PLC (line 1); Then attack detection (line 2-line 16); Finally, when there is a task change requirement, the connection is disconnected and the task is destroyed (line 17, line 18). In the attack detection phase, the first step is to load the control logic rules, valid value range rules and valid address rules (line 4) to get register addresses and data type. Second, request packets are built according to the protocol format and sent to PLC (line 5, line 6). Line 7 and line 8 represent receiving the response data and parsing the protocol to extract register variable values; Finally,the valid value range and the control logic are checked for anomalies (line 9-line 15). Line 16 represents the setting of the request frequency to achieve low-disturbance to PLC.

5 Real-World Implementation

In order to verify the feasibility of our scheme ShadowPLCs, we implemented a prototype system with Siemens S7-300 series PLCs as a case study. Although this system was developed for a specific PLC manufacturer and model, based on this idea, It also can be modified to accommodate other PLC programming and protocols, which is an engineering problem. Fig. 8 presents a systematic approach to build ShadowPLCs. This proposed detection scheme comprising of two phases: 1) Attack detection rules generation based on PLC code. 2) Active and Passive combined attack detection process.

5.1 Phase I: Attack Detection Rules Generation Based on PLC Code

Program preprocessing. Before process control attacks detection, we need to extract detection rules first. The acquisition of PLC code is a preliminary preparation. As shown in Fig. 8, there are two possible ways:① Programmer directly provides the source code of PLC controller, which usually includes code blocks and data blocks. Replace the variable name written by programmer with the actual register address name, and get the data type of each address variable. In the actual application process, manual cooperation is required to export the source code from the project file. ② Get PLC source code directly by decompiling machine code. This solution reduces manual processing and is easy to automate. However, this method requires the ability to decompile. In this work, we adopted the way ② to obtain the source code.

Siemens PLC programs can be written in low-level STL (statement list), intermediate SCL (structural control language) or advanced LAD (ladder logic) diagrams. Regardless of the input program type, the application is internally converted to STL code by Step7 or TIA, which in turn is converted to MC7 code (lower layer representation of STL) and packaged into an opaque binary blob containing block data elements, metadata description and machine code. The

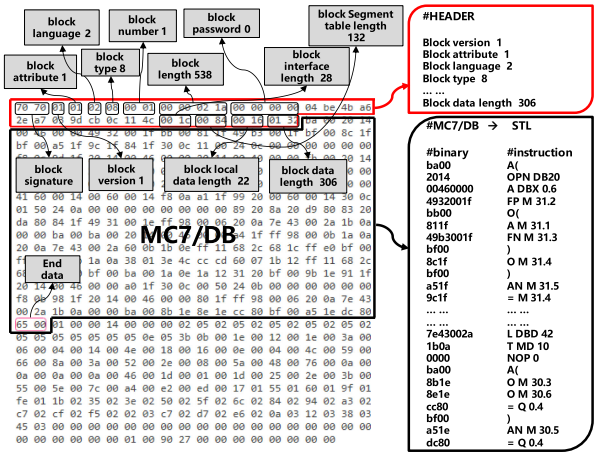


Fig. 9. MC7 code decompiled to STL code.

MC7 code is transferred to the CPU module during program download. Then, when CPU is set to run time, CPU executes MC7 code. In recent years, research work on decompilation of PLC code has appeared, such as ICSREF [22], CLIK [21], etc. We also have the same ability and reorganized the Siemens S7-300/400 MC7 code to the mapping database of the STL program and developed a decompilation module that can accurately decompile MC7 code to STL program.

Algorithm 2. Active Detection Algorithm

Input:

1. Control_Logic_WhiteList
2. Valid_Value_Range_WhiteList
3. Valid_Address_WhiteList

Output: Abnormal Alarm

```
ActiveDetection.Connect(PLC_IP, PLC_Port);
while pollFlag do
  for Control_Logic_Rule : Control_Logic_WhiteList do
    ActiveDetection.loadWhiteLists(Control_Logic_Rule,
    Valid_Value_Range_WhiteList,
    Valid_Address_WhiteList);
    ActiveDetection.buildReadMultiVars();
    ActiveDetection.send();
    result = ActiveDetection.receive();
    dataItemMap < address,value > = ActiveDetection.
    infoExtract(result);
    for (address, value) : dataItemMap do
      if (address, value) not in Valid_Value_Range_WhiteList
      then
        Alarm("Illegal Range of Value: PLC_IP, address,
        value, 'Value_Range"');
    if ActiveDetection.matchControlLogic (dataItemMap, Con-
    trol_Logic_WhiteList) is false
    then
      Alarm("Illegal Control Logic: PLC_IP, dataItem-
      Map, Control_Logic_Rule");
    time.sleep(t);
ActiveDetection.disconnect();
ActiveDetection.destroy();
```

The Table 1 shows the structure of known bytes. The header consists of 36 bytes and explains the basic

TABLE 1
Block Structure

Description	Offset	Bytes
Block signature	0	2
Block version	2	1
Block attribute	3	1
Block language	4	1
Block type	5	1
Block number	6	2
Block length	8	4
Block password	12	4
Block last modified date	16	6
Block interface last modified date	22	6
Block interface length	28	2
Block Segment table length	30	2
Block local data length	32	2
Block data length	34	2
MC7 code/ DB	36	x
Block signature	36 + x	1
Block number	37 + x	2
...	...	...

information of the current block. The data portion is a data block or code block, and the length of the portion is variable. The footer contains information about the parameters called out to the function. Fig. 9 shows an example of decompilation of binary data from an OB1 block into STL code, including a header segment and a program segment.

Detection rules generation. The goal of this phase is to automatically analyze STL programs of Siemens S7-300/400 series PLCs and generate whitelist rules such as valid addresses, valid value ranges and control logic. We use program flow analysis technology to abstract how the input, output and intermediate variables are manipulated inside the STL program and record the transformation operations of the process variables and the logical relationship between them. Finally, we implement the STL program automation analysis and detection rules generation module. Fig. 8 shows the main process of PLC program automation analysis and rules generation. We first preprocess the STL program text files of Siemens S7-300/400. The preprocessing mainly includes code segment cutting and program compression. Through the code segment cutting, the main program segment of STL can be divided; program compression can eliminate unnecessary instructions that do not affect the control logic, and reduce the burden of automatic analysis of the program and whitelist rules extraction. After the pre-processing, we conducted automatic analysis of the STL program. Through lexical analysis and grammar analysis, we solved the operation type, address type, comparison condition, etc., speculated the most likely data type and value range of process variables, and extracted the valid address whitelist and valid value range whitelist. Through semantic analysis, we can accurately extract the correlation between process variables and extract the control logic whitelist.

