Control Logic Injection Attacks on Industrial Control Systems

Hyunguk Yoo[1] and Irfan Ahmed[1,2]

[1] University of New Orleans, New Orleans LA 70148, USA
hyoo1@uno.edu
[2] Virginia Commonwealth University, Richmond VA 23221, USA
iahmed3@vcu.edu

Abstract. Remote control-logic injection attacks on programmable logic controllers (PLCs) impose critical threats to industrial control system (ICS) environments. For instance, Stuxnet infects the control logic of a Siemens S7-300 PLC to sabotage nuclear plants. Several control logic injection attacks have been studied in the past. However, they focus on the development and infection of PLC control logic and do not consider the stealthy methods of transferring the logic to a PLC over the network. This paper is the first effort to explore the packet manipulation of control logic to achieve stealthiness without modifying PLC firmware to support new (obfuscation) functionality. It presents two new control logic injection attacks: 1) Data Execution and 2) Fragmentation and Noise Padding. *Data Execution* attack subverts signatures (based-on packet-header fields) by transferring control logic to the data blocks of a PLC and then, changes the PLC's system control flow to execute the attacker's logic. *Fragmentation and Noise Padding* attack subverts deep packet inspection (DPI) by appending a sequence of padding bytes in control logic packets while keeping the size of the attacker's logic in packet payloads significantly small. We implement the attacks on two industry-scale PLCs of different vendors and demonstrate that these attacks can subvert intrusion detection methods successfully, such as signature-based intrusion detection and Anagram-based DPI. We also release the training and attack datasets to facilitate research in this direction.

Keywords: Control logic · Code Injection Attack · Programmable Logic Controller · Ladder Logic · Critical Infrastructure.

1 Introduction

Programmable logic controllers (PLCs) in industrial control systems (ICS) are directly connected to physical processes such as wastewater treatment plants, gas pipelines, and electrical power grids. They are equipped with control logic that define how to control and monitor the behavior of the processes [7]. Since ICS environments are increasingly connected to corporate network and Internet, they are susceptible to cyber attacks [6, 5]. In particular, attackers target the control logic of a PLC over the network to manipulate the behavior of a physical process. Stuxnet, for instance, infects the control logic of a Siemens S7-300 PLC to modify the motor speed of centrifuges periodically from 1,410 Hz to 2 Hz

to 1,064 Hz and then over again. Since the discovery of Stuxnet in 2010, which sabotaged Iran's nuclear facilities, the number of ICS vulnerabilities reported each year has been dramatically increased [1].

To develop defensive solutions for control logic injection attacks, it is required to explore new attack vectors that target the control logic of a PLC to sabotage a physical process. In the past, different types of control logic injection attacks have been studied [9, 16, 15, 12]. For instance, the analysis of Stuxnet reveals that it compromises the STEP 7 engineering software in a control center to establish the communication with a target PLC in a field site and then transfers a prebuilt malicious control logic to the PLC. These attacks focus on the development and infection of malicious control logic. However, they do not consider the stealthy methods of transferring control logic to a PLC over the network.

This paper is the first effort in this direction and proposes two new control logic injection attacks: 1) Data Execution, and 2) Fragmentation and Noise Padding. These attacks manipulate control logic packets without disrupting the transfer of control logic to a PLC or modifying PLC firmware to support new (obfuscation) functionality.

Data Execution attack subverts the signatures that are based on packet-header fields by transferring the (compiled) code of control logic to the data blocks of a PLC. The data blocks are used to exchange data such as sensor measurement values and states of PLC variables (e.g, `timers` and `counters`). Thus, the signatures must not block their packets. After transferring the logic to a PLC, the attack further modifies the PLC's system control flow to execute the logic located in data blocks. Note that PLCs do not enforce data execution prevention (DEP), thereby allowing the logic to execute.

Fragmentation and Noise Padding attack subverts deep packet inspection (DPI) by generating write request packets that contain a small size of a code fragment with substantial padding of noise. The ICS protocols often have address/offset fields in their headers, which are utilized by the attack to make the PLC discard the noise padding.

We implement the attacks on two PLCs used in industrial setting (i.e., Schneider Electric's Modicon M221 and Allen-Bradley's MicroLogix 1400) and evaluate them against two well-known network intrusion detection methods, i.e., signature-based intrusion detection and DPI (or payload-based anomaly detection [17]). Our evaluation results show that both intrusion detection methods fail to detect the attacks and therefore, warrant more research efforts. We create and release the training and attack datasets to facilitate research in this direction.

