Chapter 1

ATTACKING THE IEC-61131 LOGIC ENGINE IN PROGRAMMABLE LOGIC CONTROLLERS IN INDUSTRIAL CONTROL SYSTEMS

Syed Ali Qasim, Adeen Ayub, Jordan Johnson and Irfan Ahmed

Abstract In industrial control systems (ICS), programmable logic controllers (PLCs) directly monitor and control a physical process such as nuclear power plants, gas pipelines, and water treatment. They are equipped with a control logic written in IEC-61131 languages (e.g., ladder diagram and structured text) that defines how a PLC should control a physical process. A PLC's control logic is a usual target of a cyberattack to sabotage a physical process. The existing attacks in the literature generally focus only on injecting malicious control logic into a PLC. This paper presents a new dimension of control logic attacks that target the control logic engine (responsible for running a control logic) of a PLC. It demonstrates that a cyberattack can disable the control logic engine successfully by exploiting inherent PLC features such as program mode and starting/stopping engine. We develop two novel case studies on control logic engine attacks by employing the MITRE ATT&CK knowledge base on the real-world PLCs used in industry settings, i.e., 1) Schweitzer Engineering Laboratory (SEL)'s Real-Time Automation Controller (SEL-3505 RTAC) equipped with security features such as encrypted traffic and device-level access control, and 2) traditional PLCs, i.e., Schneider Electric's Modicon M221, Allen-Bradley's MicroLogix 1400 and 1100 that do not have security features. The case studies present the internals of the logic engine attacks and facilitate the ICS research community and industry to understand the attack vectors on the control logic engine. We evaluate the effectiveness of the control engine attacks on a power substation, a 4-floor elevator, and a conveyor belt to demonstrate their real-world impact of halting a physical process

Keywords: Critical Infrastructure Industrial Control Systems SCADA Programmable Logic Controller IEC-61131 MITRE ATT&CK Stuxnet PLC TRISIS CPS Security Cyber-physical Systems

1. Introduction

Industrial control systems (ICS) monitor and control industrial physical and infrastructure processes such as power grids, nuclear plants, water treatment facilities, and gas pipelines [16, 4, 5]. An ICS environment consists of a control center and a field site. The control center consists of ICS services such as human-machine interface (HMI) and engineering workstation, while the field sites consist of the actual physical processes monitored and controlled via sensors, actuators, and programmable logic controllers (PLCs). PLCs are embedded devices that directly automate industrial processes [7]. They have a control logic program written in IEC 61131 languages such as instruction list, ladder diagram, and structured text that defines how a physical process is controlled.

A PLC's control logic is a usual target of a cyberattack to sabotage a physical process [8]. However, in the literature, the existing control logic attacks only focus on injecting malicious control logic in a target PLC over the network to disrupt or cause damage to the underlying physical process [28, 19, 20, 27, 15, 25, 9, 21, 6]. For instance, Stuxnet, a piece of ICS malware, targets Siemens Step 7 engineering software and S7-300 PLCs in a nuclear plant facility to inject malicious control logic [11].

This paper introduces a new dimension of control logic attacks by targeting the control engine (responsible for running a control logic) in a PLC. It shows that a cyberattack can successfully disable an IEC-61131 control-logic engine to halt a physical process controlled by a target PLC by exploiting the PLC design features such as program mode and starting/stopping engine. We develop two novel case studies on control logic engine attacks by utilizing the attacks in MITRE ATT&CK knowledge base such as denial of control (T0813), loss of availability (T0826), manipulation of control (T0831), unauthorised command message (T0855), and man in the middle (T0830) [2]. We employ the MITRE ATT&CK on the logic engine in four real-world PLCs used in industrial settings. The first case-study covers Schweitzer Engineering Laboratory (SEL) Real-Time Automation Controller (SEL RTAC 3505), which is well-equipped with security features such as device access control encrypted traffic. The second case-study consists of three traditional PLCs with no security features, i.e., Schneider Electric's Modicon M221 and Allen-Bradley's Micrologix 1400 and 1100 PLCs. The case studies discuss the internals of the control logic engine attacks, including proprietary PLC communication protocols, and facilitate the ICS research community and industry to understand the attack vectors on the control logic engine.

We evaluate the effectiveness of the control engine attacks on a power substation, a conveyor belt, and a 4-floor elevator to demonstrate their

real-world impact. In the conveyor-belt, a PLC controls the sorting of different types of objects using sensors and air solenoid. In the substation, a PLC opens a circuit breaker when the voltmeter reports a voltage level higher than a given threshold. The control engine attacks halt the conveyor belt and prevent substation to control high voltage. In the elevator, a user can select a floor both from inside and outside the elevator as an input to a PLC. In response, the PLC moves the elevator to the desired floor.

Contributions. Our contributions are threefold:

- We have introduced a new attack vector that targets an IEC-61131 control-logic engine in a PLC to halt a physical process.
- We successfully utilize the MITRE ATT&CK knowledge base to develop and demonstrate control engine attacks on four real-world PLCs.
- We evaluate the effectiveness of the control engine attacks on connected physical processes i.e., a power substation, a 4-floor elevator, and a conveyor belt to demonstrate their real-world impact of halting a physical process.

2. Background and Related Work

2.1 Primer of Industrial Control Systems

Figure 1 shows a typical example of an industrial control system environment. An industrial control system can be divided into two parts: 1) field site, and 2) control center.