5.2 Phase II: Active and Passive Combined Attack Detection

In this phase, we will introduce the implementation details of process control attack detection. In this work, we use low-interference, active-passive combination of intrusion

detection methods, and does not affect the normal operation of PLCs. To the best of our knowledge, this is the first work to apply the combination of active and passive to ICS attack detection. As shown in Fig. 8, our process control attack detection consists of two parts: a passive detection engine and an active detection engine. Before the attack detection, the valid value range whitelists, the valid address whitelists and the control logic whitelists are implanted into the detection engines. The two detection engines run simultaneously and work together to detect the attack behavior of an ICS from different perspectives.

The deep parse of the ICS protocol is the basis of ICS network intrusion detection. S7 Communication (S7comm) is a dedicated communication protocol employed by Siemens S7-300/400 series PLCs. Control center uploads/downloads PLC program, read/write request of the control command, and response corresponding to PLCs are all carried out in accordance with this protocol. In Fig. 8, the active detection engine and the passive detection engine cooperate with each other to detect abnormal behavior from different perspectives. The passive detection engine first mirrors the network traffic flowing through the industrial switch, then parses S7comm protocol in depth, extracts *five-tuple, operations, addresses*, and *values*, and finally matches the valid address whitelist and valid value range whitelist to detect illegal links, illegal operations address and illegal values. In particular, the active detection engine reads the current value of the valid address space through low disturbance, slice, polling and active packet sending. The active detection engine first loads control logic rules and constructs a multi-address read data packet according to the S7comm protocol format. It then communicates with the PLC to send and receive data. Finally, verify that the control logic is abnormal.

6 Experimental Evaluation

In order to evaluate the detection effect of ShadowPLCs on industrial process control attacks, we conducted experiments using two Siemens S7-300 PLCs deployed on a gas pipeline network platform. First, we verified the accuracy of decompiling the MC7 code of the S7-300 PLC into the STL source code. Second, we analyzed and generated detection rules for multiple PLC programs from different sources. Owing to the lack of real-world process control attacks, we implemented the five attack scenarios discussed in Section 3 and simulated the effects of the attacks. We then compared ShadowPLCs with four other detection models for the detection performance of process control attacks. Finally, in order to demonstrate that our scheme has almost no impact on PLC, we monitored the PLC's operating cycle changes during the detection phase and tested the PLC's ability to handle network traffic through stress testing.

6.1 Experimental Environment

The gas pipeline network platform simulates the transfer of natural gas from a gate station to a residential building. As shown in Fig. 10, it consists of a gas pump that mimic the gate station, a section of pressure piping, three pressure transducers, four valves, and multiple sensors for monitoring pressure, flow rate, and temperature. Among them, the air pump supplies high-pressure gas to the physical

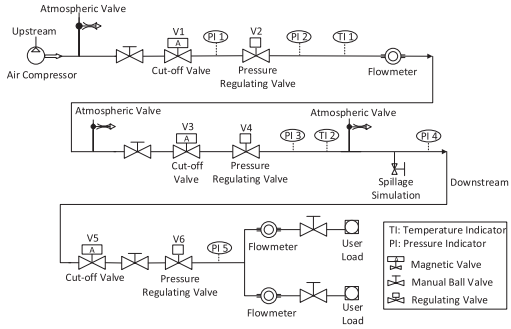


Fig. 10. Architecture of the gas pipeline network experimental platform.

environment; the gas pipeline is divided into three parts, namely high-pressure pipeline, medium-pressure pipeline and low-pressure pipeline, and each section of the pipeline controls the pressure drop of the gas through a pair of pressure converters and switches. Multiple sensors are responsible for collecting measurements such as temperature, flow rate, and pressure of the pipe. In addition, the scene also utilizes two Siemens S7-300 PLCs and a control center.

The physical process of the gas pipeline network platform is as follows: open the air pump to generate high pressure; open all valves in turn to construct the gas pipeline transportation path from the gate station to the residential quarter; open all pressure converters to reduce the gas pressure to the range that the residents can use. When the pipeline is damaged (implemented by the leakage valve simulation), gas leakage occurs. The two valves closest to the damage and the air pump are immediately closed. When the pipeline is repaired (also realized by the leakage valve simulation), the air pump is turned on and the pressure gradually returns to normal. In turn, the closed valve is opened successively to restore the communication of the pipeline. The automation of this series of physical processes is controlled by two Siemens S7-300 PLCs.

As shown in Fig. 11, the left part is the network topology diagram. We connect ShadowPLCs to the industrial switch between the control center and the PLCs, and configure its IP address to be in the same network segment as other devices, so that it can communicate normally with the PLCs. The upper right part is the on-site layout of the gas pipeline network platform, including the industrial site, the controller (PLC) and the control center. The lower right part is the connection status of each port of IDS probe equipment and industrial switch.

6.2 Decompilation Accuracy of PLC Code

We collected 106 source programs from our gas pipeline platform, github.com, and Shodan's exposed S7-300/400 PLCs. These procedures involve a variety of sensors, equipment controllers, and counters different physical processes (such as gas pipeline, traffic light, thermal power plants, fan operation, etc.). We have summarized that these program files contain a total of 5,093 instructions, the minimum is only 2 instructions, and the maximum is 254 instructions. We have compiled a mapping relation database of MC7 code and STL instruction list of Siemens S7-300/400 series PLCs. The database stores 1,853 mapping relationships, which contains all operation instructions and register addresses. We