Contributions. The contributions of the paper are summarized as follows:

- We propose two new stealthy methods to subvert the detection of control logic code in ICS network traffic.
- We implement and demonstrate the attacks successfully on two industry-scale PLCs of different vendors supporting two proprietary protocols, Allen-Bradley's PCCC and Schneider Electric's M221.
- We evaluate the attacks against both signature-based intrusion detection and DPI to provide evidence of stealth.

– To facilitate research on the defensive solutions for the control logic injection attacks, we create and release the normal and attack traffic datasets[1].

2 Background and Related Work

2.1 PLC & Control Logic

A PLC is an embedded device with multiple inputs/outputs connected to sensors and actuators, used for real-time control and automation of physical processes such as assembly lines and gas pipelines. The PLC primarily provides closed-loop control to maintain the desired state of a physical process. *Control logic* is a program that is executed repeatedly by the PLC in a *scan cycle* consisting of three steps: 1) the sensor inputs are copied to the I/O image table of the PLC, 2) the control logic is executed based on the input values and the state from the previous scan cycle, 3) the result of control logic execution is reflected in the I/O image table from where the output values are transmitted to the connected actuators. PLC vendors provide *engineering software* to program and compile source code of control logic[2]. The engineering software communicates with a PLC to transfer control logic through various interfaces such as RS-232.

Generally, we can divide control logic into four types of blocks that are transferred to/from a PLC: configuration blocks, code blocks, data blocks, and information blocks. The *configuration blocks* contain information about the other blocks such as the address and size of a block, PLC's IP address, etc. The *code blocks* contain the *compiled* control logic code that runs by a PLC. The *data blocks* maintain PLC variables (e.g, `input`, `output`, `timer`, `counter`, etc.) used in the code blocks. The *information blocks* are only used by the engineering software to recover the original project file from decompiled source code when the control logic is retrieved from a PLC.

2.2 Attacks on PLCs

We categorize the existing control logic attacks on PLCs into two groups: 1) traditional control logic attacks, and 2) through firmware modification.

Traditional control logic injection attacks. They involve modifying the original control logic running on a target PLC by engaging its engineering software, typically employing a man-in-the-middle attack [9, 16]. The primary vulnerability involved in this type of attack is the lack of authentication measures in the PLC protocols. In many cases, authentication is not supported for control logic download/upload operations or is supported in only one direction, either download or upload. When authentication measures are supported by PLCs, they are rarely used in practice. This section will discuss attacks from literature.

Stuxnet [9] is a representative example of this type of attack, which infects Siemens's STEP 7 (engineering software) and downloads malicious control logic to target PLCs (Siemens S7-300) by utilizing the infected engineering software.

Senthivel *et al.* [16] presents three control logic injection attack scenarios referred to as denial of engineering operations (DEO) attacks where an attacker

[1] https://gitlab.com/hyunguk/control-logic-attack-datasets

[2] IEC 61131-3 standard defines five PLC programming languages: ladder diagram, instruction list, functional block diagram, structured text, and sequential flow chart.

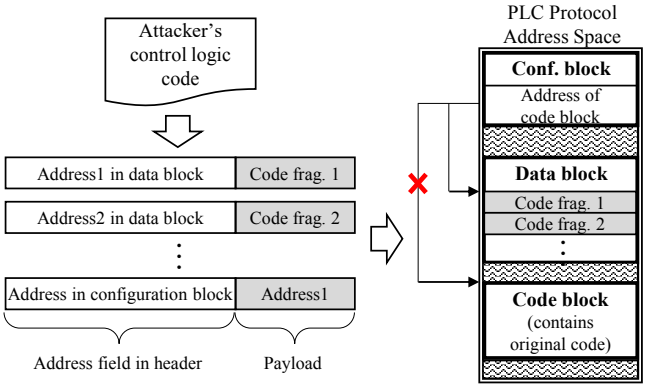

Fig. 1. Data Execution attack on protocol header inspection

can interfere with the normal engineering operation of downloading/uploading of PLC control logic. In DEO I, an attacker sitting in a man-in-the-middle position between a target PLC and its engineering software injects malicious control logic to the PLC and replaces it with normal (original) control logic to deceive the engineering software when uploading operation is requested.

DEO II is similar to DEO I except that it uploads malformed control logic instead of the original control logic to crash the engineering software. DEO III does not require a man-in-the-middle position, in which the attacker just injects specially crafted malformed control logic to the target PLC. The malicious control logic is specially crafted in a way that it can be run in the PLC successfully, but the engineering software can not decompile the control logic.