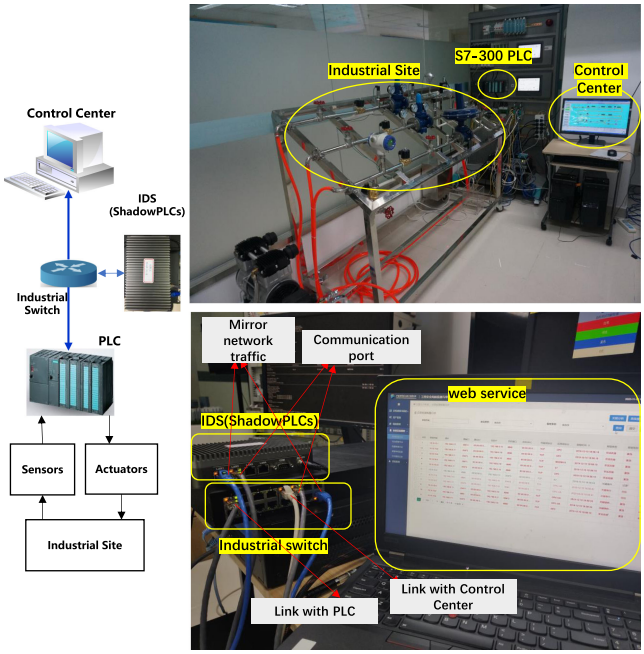


Fig. 11. Physical environment.

capture the network traffic that transfers programs to PLCs, extracting the binary program and its associated configuration and data. Use the mapping database to decompile the binary program and manually compare the results with the original program. The experimental results are summarized in Table 2, and it is concluded that our decompilation module can accurately decompile MC7 machine code into STL code and the accuracy reaches 100 percent.

6.3 Attack Detection Rule Generation

We analyze PLC codes from GitHub, Shodan, and our gas pipeline network platform and extract attack detection rules. PLC programs are not as complicated as other high-level language programs. A PLC program file is generally composed of multiple rungs and a rung consists of one or more instructions. We use rungs to represent the size of the program, and we specify that a rung usually extracts a control logic rule and multiple valid addresses and valid value range rules.

The data in Table 3 shows the number of valid address rules, valid value range rules, and control logic rules that can be generated by analyzing each real PLC program. All PLC programs from Github contain a total of 295 rungs, generating a total of 1,233 valid address rules, 121 valid range rules and 295 control logic rules. We stole some Shodan PLC programs published on the Internet, and counted that they contained 187 rungs. The valid address rules, valid value range rules, and control logic rules were 725, 96, and 187, respectively. Finally, we analyzed the PLC code carried by the gas pipeline platform that simulates a city gas pipeline network. It contains a total of 70 rungs, generating 152 valid address rules, 26 valid value range rules, and 70 control logic rules. Finally, we use the detection rules generated by the PLC code of the gas pipeline network platform as a benchmark to verify the detection effect of process control attacks.

6.4 Process Control Attack Detection Results

In Section 3, we investigated the research work of state-of-the-art process control attacks and summarized five attack scenarios. Because the disclosed process control attack methods or attack data sets are usually targeted at specific ICS scenarios, specific devices and specific processes, there is no practical process control attack that is universal. So in this work, we evaluate the detection effect of SHADOWPLCS by implementing 5 attack scenarios.

6.4.1 *Attacks Setting*

In this work, we discussed two process control attack methods against PLC, and summarized five attack scenarios

TABLE 2
Decompilation Accuracy

MC7 Code	Instruction	Description	Actual Number	Number of rescores	Accuracy
00 10	A I XXXX.0	Input register XXXX.0 "And" operation	33	33	100%
20	OPN DBXX	Open data block DBXX	21	21	100%
00 46	A DBX XXXX.6	DBX XXXX.6 "And" operation	5	5	100%
bb 00	O(	"Or" operation nesting begins	9	9	100%
00 00	NOP 0	Empty instruction	21	21	100%
8c	O M XX.4	Middle register M XX.4 "or" operation	6	6	100%
d9	=I XX.1 (I0.1 ..I127.1)/ =Q XX.1 (Q0.1 ..Q127.1)	"Output" operation:= I XX.0 (I0.0..I127.0) if XX = 00h..7Fh / = Q XX.0 (Q0.0..Q127.0) if XX = 80h..FFh	15	15	100%
f8	A T XX (T0 ..T255)	Timer T XX "and" operation	7	7	100%
a5	AN M XX.5	Middle register "And Not" operation	1	1	100%
30 0C	L S5T# XXXX (S5T# literal, S5T#0MS-S5T#2H46M30S)	Set the timer duration	7	7	100%
49 32	FP M XXXX.2	M XXXX.2 Pulse rising edge	2	2	100%
31 20	>R	Compare instruction, if accumulator 1 is greater than accumulator 2, the result is 1, otherwise 0	3	3	100%
7e 43	L DBD XXXX (DBD 0..DBD 65535)	Load DBD XXXX into the accumulator 1	1	1	100%
1b	T MD XX (MW0 .. MW255)	The current operation result is transferred to MD XX	3	3	100%
ff 98	JNB XXXX (XXXX = relative address)	Jump instruction	13	13	100%
60 07	*R	Multiply the contents of accumulator 1 and accumulator 2	2	2	100%
...	...	...	...	...	100%

TABLE 3
Rule Count for Attack Detection in PLC Code

Program Source	Number of Rungs			Number of Detection Rules		
	Min	Max	Total	Valid Address	Valid Value Range	Control Logic
github.com	1	45	295	1,233	121	295
Shodan	2	63	187	725	96	187
Gas pipeline platform	2	23	70	152	26	70

taken into account these two attack methods. Owing to the lack of readily available attacks, we implemented these five attack scenarios. We implement *illegal address access attack* by randomly reading the input and output registers, timers, counters, middle registers, data blocks, etc. of the PLC; by injecting incorrect data into important registers that may cause industrial process disturbances, we implement *malicious data injection attack*, Such as forcing a write operation to a register address to overwrite the current value of the register; the implementation method of *Configuration tampering attack* is to tamper with the PLC configuration information, such as timers, counters, PID parameters, etc.; the implementation method of *control logic infection attack* and *control program replacement attack* is similar. All use the tool snap7-full-1.4.0 to download malicious control code to PLCs and respond to historical data to confuse the control center during these attacks. *Control logic infection attack* is designed to infect the original code, such as replace the value in the I/O register with a controllable value somewhere in memory to control the state of the actuator, insert/delete instructions/rungs, replace operators in equations, modify control flow determinants, etc; *control program replacement attack* is to implant the malicious control code, overwrite the original program, and try to let the PLC execute it.

6.4.2 Attack Detection Results

We launched the above attacks to evaluate the detection capability of ShadowPLCs. Table 4 presents the results of attack detection. The passive detection engine can successfully detect *illegal address access attack* and *configuration tampering attack* by monitoring network traffic. These two attacks are difficult to escape the detection of valid address rules and valid value range rules. *Malicious data injection attack* can successfully bypass the detection of valid address rules. *Control logic infection attack* and *control program replacement attack* can intercept and replay historical data after launch, so it is often difficult for passive detection engines to detect such covert attacks. The active detection engine can successfully detect *malicious data injection attack*, *configuration tampering attack*, *control logic infection attack* and *control program replacement attack*, because these four attacks will destroy the normal control logic and cause the logical relationship between process variables to be illegal. The active detection engine detects process control attacks by actively communicating with the PLC to map its memory. This active detection method makes it difficult to hide the attack behavior, because it requires attackers to have the ability to deeply understand the real industrial process and be able to feed back network data that conforms to the control logic at any time (for example, by modifying PLC firmware and tampering with network communication modules) .

TABLE 4
Experimental Results of Process Control Attack Detection

Attack Type	False Instruction Injection			Control Logic Injection	
	Illegal Address Access Attack	Malicious Data Injection Attack	Configuration Tampering Attack	Control Logic Infection Attack	Control Program Replacement Attack
Passive Detection Engine	✓	X	✓	X	X
Active Detection Engine	X	✓	✓	✓	✓
ShadowPLCs	✓	✓	✓	✓	✓

"✓" indicates successful detection of attacks.
"x" indicates that no attack was detected.

6.4.3 Comparison With Other Anomaly Detection Models

To convincingly demonstrate the effectiveness of our detection scheme, we also compared its performance with other commonly used ICS anomaly detection models. Specifically, we compared with representative models for detecting industrial process control attacks. They are Auto-Regressive (AR) models [19], SRID [46], LSTM [17], and S-IDS [10]. The network topology between control center and PLCs is usually static, and the routine polling of I/O devices is somewhat predictable. AR and LSTM learn and model historical network traffic sequences to predict future states. S-IDS establishes a state transition diagram model for network traffic sequences. The prerequisite of SRID is to know the relationship between process variables and establish a relationship graph model.

The number of times we launched *illegal address access attack, malicious data injection attack, configuration tampering attack, control logic infection attack, and control program replacement attack* were 500, 500, 500, 150 and 150 respectively. We recorded the time and detailed information log of each attack (such as: forced register M0.0 to write "1"; tampered program "A I0.0" to "AN I0.0", etc.). We also recorded the detection results, including detection time and detection details (such as: illegal "write" operation, M0.0 is set to "1"; abnormal control logic "& ¬ I0.0", etc.). Engineers compare attack logs and detection logs to determine whether there are false positives or false negatives. The detailed results are summarized in Table 5. In addition to SRID, other detection methods have good detection effects on *illegal address access attack* and *configuration tampering attack*. This is because these attack scenarios will not destroy the relationship between process variables, however, SRID mainly alerts for abnormal phenomena that occur after the relationship

TABLE 5
Performance Comparison With Other Anomaly Detection Models

Attack	Number of Attacks	Model	Detection Accuracy	Number of False Alarms/hour
Illegal Address Access Attack	500	ShadowPLCs	**100%**	7
		AR	**100%**	37
		SRID	0%	6
		LSTM	**100%**	15
		S-IDS	**100%**	11
Malicious Data Injection Attack	500	ShadowPLCs	**93.4%**	3
		AR	70.2%	45
		SRID	90.4%	12
		LSTM	76.8%	21
		S-IDS	88.0%	17
Configuration Tampering Attack	500	ShadowPLCs	**100%**	5
		AR	95.0%	34
		SRID	47.2%	9
		LSTM	91.6%	36
		S-IDS	**100%**	17
Control Logic Infection Attack	150	ShadowPLCs	**94.7%**	9
		AR	29.3%	29
		SRID	78.7%	7
		LSTM	13.3%	32
		S-IDS	21.3%	19
Control Program Replacement Attack	150	ShadowPLCs	**97.3%**	6
		AR	28.7%	27
		SRID	87.3%	10
		LSTM	14.0%	31
		S-IDS	24.7%	14

between process variables changes. ShadowPLCs, SRID and S-IDS have good detection accuracy for *malicious data injection attacks*, but the detection results of the prediction models AR and LSTM are relatively poor. ShadowPLCs has a huge advantage for relatively covert attacks such as *control logic infection attack* and *control program replacement attack*. In particular, the detection accuracy of *control logic infection attack* and *control program replacement attack* can reach 94.7 and 97.3 percent, respectively, while the best detection results in other approaches are 78.7 and 87.3 percent, respectively. These attack scenarios are relatively hidden, and their identification is very difficult for detection methods that rely heavily on network traffic modeling, so the detection efficiency of AR, LSTM and S-IDS is not good. SRID has a reliable detection effect on covert attacks, but its passive monitoring of network traffic can easily cause process variables to be out of synchronization, so the detection effect is worse than our scheme. Finally, we lasted 24 hours for each attack scenario, and counted the average number of alarms per hour except for successfully detected attacks, that is, the number of false alarms (identifying normal traffic as abnormal). It can be seen from Table 5 that ShadowPLCs has the least false alarm per hour, followed by SRID. However, the prediction models AR and LSTM have more forecasts for each attack scenario. The number of false positives of S-IDS is also relatively high. Experiments have proved that our scheme is significantly better than other detection methods in terms of accuracy and false positives.

6.5 Measuring PLC Cycle Times Under Active Detection

The computing performance of PLCs is not as powerful as traditional PCs. Malicious network traffic may affect the electrical aspects of PLCs, causing controlled processes to be disrupted or even completely stopped working. Therefore, we need to pay attention to whether the active detection method will affect the electrical aspects of PLC. Reference [34] studied the ability of the controlled process of multiple brands of PLCs to withstand flooding attacks. The difference is that our active detection is based on the ICS protocol format, using low-interference, polling to request PLC register data, and then performing data processing and attack behavior detection at the remote end.