Firmware modification attacks. This type of attack [10] infect a PLC at firmware level. Since the firmware provides the control logic with an interface to the hardware such as inputs and outputs, malicious firmware can manipulate a connected physical process without tampering the control logic, which makes it difficult to detect the infection from engineering software or human-machine interfaces (HMIs) [10]. However, infecting PLC firmware would be a challenging task in a real ICS environment. Most PLC vendors protect their PLCs from unauthorized firmware update by cryptographic methods (e.g., digital signature) or allowing firmware update only by local access (e.g., SD cards and USB).

3 Stealthy Control Logic Injection Attacks

We propose two new attacks on PLC control logic in industrial control systems: 1) Data Execution, and 2) Fragmentation and Noise Padding, with the goal of subverting both signature-based intrusion detection and DPI.

3.1 Data Execution Attack

We present a typical signature-based approach to detect the packets of control logic attacks, identify the vulnerability that is exploitable and then, present the data execution attack based on the vulnerability.

Signatures for Detecting Control Logic Attacks. Stuxnet presents a typical control logic injection attack. It compromises the STEP 7 engineering software in a control center to transfer a prebuilt malicious control logic to a target

PLC. Generally, control logic injection attacks must transfer a code block of malicious control logic to a PLC. These attacks can be prevented by blocking the packets containing a code block over the network. The PLC protocol header has fields that indicate payload type and are utilized to detect the code-block packets via accurate signatures. For instance, Digital Bond's Quickdraw provides signatures for Snort IDS to monitor ICS traffic including the transfer of control logic [14]. An example signature monitors the network traffic of Schneider Electric's Modicon PLC and raises an alert on the Modbus function code `90` for the uploading/downloading of control logic [3]. Generally, the signatures can be based on any protocol header field that indicates code blocks such as the address field that indicates a restricted address range for code blocks, and the payload type field that identifies code blocks in packet payload.

Vulnerability. We make two observations about PLC communication that cause exploitable vulnerability. First, the data blocks cannot be blocked by the signatures because they are required to exchange the current state of a physical process (being controlled by a PLC) with the control center services such as HMI. Second, the PLCs do not enforce data execution prevention (DEP). Thus, PLCs may be manipulated to execute data blocks.

Subverting Signatures via Data Execution Attack. Figure 1 shows a high-level concept of the Data Execution attack that can subvert the signatures for preventing code blocks over the network. The attack consists of two steps: *First*, the attacker transfers (malicious) control logic code to an arbitrary address of a target PLC, which is not in the address range for code block, and thus, is not blocked by the signatures (e.g., transferring it to a data block).

Second, after sending the entire control logic, the attacker targets a special pointer in configuration block, that indicates the start address of code block and is used by the PLC to execute the control logic. The attacker modifies the pointer to the base address of the malicious control logic code, which in turn, redirects the PLC's system control flow to the malicious logic and forces the PLC to start executing it.

Note that while updating control logic do not occur frequently in real-world ICS environment, data values or configuration setting can be exchanged more frequently between PLCs and ICS services (such as HMI and engineering software) and thus, cannot be blocked by the signatures, making this attack stealthy for signature-based detection.

3.2 Fragmentation and Noise Padding Attack

We first discuss payload-based anomaly detection (DPI) that is suitable for ICS traffic monitoring for control logic attacks, identify an exploitable vulnerability and then, present the fragmentation/noise padding attack on the vulnerability.

Deep Packet Inspection for Control Logic Attacks. The semantics of ICS packet payload are often proprietary and unknown. A practical approach for the DPI of ICS traffic is to utilize byte level features without involving semantics. These approaches have been studied extensively in the literature such as PAYL [18] and Anagram [17]. Generally, they obtain n-gram (a sequence of

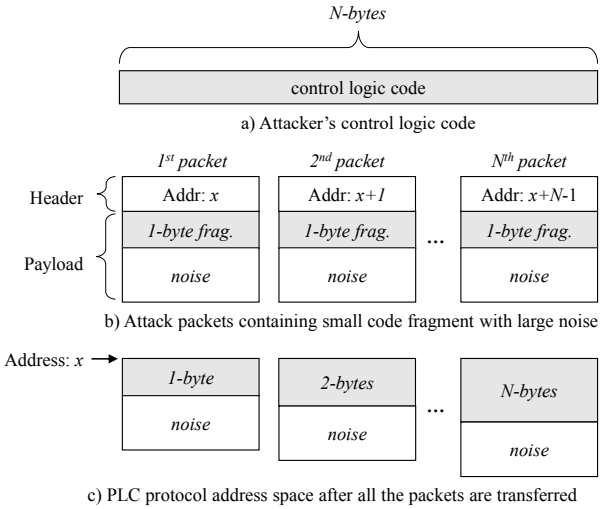

Fig. 2. Fragmentation and Noise Padding attack on deep packet inspection

consecutive `n` byte) frequency as features from packet payloads and then, apply statistics or machine learning algorithms for anomaly detection.