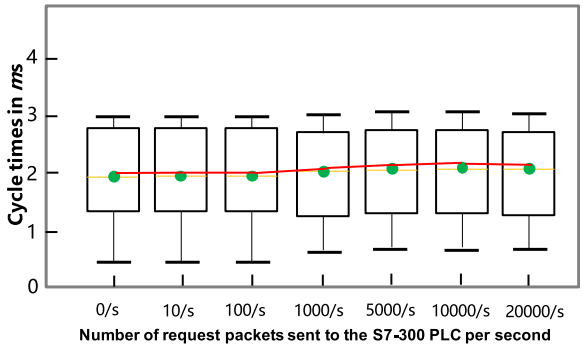


Fig. 12. Influence of network traffic on S7-300 PLC cycle.

In order to demonstrate that our detection scheme has little effect on the controlled process of PLC, we set different request frequencies in the active detection process to observe the changes of the PLC cycle times. In order to avoid the current limitation of the router, we directly connect the PLC with a network cable and send 10, 100, 1000, 5,000, 10,000, 20,000 request packets to the PLC for each second. As shown in Fig. 12, without any external interference, the cycle period of the PLC of the gas pipeline network platform is between 0-3 ms and most of them are maintained at about 2 ms. When the S7-300 PLC was subjected to external interference to varying degrees, we found that its cycle times were hardly affected. By analyzing the above phenomenon, we suspect that the Siemens S7-300 PLC may have internal defense against denial of service attacks, and there are restrictions on the network traffic handled. We tested the case of 5000 packet requests per second, and found that there was no packet loss through statistical data response information. This means that Siemens S7-300 PLC has the capacity to process at least 5,000 packets per second without affecting its normal operation.

Different manufacturers and models of PLCs usually have different processing capabilities for network traffic. Therefore, the setting of the frequency of data requests for active detection is also different. Almost all PLC programming software will provide a program remote monitoring function. When custom IDS development is performed for different PLCs, the frequency of active requests can be set to be the same as the frequency of interaction between the programming software and the PLC when the monitoring function is started. This is a low-disturbance detection method. It only reads the PLC register data at a low frequency, and most of the calculation and detection work is in a remote high-performance IDS device.

7 Discussion

7.1 Advantage

The advantages of ShadowPLCs are mainly four aspects, as follows:

① Deep understanding of industrial processes. PLCs automatic controls actual industrial processes through PLC programs. Therefore, detection rules generated by PLC codes are more comprehensive than modelling and rule extraction based on network traffic, and they can fully reflect the real controlled processes. At the same time, the use of procedural rules to detect industrial process control attacks is more effective.

② No human effort. ShadowPLCs can automatically and comprehensively generate authoritative detection rules through automatic analysis of PLC control codes, without the need for humans to rely on experience and manual configuration and modeling.

③ Process control attacks are difficult to hide. Our scheme is combined of active and passive methods to detect attack behaviors from different perspectives. Since we deeply understand the controlled process and actively map the corresponding PLC register variables, this makes it more difficult for attackers to covertly attacks. To escape detection, attackers need to understand the real industrial process and be able to respond in real time to fake data that IDS complies with the controlled process. This usually requires going deep into PLC, attacking the PLC firmware and modifying its network processing module, rather than simply avoiding detection by means of middlemen.

④ Wide range of applications. According to the detection scheme proposed by us, we can customize the development of different PLC devices, and can actually put it into practice to make substantial defense against ICS.

7.2 Limitation

Any kind of intrusion detection scheme will have its limitations, and ShadowPLCs is no exception.

① Although almost all PLC languages follow the IEC 61131-3 standard, there may be some differences in the programming languages of different brands and models of PLCs. Therefore, when generating detection rules by analyzing PLC codes, custom development for different types of PLCs is mandatory.

② ShadowPLCs does not apply in respect of encrypted private protocols. Whether it is passive or active detection, the analysis of the ICS protocol is the most basic and necessary. IDS engineers need to be able to understand the protocol format and communication methods in depth, and be able to extract important information and build data packets to communicate with PLC based on the ICS protocol.

③ The active detection method cannot perform attack detection on those PLC devices with access rights set. However, a large number of PLC devices currently do not restrict the remote reading of PLC register variables.

8 Related Work

8.1 PLC Program Analysis

Current analytical methods for PLC programs mainly include model detection [2], [6], [15], static analysis [36], [42], theorem proving [7], [20], symbolic execution [31], and so on. The static analysis of the PLC program utilizes techniques such as control flow and data flow analysis, and is mainly applied to the verification of the program syntax. The literature [7], [20] uses the method of theorem to prove the PLC program into a mathematical formula, and uses the theorem prover to analyze and verify. In [31], the method of symbolic execution is used to analyze the path constraint of the program, and the specific value of the target constraint is calculated by the constraint solution, which eliminates the access of the unreachable path and effectively solves the problem of state explosion in the detection of the PLC program model. The above work has laid a good foundation for the verification and analysis of PLC programs. Unlike them, we do not pay attention to whether the program itself is abnormal or not. Instead, we generate detection rules by analyzing the programs and apply them to ICS network detection.

8.2 ICS Attack Detection

With the advent of the industrial Internet era, network intrusion detection has begun to be applied in the ICS field. Since the outbreak of the "Stuxnet" incident in 2010, the anomaly detection of ICS has attracted great attention from academia and industry, and has become a very popular research field. In addition to traditional cyber attacks (such as Dos, scanning, IP spoofing, etc.), the attack behavior of ICS is more about the destruction of industrial process control, which is the focus of ICS anomaly detection research. We analyzed the research status in this field in recent years, and concluded that the ICS process control attack detection schemes are mainly divided into three categories: 1) Attack detection based ICS protocol specification. 2) Detect attacks by modeling ICS network traffic. 3) Detect attacks by modeling physical processes. The specific details are as follows.

8.2.1 Attack Detection-Based ICS Protocol Specification

Early ICSs are relatively closed. The design of industrial control protocol did not fully consider security, so there are plenty of security risks. With the gradual liberalization of ICSs, these protocols all present more security vulnerabilities, such as the lack of data encryption and identity authentication mechanisms, and are vulnerable to attacks such as stealing, tampering, forging message data, and injecting malicious code. In ICS, researchers analyzed the ICS protocol to form a protocol specification model. Protocol specifications typically define the message format and the communication modes permitted by the protocol.