Vulnerability. Hadziosmanovic *et al.* [11] report that the aforementioned DPI techniques cannot detect attack packets that contain significantly small-size attack payload because these packets tend to blend with normal packets. In other words, the smaller the attacker's code in the payload is, the harder it is to detect.

DPI Subversion via Fragmentation and Noise Padding Attack. We utilize Hadziosmanovic *et al.* [11]'s findings in the control logic traffic by fragmenting the payload of the attacker's control logic and combining it with noise data to make it stealthy against byte-level features for DPI. We notice that some protocols of PLC such as Modicon M221 may allow an attacker to reduce the control logic fragment size to one or two bytes.

Figure 2 describes the *Fragmentation and Noise Padding* attack. Each *write request* packet contains only one-byte fragment of the attacker's code with a large noise of data (i.e., a sequence of padding bytes). To ensure that a target PLC does not use the noise and only execute actual control logic code in packet payloads, the attacker manipulates the address field stating the start location of the write operation in PLC memory for *write request* messages.

The core idea is that the next *write request* overwrites the noise data of the previous *write request* in PLC memory. In other words, the address of each write request packet increases by one (which is the size of an actual code fragment per packet). It keeps overwriting the noise data one byte per packet with the actual code bytes in contiguous memory locations. After all the packets are transferred to the target PLC, the whole code is placed from the address x to x+N-1 where x is the address of the first write request and N is the size of attacker's code.

Note that the attacker can evade both signatures and DPI by using both attacks together. The attacker transfers control logic code to a non-code block

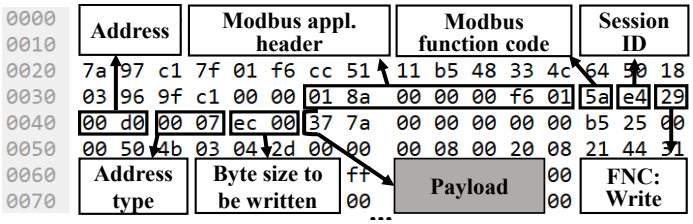

Fig. 3. The write request message format of the M221 protocol

in PLC (i.e., data execution), while keeping the size of code to one byte per packet blended with noise data (i.e., fragmentation and noise padding).

4 Implementation

We have implemented the attacks on two industry-scale PLCs: Schneider Electric's Modicon M221 and Allen-Bradley's MicroLogix 1400. To demonstrate the stealthiness of the attacks, we have also implemented two well-known network intrusion detection methods for proof-of-concept, 1) Scapy-based signatures, and 2) Anagram-based DPI [17]. This section further discusses the implementation details for each PLC.

4.1 Attacks on Modicon M221

We have implemented both attacks (i.e., data execution, fragmentation and noise padding) together for Modicon M221 as one stealthy attack against signatures and DPI. The attack consists of two phases: 1) preparation and 2) execution.

Preparation Phase. In this phase, an attacker prepares *compiled* control logic code, which will be transferred to a target PLC, in a lab environment.

Getting compiled code block. The attacker acquires compiled control logic code in the following way. She programs control logic using her engineering software and then, captures network packets when the control logic is being downloaded to a M221 PLC. From the captured packets, compiled logic code can be extracted by assembling the packet payloads containing logic code, indicated by the address fields of write request messages (refer to Figure 3)[3].

Execution Phase. The attack on Modicon M221 is conducted in five steps.

Step 1) Getting the address space layout of the target PLC. There are four types of block (configuration, code, data, and information) which are transferred to/from a PLC in the normal engineering operation of control logic. We refer to other areas in the address space of the PLC as *unknown areas* which are not occupied by the four types of blocks. Most of the unknown areas are filled with zero but they also contain the chunks of binary data which could be firmware.

To select an injection area, the attacker needs to know the address ranges for each control logic block. In the Modicon M221 PLC, there are two configuration

[3] By reverse engineering the M221 protocol (referred it to the PLC's model in this paper), we have figured out that the code block is always written between the address range `0xe000` $\sim$ `0xfed4`.

blocks which contain the start addresses and the sizes of other blocks. By reading the first configuration block[4], the attacker can locate the second configuration block and an information block. By further reading the second configuration block, the information of other blocks (i.e., two data blocks and one code block) can be acquired.

Step 2) Select an injection area. To evade signature-based header inspection, the attacker injects her code into an area that does not belong to code block. We have discovered that the address ranges for code block always fall between `0xe000` and `0xfed4`, which can be used as a signature to detect control logic over the network. Thus, the attacker can select an injection area except that address range. More specifically, the attacker selects an injection area among the information block, the data blocks, and the unknown areas, since there is no enough *available space* in the configuration blocks to fit logic code.

Step 3) Check the availability of the target injection area. Before injecting the logic code, the attacker needs to check if the target area is *available*, namely, overwriting the area does not affect the operations during the PLC's scan cycle. If the target area falls into the information block, it is not required to check the availability because the information block is only parsed in the engineering software, after being retrieved from a PLC. On the other hand, the attacker has to examine whether the target area is not currently used in the target PLC for the other locations (i.e., the data blocks and the unknown areas). Overwriting critical data/code of the unknown areas (e.g., firmware) will render the PLC inoperable. Similarly, if the attacker's code is written at the area for the I/O table which is maintained in the data blocks, the code will be overwritten with some arbitrary values at the next scan cycle of the PLC because the I/O table will be updated based on the inputs and outputs.

As a heuristic method, the availability of target area is tested by checking the entire area is filled with zero, based on our observation that the large amount of space in the data blocks and the unknown areas are just filled with zero. Our experimental results have shown that this method is actually effective since all the injected logic code is successfully run on a PLC.

Step 4) Transferring attacker's code using the fragmentation and noise padding. The attacker's logic code (i.e., the compiled code acquired in the preparation phase) is transferred to the target PLC using the fragmentation and noise padding attack. Each write request packet of the attack contains only one-byte of code fragment followed by 235 bytes of zero padding[5]. To perform the data execution attack simultaneously, the addresses of the write request messages start from the target address (i.e., the start address of the injection area) which was selected in the previous step.

Step 5) Change the pointer to code block. Lastly, the attacker modifies the pointer that points to the original logic code to point to the attacker's logic code. The pointer is two-byte size and located at the address `0xff90`, which is

[4] The first configuration block has a fixed start address (0xfed4) and size (300 bytes).
[5] The maximum payload size of the M221 protocol is 236 bytes

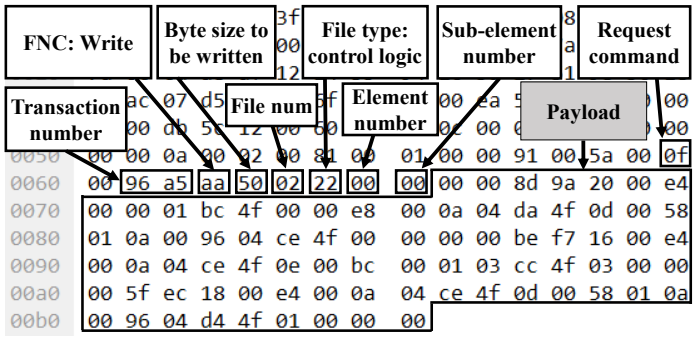

Fig. 4. The write request message format of the PCCC protocol

included in the first configuration block. Consequently, the PLC executes the attacker's code instead of the original code for each PLC scan cycle.

4.2 Attacks on MicroLogix 1400

Allen Bradley's MicroLogix PLC family uses the PCCC protocol to communicate with its engineering software [16]. Unlike the attack on Modicon M221, we could not find a way to make the MicroLogix 1400 PLC execute attacker's logic code which is stored in a non-code block and thus the data execution attack could not be implemented. We have implemented only the fragmentation and noise padding attack for the MicroLogix 1400 PLC.

Preparation Phase. In the preparation phase, an attacker prepares captured packets (i.e., a `pcap` file) which are generated in the downloading of attacker's control logic to a PLC from engineering software, in a lab environment.

Packets of attacker's control logic The MicroLogix 1400 PLC requires a legitimate sequence of communication including all the control logic blocks[6] for updating the logic of the PLC, whereas an attacker could transfer only a code block to the Modicon M221 PLC. To transfer an attacker's control logic to the MicroLogix 1400 PLC, the attacker uses the packets captured when the logic is being downloaded to the PLC from engineering software.

Execution Phase. We have developed an attack tool which takes as input a `pcap` file that is generated in the preparation phase, and replays all the PCCC request messages to the target PLC without modification except the write request messages containing logic code. For the messages indicated to contain logic code (i.e., write request messages with file number `0x02` and file type `0x22` [16]), their payloads (i.e., logic code) are fragmented into two-byte pieces[7] and a padding consisting of zero is appended to each code fragment. Then, new write request messages with code fragments and padding are sent to the target PLC, instead of the original messages.