Cheung *et al.*[13] constructed a protocol specification model for the legal value of different fields and the relationship between different fields in a data packet to describe the expected or acceptable behavior of the system. Morris *et al.* [33] rely on protocol specification whitelist and business logic whitelist to generate Snort detection rules, which can effectively identify intrusion behaviors such as denial of service, command injection, and response injection. Yang *et al.* [49] established a single-domain information model and a multi-domain information association model according to the rules designed by the IEC 60870-5-104 protocol to construct a model describing the expected behavior of a specific protocol. On the basis of [49], the literature [48] proposes a protocol state IDS using the deep packet inspection method, and improves the network security of the SCADA system

by establishing a finite state machine model for the IEC 60870-5-104 protocol.

Most of the IDS technology based on the ICS protocol specification has the advantages of simple model, simple malware such as malformed messages, but poor detection of process control attacks.

8.2.2 Detect Attacks by Modeling ICS Network Traffic

ICS network topology is single, the number of applications is small, and there is not much manual intervention. Under normal circumstances, the traffic characteristics of an ICS remain relatively stable. Therefore, modeling ICS network traffic to detect attacks is prevalent.

Considering the high periodicity and correlation of ICS traffic, Yoon *et al.*[51] used dynamic Bayesian networks and probabilistic suffix trees to establish communication models for Modbus protocol traffic. S-IDS [10], [11] extracts the operation, time, transition conditions, etc. of the protocol, host log entity and running variables, and then models it with discrete Markov chains. Literature [17], [28] developed a multi-level anomaly detection framework based on network packet signature and deep learning. They contain a neural network-based detection module that can solve the time dependence between consecutive packets. Due to asynchronous scheduling, communication is sometimes multiplexed, which makes the above-mentioned detection model huge in size and high in false positives. Kleinmann *et al.* [23] proposed an automated construction method based on Deterministic Finite Automata (DFA) for multi-threaded ICS anomaly detection model. Compared with the single DFA model, this scheme reduces the size of the detection model and reduces the false alarm rate. Rohit Negi *et al.* [37] consider fake data injection or inflight data tampering at the industrial control communication layer, as a threat model, and designs a lightweight PLC real-time detection and prevention data tampering solution.

8.2.3 Detect Attacks by Modeling Physical Processes

Some researchers have attempted to model the physical processes of industrial facilities. For example, Hadziosmanovi *et al.* [19] modeled the semantics of process variables, and Wang *et al.* [46] used a graph-based detection scheme to detect fake data injection attacks, Krotofil *et al.* [26] proposed a set of algorithms for lightweight real-time spoofing sensor signals for field device microcontrollers. The correlation entropy is utilized to detect fraudulent measurement data in the form of credibility and consistency check. Yang *et al.* [47] modeled the behavior of legitimate PLC load program runtimes and used runtime behavior monitoring in the PLC firmware to detect load attacks. Urbina *et al.* [45] studied the physics-based attack detection method for control systems, developed an adaptive adversarial model and a new metric to measure the impact of stealth attacks. In addition, some researchers [7], [20], [31] formalized the PLC program, and converted the PLC program into a mathematical model by the theorem proof method to prove the security performance of the PLC program.

In summary, most of the current network intrusion detection work for ICS is tantamount to model the protocol or network traffic itself. A more in-depth study is to passively monitor historical process variables for future predictive evaluation. These works have no actual understanding of the actual physical control process, nor are they in contact with actual control devices. Therefore, these detection methods are limited by the network traffic itself, the detection surface is narrow, the false alarm rate is high, and it is easy to be bypassed (such as replay attack). In contrast, we conducted exploratory research and developed tools for automated analysis of PLC control programs to gain insight into real-world physical processes. It was able to extract three types of whitelisting rules: valid address rules, valid value range rules, and control logic rules. Finally, we are combined with passive and passive to remotely detect process control attacks in real time with little impact on the normal operation of the PLC.

9 Conclusion

We present a novel scheme for remotely detecting industrial process control attacks in real-time, called ShadowPLCs. It combines active and passive methods, and as an example, it implements a prototype system with Siemens S7-300 PLCs. Specifically, the system automatically analyzes the PLC program and generates attack detection rules, including valid addresses, valid value ranges, and control logic rules. Then, by actively communicating with the PLC and passively monitoring network traffic, the system can perform real-time detection of process control attacks from different perspectives. In the end, we evaluated our detection scheme using two Siemens S7-300 PLCs deployed on the gas pipeline network platform. Experiments show that the scheme can accurately detect hidden process control attacks in real-time without affecting the normal operation of the PLC. Compared with the other four representative detection models, our scheme has better detection performance.

Acknowledgments

This work was supported by Guangdong Province Key Area R&D Program of China (Grant No. 2019B010137004), Key Program of National Natural Science Foundation of China (Grant No. U1766215), Science and technology project of State Grid Corporation of China(Grant No. 521304190004), Natural Sciences and Engineering Research Council of Canada(NSERC), Canada, and National Natural Science Foundation of China under (Grant No. 61802016).

References

[1] Dell threat report claims 100 percent increase in SCADA attacks, 2015. [Online]. Available: https://threatpost.com/dell-threat-report-claims-100-percent-increase-insca da-attacks/112241

[2] B. F. Adiego *et al.*, "Applying model checking to industrial-sized PLC programs," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1400–1410, Dec. 2015.

[3] C. M. Ahmed *et al.*, "NoisePrint: Attack detection using sensor and process noise fingerprint in cyber physical systems," in *Proc. Asia Conf. Comput. Commun. Secur.*, 2018, pp. 483–497.

[4] I. Ahmed, V. Roussev, W. Johnson, S. Senthivel, and S. Sudhakaran, "A scada system testbed for cybersecurity and forensic research and pedagogy," in *Proc. 2nd Annu. Ind. Control Syst. Secur. Workshop*, 2016, pp. 1–9.

[5] D. Alert, ICS-CERT, "Cyber-attack against ukrainian critical infrastructure," *Cybersecurity Infrastruct. Secur. Agency*, Washington, DC, USA, Tech. Rep. ICS Alert (IR-ALERT-H-16-056-01), 2016.

[6] S. Biallas, J. Brauer, and S. Kowalewski, "Arcade. PLC: A verification platform for programmable logic controllers," in *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, 2012, pp. 338–341.

[7] S. O. Biha, "A formal semantics of PLC programs in Coq," in *Proc. IEEE 35th Annu. Comput. Softw. Appl. Conf.*, 2011, pp. 118–127.

[8] S. Bologna, B. Hämmerli, D. Gritzalis, and S. Wolthusen, *Critical Information Infrastructure Security*. Springer, 2011. [Online]. Available: https://link.springer.com/content/pdf/10.1007/978-3-642-41476-3.pdf

[9] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," in *Proc. 3rd Conf. Hot Topics Secur.*, 2008, Art. no. 6.

[10] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware intrusion detection in industrial control systems," in *Proc. 1st ACM Workshop Cyber-Physical Syst. Secur.*, 2015, pp. 13–24.

[11] M. Caselli, E. Zambon, J. Petit, and F. Kargl, "Modeling message sequences for intrusion detection in industrial control systems," in *Proc. Int. Conf. Critical Infrastructure Protection*, 2015, pp. 49–71.

[12] ICS Cert, "Incident response/vulnerability coordination," 2014. [Online]. Available: http://www.securityweek.com/critical-infrastructureincidents-increased-2015-ics-cert

[13] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using model-based intrusion detection for scada networks," in *Proc. SCADA Secur. Sci. Symp.*, 2007, vol. 46, pp. 1–12.

[14] International electrotechnical commission, *61131-3: Programmable controllers–Part 3: Programming languages*. Int. Standard, 2nd ed., International Electrotechnical Commission, Geneva, vol. 1, pp. 2003, 2003.

[15] D. Darvas, E. B. Vinuela, and B. F. Adiego, "PLCverif: A tool to verify PLC programs based on model checking techniques," in *Proc. 15th Int. Conf. Accel. Large Exp. Phys. Control Syst.*, pp. 911–914, 2015.

[16] N. Falliere, L. O. Murchu, and E. Chien, "W32. Stuxnet dossier," White Paper, Symantec Corp., Security Response, vol. 5, no. 6, 2011, Art. no. 29.

[17] C. Feng, T. Li, and D. Chana, "Multi-level anomaly detection in industrial control systems via package signatures and LSTM networks," in *Proc. 47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2017, pp. 261–272.

[18] E. Goetz and S. Shenoi, Eds., *Critical Infrastructure Protection*, vol. 253. Berlin, Germany: Springer, 2007.

[19] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, 2014, pp. 126–135.

[20] R. Huuck, "Semantics and analysis of instruction list programs," *Electron. Notes Theor. Comput. Sci.*, vol. 115, pp. 3–18, 2005.

[21] S. Kalle, N. Ameen, H. Yoo, and I. Ahmed, "CLIK on PLCs! Attacking control logic with decompilation and virtual PLC," in *Proc. Binary Anal. Res. (BAR) Workshop, Netw. Distrib. Syst. Secur. Symp.*, 2019, pp. 1–12.

[22] A. Keliris and M. Maniatakos, "ICSREF: A framework for automated reverse engineering of industrial control systems binaries," *Symp. Netw. Distrib. Syst. Secur.*, pp. 1–16, 2019.

[23] A. Kleinmann and A. Wool, "Automatic construction of statechart-based anomaly detection models for multi-threaded industrial control systems," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 4, 2017, Art. no. 55.

[24] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Rockland, MA, USA: Syngress, 2014.

[25] M. Krotofil, A. A. Cárdenas, B. Manning, and J. Larsen, "CPS: Driving cyber-physical systems to unsafe operating conditions by timing dos attacks on sensor signals," in *Proc. 30th Annu. Comput. Secur. Appl. Conf.*, 2014, pp. 146–155.

[26] M. Krotofil, J. Larsen, and D. Gollmann, "The process matters: Ensuring data veracity in cyber-physical systems," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Secur.*, 2015, pp. 133–144.

[27] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security Privacy*, vol. 9, no. 3, pp. 49–51, May/Jun. 2011.

[28] J. Liu, L. Yin, Y. Hu, S. Lv, and L. Sun, "A novel intrusion detection algorithm for industrial control systems based on CNN and process state transition," in *Proc. IEEE 37th Int. Perform. Comput. Commun. Conf.*, 2018, pp. 1–8.

[29] S. McLaughlin and P. McDaniel, "SABOT: Specification-based payload generation for programmable logic controllers," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 439–449.

[30] S. E. McLaughlin, "On dynamic malware payloads aimed at programmable logic controllers," in *Proc. 6th USENIX Conf. Hot Topics Secur.*, 2011, Art. no. 10.

[31] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A trusted safety verifier for process controller code," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, vol. 14, 2014, pp. 1–15.

[32] B. Meixell and E. Forner, "Out of control: Demonstrating scada exploitation," *Black Hat*, vol. 1, pp. 1–7, 2013.

[33] T. Morris, R. Vaughn, and Y. Dandass, "A retrofit network intrusion detection system for MODBUS RTU and ASCII industrial control systems," in *Proc. 45th Hawaii Int. Conf. Syst. Sci.*, 2012, pp. 2338–2345.

[34] M. Niedermaier *et al.*, "You snooze, you lose: Measuring PLC cycle times under attacks," in *Proc. 12th USENIX Workshop Offensive Technol.*, 2018, Art. no. 12.

[35] Y. Peng, C. Jiang, F. Xie, Z. Dai, Q. Xiong, and Y. Gao, "Industrial control system cybersecurity research," *J. Tsinghua Univ. Sci. Technol.*, vol. 52, no. 10, pp. 1396–1408, 2012.

[36] H. Prähofer, F. Angerer, R. Ramler, H. Lacheiner, and F. Grillenberger, "Opportunities and challenges of static code analysis of IEC 61131–3 programs," in *Proc. IEEE 17th Int. Conf. Emerg. Technol. Factory Autom.*, 2012, pp. 1–8.