When manipulating the messages of logic code, the payload size field next to the function code field (refer to Figure 4) should be fixed according to the

[6] Control logic blocks are referred to as files in MicroLogix PLC family.

[7] Two-byte is the smallest size of payload in the PCCC protocol since the basic data unit of the PLC is 16-bit word.

modified payload size. Similarly, the sub-element number field which indicates an offset to write within a file (or a block) should be adjusted to the total size of attacker's code fragments transferred previously, to overwrite the padding.

4.3 Network Intrusion Detection Systems (NIDS)

We have implemented two proof-of-concept tools for network intrusion detection, one represents signature-based detection for ICS protocols and other is DPI based on Anagram [17], which is a popular payload-based anomaly detection approach that uses byte-level (n-gram) features. We will show how they effectively detect a traditional control logic injection attack which does not involve any evasion techniques, and evaluate the proposed attacks against them, in Section 5.

Scapy-based Signatures using Packet Header Fields. We have developed a signature-based control logic detection tool in Python using the Scapy library. In the Modicon M221 PLC, control logic code always exists in the address range between `0xe000` and `0xfed4`. Based on this feature, we configure a signature that checks if the address field of write request message is greater than `0xe000` and less than `0xfed4`, which means the write request message contains logic code.

Anagram-based Deep Packet Inspection Method. Anagram [17] is a payload-based anomaly detector that models a higher-order n-grams ($1 < n$) analysis, which is one of the most effective payload-based anomaly detector for ICS network traffic [11], not requiring semantic knowledge about packet payloads. In a nutshell, it stores n-grams ($1 < n$) observed in the payloads of normal datasets (i.e., training datasets) in a bloom filter and then, scores each packet by measuring the number of n-grams that were not observed in the normal datasets (i.e., the n-grams which are not stored in the bloom filter), to classify abnormal packets in the detection phase.

In our implementation of Anagram, we models the payload of logic code instead of the payload of normal packets. Remember that our goal is not to detect abnormal packets but to detect packets containing control logic code. Verifying control logic is out of the scope of this work, which has been studied in the literature such as TSV [13]. Note that the detection of control logic must be done to verify it.

In the training phase, we store all the n-grams seen in the payloads of logic code in a bloom filter, based on a training dataset. We examine different n-gram sizes from 2-gram to 20-gram, to find the optimal size of n-gram for each PLC. From our experiment, we have found that 4-gram is the optimal size for Modicon M221 whereas 8-gram for MicroLogix 1400.

In the detection phase, each packet is scored by counting the n-grams in its payload that are present in the bloom filter. If the score is equal or greater than a threshold, it is classified as containing logic code. We use 4 as thresholds for both PLCs with which false positive rates are 0% as discussed in Section 5.

5 Evaluation

5.1 Experimental Settings

We evaluate the proposed attacks on two different PLCs used in industrial setting, Schnider Electric's Modicon M221 and Allen-Bradley's MicroLogix 1400,

Table 1. Description of the datasets for Modicon M221

Datasets	Size (MB)	# of control logic	# of M221 packets	# of write request packets	# of packets w/ code
Training	2.1	22	10,148	1,101	27
Code injection w/o evasion	3.7	29	11,092	1,535	38
Data execution & Noise padding	2.2	29	8,168	5,362	3,865

Table 2. Description of the datasets for MicroLogix 1400

Datasets	Size (MB)	# of control logic	# of PCCC packets	# of write request packets	# of packets w/ code
Training	19.9	52	71,824	4,084	646
Code injection w/o evasion	39.7	75	168,736	5,465	684
Noise padding	38.6	75	238,657	29,647	24,866

against two different intrusion detection methods, signature-based intrusion detection and DPI. For each PLC, we use its engineering software, SoMachine-Basic v1.6 (Modicon M221) and RSLogix 500 v9.05.01 (MicroLogix 1400), to generate training datasets and in the preparation phase of attack.

The system implementing the proposed attacks runs on Ubuntu 16.04 virtual machine, which communicate with the PLCs over Ethernet. We also place a traditional attacker in the network who utilizes the engineering software (running on Windows 7 virtual machine) to inject control logic to the PLCs. Then, we capture the network traffic from each attack to generate attack datasets, and examine if our intrusion detection systems can detect the packets of attacker's logic code.

5.2 Datasets

For the evaluation, we developed 51 different programs of control logic for the Modicon M221 PLC (22 programs for training and 29 programs for attack), and 127 programs for the MicroLogix 1400 PLC (52 programs for training and 75 programs for attack), referred to Tables 1 and 2. The programs are written for different physical processes such as water tanks, gas pipelines, and traffic light involving various complexity.