[37] A. Handa, U. Ayyangar, S. K. Shukla, R. Negi, and A. Dutta, "Intrusion detection & prevention in programmable logic controllers: A model-driven approach," in *Proc. IEEE Int. Conf. Ind. Cyber-Physical Syst.*, 2020, pp. 1–8.

[38] J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell, "A quantitative evaluation of the target selection of havex ICS malware plugin," in *Proc. Ind. Control Syst. Secur. Workshop*, 2015, pp. 1–5.

[39] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," in *Proc. 52nd ACM/EDAC/IEEE Des. Autom. Conf.*, 2015, pp. 1–6.

[40] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, and V. Roussev, "Denial of engineering operations attacks in industrial control systems," in *Proc. 8th ACM Conf. Data Appl. Secur. Privacy*, 2018, pp. 319–329.

[41] K. Serebryany, D. Bruening, A. Potapenko, and D. Vyukov, "AddressSanitizer: A fast address sanity checker," in *Proc. USENIX Annu. Tech. Conf.*, 2012, pp. 309–318.

[42] S. Stattelmann, S. Biallas, B. Schlich, and S. Kowalewski, "Applying static code analysis on industrial controller code," in *Proc. IEEE Emerg. Technol. Factory Autom.*, 2014, pp. 1–4.

[43] K. A. Stouffer, J. A. Falco, and K. A. Scarfone, "SP 800–82. Guide to industrial control systems (ICS) security: Supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations such as programmable logic controllers (PLC)," NIST, Gaithersburg, Nat. Inst. Standards Technol., MD, USA, Tech. Rep., 2011.

[44] G. Tzokatziou, L. Maglaras, and H. Janicke, "Insecure by design: Using human interface devices to exploit SCADA systems," in *Proc. 3rd Int. Symp. ICS SCADA Cyber Secur. Res.*, 2015, pp. 103–106.

[45] D. I. Urbina *et al.*, "Limiting the impact of stealthy attacks on industrial control systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1092–1105.

[46] Y. Wang, Z. Xu, J. Zhang, L. Xu, H. Wang, and G. Gu, "SRID: State relation based intrusion detection for false data injection attacks in SCADA," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2014, pp. 401–418.

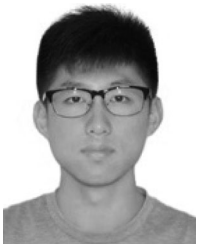
[47] H. Yang, L. Cheng, and M. C. Chuah, "Detecting payload attacks on programmable logic controllers (PLCs)," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2018, pp. 1–9.

[48] Y. Yang, K. McLaughlin, S. Sezer, Y. B. Yuan, and W. Huang, "Stateful intrusion detection for IEC 60870–5-104 SCADA security," in *Proc. IEEE PES General Meeting | Conf. Exposition*, 2014, pp. 1–5.

[49] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, B. Pranggono, and H. F. Wang, "Intrusion detection system for IEC 60870–5-104 based SCADA networks," in *Proc. IEEE Power Energy Soc. General Meeting*, 2013, pp. 1–5.

[50] H. Yoo and I. Ahmed, "Control logic injection attacks on industrial control systems," in *Proc. IFIP Int. Conf. ICT Syst. Secur. Privacy Protection*, 2019, pp. 33–48.

[51] M.-K. Yoon and G. F. Ciocarlie, "Communication pattern monitoring: Improving the utility of anomaly detection for industrial control systems," in *Proc. NDSS Workshop Secur. Emerg. Netw. Technol.*, 2014, pp. 1–10.



Junjiao Liu (Student Member, IEEE) received the BE degree from Beijing Jiaotong University, China, in 2016. He is currently working toward the PhD degree with the Institute of Information Engineering, Chinese Academy of Sciences, China. His research interests include the security of the Internet of Things and industrial control systems.

Xiaodong Lin (Fellow, IEEE) received the PhD degree in information engineering from the Beijing University of Posts and Telecommunications, China, and the PhD degree in electrical and computer engineering from the University of Waterloo, Canada. He is currently an associate professor at the School of Computer Science, University of Guelph, Canada. His research interests include wireless network security, applied cryptography, computer forensics, and software security.

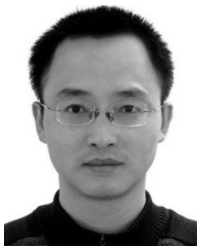
Xin Chen (Member, IEEE) received the MS degree from the School of Electrical Engineering, Zhengzhou University, China, in 2016. He has been an engineer with the Institute of Information Engineering, Chinese Academy of Sciences, China, since July 2016. His research interest includes industrial control system intrusion detection.

Hui Wen received the PhD degree from the University of Chinese Academy of Sciences, Beijing, China, in 2016. He is an assistant professor with Institute of Information Engineering, Chinese Academy of Sciences, China. His research interests are related to malware anaysis and IoT security.

Hong Li received the BS degree in computer science from Xi'an Jiao Tong University, China, in 2011, and the PhD degree in cyber security from the University of Chinese Academy of Sciences, China, in 2017. Since 2017, he is an associate professor at the Institute of Information Engineering, Chinese Academy of Sciences, China. His current research interests include IoT security, ICS security, and blockchain.

Yan Hu received the BS degree in automation from Xi'an Jiaotong University, Xi'an, Shanxi, China, in 2011, and the PhD degree in computer science from the University of Chinese Academy of Sciences, Beijing, China, in 2017. Since 2017, she has been an assistant professor with the University of Science and Technology Beijing, China. Her main research interests include smart city, IoT security, and ICS security.

Jiawei Sun is currently working toward the PhD degree at the Institute of Information Engineering, Chinese Academy of Sciences, China. His main research interests include network and information security, in particular DNS data mining.



Zhiqiang Shi is currently a professor at the Institute of Information Engineering, Chinese Academy of Sciences (CAS), China, His research interests include critical infrastructure security, software security, embedded security, and machine learning.

Limin Sun received the PhD degree from the National University of Defense Technology, China. He is currently a professor with the Institute of Information Engineering, Chinese Academy of Sciences, China. He is also the secretary general of the Select Committee of CWSN and the director of the Beijing Key Laboratory of IoT Information Security Technology. His main research interests include the Internet-of-Things (IoT) security, and industrial control system security. He is an editor of the *Journal of Computer Science* and the *Journal of Computer Applications*.