Training Datasets. The training datasets are generated by capturing network traffic when control logic is being downloaded to a PLC. Based on the training datasets, we have generated the bloom filters of n-grams for each PLC (4-gram for Modicon M221 and 8-gram for MicroLogix 1400).

Traditional Attack Datasets. Generally, traditional attackers remotely targeting the control logic of PLCs engage engineering software typically in a way of man-in-the-middle attack as they did in Stuxent [9] and the DEO attacks [16]. To generate the traditional attack datasets, therefore, we bring an attacker who utilize engineering software to inject control logic to a PLC. Note that the traditional attack does not include any evasion techniques.

Table 3. Detection rates of signautres and DPI against the attacks on Modicon M221

Attacks	# of write request packets	# of packets w/ code	TPR	FPR
Code injection w/o evasion	1,535	38	100% (38/38)	0% (0/1497)
Data execution & Noise padding	5,362	3,865	0% (0/3865)	0% (0/1497)

Table 4. Detection rates of DPI against the attacks on MicroLogix 1400

Attacks	# of write request packets	# of packets w/ code	TPR	FPR
Code injection w/o evasion	5,465	684	96.78% (662/684)	0% (0/4781)
Noise Padding	29,647	24,866	0% (0/24866)	0% (0/4781)

Attack Datasets for Data Execution/Noise Padding. For the datasets of the proposed attacks, we transfer the same set of control logic used in the traditional attack to a PLC but in different ways, as described in Section 4. The attack dataset of Modicon M221 includes both the data execution and noise padding attacks[8] whereas the attack dataset of MicroLogix 1400 includes only the noise padding attack.

5.3 Evaluation Results

We present the evaluation results on the traditional attack and the proposed attacks against our implementations of signatures and DPI, described in Tables 3 and 4. The packets of the traditional attack targeting the Modicon M221 PLC is detected on both signatures and DPI with 100% true positive rate (TPR) and 0% false positive rate (FPR). For the MicroLogix 1400 PLC, the traditional attack is detected on DPI with 96.78% of detection rate and 0% of false alarm[9] This result shows that our implementations of signatures and DPI are very effective to detect traditional control logic injection attacks which do not involve any evasion techniques.

Data Execution Attack. We evaluate the effectiveness of the data execution attack on the Modicon M221 PLC against the signature-based IDS. Table 3 shows that the IDS was fully-functional to detect the control logic packets that do not involve the data execution attack. However, the IDS was subverted completely and could not detect a single attack packet when data execution attack was employed.

Fragmentation and Noise Padding Attack. We evaluate the effectiveness of the fragmentation and noise padding attack on both Modicon M221 and MicroLogix 1400 against the Anagram-based DPI. Tables 3 and 4 show the evaluation results. We noticed a significant increase on the number of packets due to the high fragmentation of control logic i.e., one or two bytes of control logic

[8] Short for the fragmentation and noise padding attack

[9] With the classification threshold of 3 for MicroLogix 1400 (instead of 4), the detection rate can be slightly increased (97.95%) in return for increased FPR (0.08%).

code per packet. We also found that the noise padding attack subverted the DPI completely. Recall that the DPI was fully functional and detected the control logic packets successfully when the noise padding attack was not used.

6 Countermeasures

This section discusses the countermeasures against the proposed attacks in the following four categories.

User Authentication. Authentication for control logic download operation (or write requests) can defeat the proposed attacks, although it is not supported in many PLCs or just rarely used in practice. When authentication is supported in a PLC, an attacker may find a vulnerability to bypass it [2].

Data Execution Prevention. Data execution prevention (DEP) technique is a well-known protection mechanism used in modern desktop/server environments to prevent code injection attacks to stack or heap. It enforces $W \oplus X$ on all memory regions using a memory management unit (MMU), which is usually not present on microcontrollers. To enforce no-execution of code on non-code regions of a PLC, a memory protection unit (which is present in many microcontrollers) can be utilized as studied in [8].

Control Flow Integrity. Control-flow integrity (CFI) mechanism is a protection technique to detect control-flow hijacking attacks by restricting control-flow transfers based on a pre-determined control-flow graph. Most of CFI solutions designed for general-purpose computers may occur significant performance overhead which can be a critical problem for hard real-time PLCs. Nonetheless, a specially designed CFI solution such as [4], which consider availability and real-time requirements of the ICS systems, could be used.

Robust Network Intrusion Detection Systems. The previously discussed countermeasures require significant change/update on PLC design. It is not expected that those solutions could be widely deployed in the near future, considering the long life-cycle of ICS systems. On the other hand, network-based IDS solutions have the advantages of being deployed without significantly affecting existing systems. Although we have demonstrated that the existing approaches can be deceived by the proposed attacks, there is a room for research to subvert the attacks. For instance, the fragmentation and noise padding attack could be subverted if a NIDS keeps the state of PLC's address space using a mirrored space. Even though there is only small amount of code fragment in an individual attack packet (which is undetectable), it could detect attacker's code by scanning the mirrored space instead of individual packet.

7 Conclusion

We presented two new stealthy control logic injection attacks to demonstrate that an attacker can subvert both signature-based header inspection and DPI to transfer control logic to a PLC successfully. The attacks were implemented on two industry-scale PLCs and demonstrated their effectiveness against two well-known intrusion detection approaches, Scapy-based Signature and Anagram-based DPI. Our evaluation results showed that the attacks were stealthy and warrants for more research on IDS for ICS network traffic monitoring.

References

1. ICS-CERT Annual Vulnerability Coordination Report. Report, National Cybersecurity and Communications Integration Center (2016)
2. ICS-CERT Advisory on Modicon M221 (ICSA-18-240-01). https://ics-cert.us-cert.gov/advisories/ICSA-18-240-01 (2018), [Online; accessed 15-Dec-2018]
3. Quickdraw Snort – modicon.rules. https://github.com/digitalbond/Quickdraw-Snort/blob/master/modicon.rules (2018), [Online; accessed 15-Dec-2018]
4. Abbasi, A., Holz, T., Zambon, E., Etalle, S.: Ecfi: Asynchronous control flow integrity for programmable logic controllers. In: Annual Computer Security Applications Conference (ACSAC) (2017)
5. Ahmed, I., Obermeier, S., Naedele, M., III, G.G.R.: SCADA systems: Challenges for forensic investigators. Computer **45**(12), 44–51 (Dec 2012). https://doi.org/10.1109/MC.2012.325
6. Ahmed, I., Obermeier, S., Sudhakaran, S., Roussev, V.: Programmable logic controller forensics. IEEE Security Privacy **15**(6), 18–24 (November 2017). https://doi.org/10.1109/MSP.2017.4251102
7. Ahmed, I., Roussev, V., Johnson, W., Senthivel, S., Sudhakaran, S.: A scada system testbed for cybersecurity and forensic research and pedagogy. In: Proceedings of the 2Nd Annual Industrial Control System Security Workshop. pp. 1–9. ICSS, ACM, New York, NY, USA (2016). https://doi.org/10.1145/3018981.3018984
8. Clements, A.A., Almakhdhub, N.S., Saab, K.S., Srivastava, P., Koo, J., Bagchi, S., Payer, M.: Protecting bare-metal embedded systems with privilege overlays. In: 2017 IEEE Symposium on Security and Privacy (SP) (2017)
9. Falliere, N., Murchu, L.O., Chien, E.: W32. stuxnet dossier. White paper, Symantec Corp., Security Response **5**(6), 29 (2011)
10. Garcia, L., Brasser, F., Cintuglu, M.H., Sadeghi, A.R., Mohammed, O., Zonouz, S.A.: Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In: Network and Distributed System Security Symposium (NDSS) (2017)
11. Hadžiosmanović, D., Simionato, L., Bolzoni, D., Zambon, E., Etalle, S.: N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. In: International Conference on Research in Attacks, Intrusions, and Defenses (RAID) (2012)
12. Kalle, S., Ameen, N., Yoo, H., Ahmed, I.: CLIK on PLCs! Attacking Control Logic with Decompilation and Virtual PLC. In: Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS) (2019)
13. McLaughlin, S.E., Zonouz, S.A., Pohly, D.J., McDaniel, P.D.: A trusted safety verifier for process controller code. In: Network and Distributed System Security Symposium (NDSS) (2014)
14. Peterson, D.: Quickdraw: Generating security log events for legacy scada and control system devices. In: IEEE Conference For Homeland Security (2009)
15. Senthivel, S., Ahmed, I., Roussev, V.: Scada network forensics of the pccc protocol. Digital Investigation **22**, S57–S65 (2017)
16. Senthivel, S., Dhungana, S., Yoo, H., Ahmed, I., Roussev, V.: Denial of engineering operations attacks in industrial control systems. In: ACM Conference on Data and Application Security and Privacy (CODASPY) (2018)
17. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A content anomaly detector resistant to mimicry attack. In: International Conference on Recent Advances in Intrusion Detection (RAID) (2006)
18. Wang, K., Stolfo, S.J.: Anomalous payload-based network intrusion detection. In: RAID 2014. Lecture Notes in Computer Science. vol. 3224. Springer (2004)