Field Site. The physical process is controlled and monitored with sensors, actuators, and programmable logic controllers (PLC) on the field sites. Figure 1 shows a typical industrial process of generating steam. The water is added into the boiler and heated to convert it into vapor, then transferred via a pipeline. The PLC receives different sensor data on the water-level, pressure, and temperature in the tank and then processes the data using a control logic to control these parameters through operating valves. The PLC also sends the process state to the control center over the network.

Control Center. The control center consists of a human-machine interface (HMI), historian, control server, and engineering workstation. The human-machine interface (HMI) shows the current state of the physical process. At the same time, the historian keeps logs of programmable

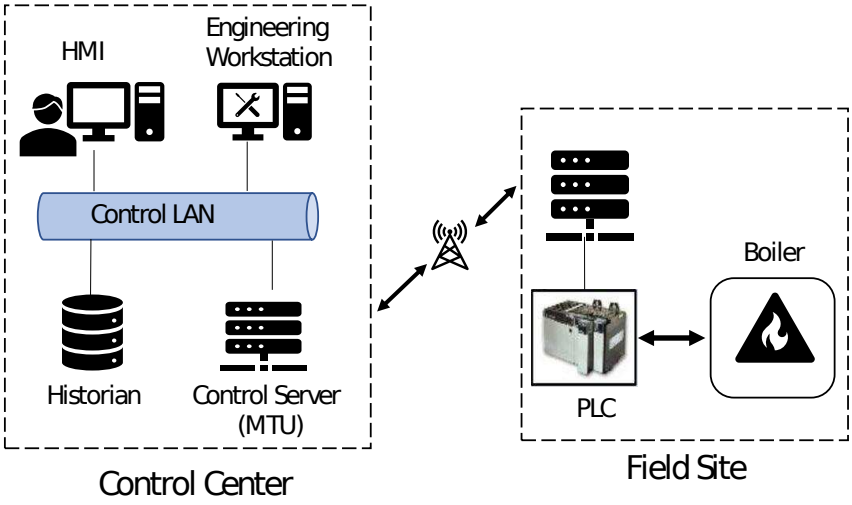


Figure 1: Industrial control system scenario for industrial boiler

controllers' input and output data for forensics and analytic purposes. The control server communicates with the field site over the network. The engineering workstation runs engineering software provided by the programmable controller's vendor to maintain and program the controllers remotely. The control engineer can write a control logic program in the engineering software and then download (write) it to a PLC or upload (read) the existing code running on a PLC. IEC 61131-3 Standard allows five languages, i.e., ladder diagram (LD), sequential function charts (SFC), function block diagram (FBD), structured text (ST), and instruction list (IL), to write a control logic program.

2.2 Related Work

Existing control logic attacks in the literature either target a control logic code running on a PLC (also referred to as control logic injection attacks [26, 14, 23]) or compromise PLC firmware to manipulate control logic execution [12]. Our attacks are new in that they target control logic engine instead of control logic code in a target PLC without modifying the PLC firmware. Table 1 shows the comparison of different attacks on PLCs.

Senthival *et al.* [25] present three types of denial of engineering operations (DEO) attacks. In the first DEO attack, the attacker intercepts the network traffic between the engineering workstation and a target PLC and replaces the original ladder diagram program with an infected

	Senthival et al.	Yoo et al.	Govil et al.	Stuxnet	Kalle et al.	McLaughlin et al.	Garcia et al.	Schuett et al.	This paper
Firmware Modification							x	x	
Malicious Control Logic	x	x	x	x	x	x			
Control Engine State									x

Table 1: The comparison of different attacks on PLCs.

one and vice versa when a control-logic program is downloaded and uploaded, respectively. Similarly, for the second DEO attack, the man-in-the-middle attacker replaces part of the original ladder diagram program with noise when a control logic program is uploaded, thereby crashing the engineering software. The third DEO attack also crashes the implementing software, but this time the attacker remotely downloads an infected control logic to a PLC instead of performing man-in-the-middle.

Kalle *et al.* [15] present CLIK, a control logic infection attack, comprising of four phases. First, it compromises PLC security measures and steals the control logic from it. Then, it decompiles the stolen binary of the control logic to inject the malicious logic, followed by transferring the infected binary back to the PLC. Finally, it hides the malicious logic written into the PLC from the engineering software by employing a virtual PLC that first captures the original logic's network traffic and then sends this network traffic to the engineering software when it tries to read the control logic written inside the PLC.

Similar to CLIK, McLaughlin *et al.* presented SABOT [18] a tool that first uploads the targeted PLCs control logic byte code and decompiles it into logical model and find a mapping between the devices connected to the PLC and variables within the control logic. The attacker can then change this mapping arbitrarily and download the control logic back to

the PLC to cause damage to the plant. SABOT assumes the attacker has the knowledge of ICS operations.

Yoo *et al.* [27] present two control logic injection attacks, namely 1) data execution and 2) fragmentation and noise padding. In the data execution attack, the attacker exploits the fact that the PLCs do not enforce data execution prevention (DEP) and transfers the attacker's control logic to the data blocks of the PLC. The attacker then changes the PLC's system control flow to execute the logic located in data blocks. Fragmentation and noise padding attack subverts deep packet inspection by sending write requests with the attacker's control logic. Each write-request contains one byte of the control logic while the rest of the packet includes noise. For every next write-request, the attacker attempts to overwrite the PLC memory region previously written with noise due to the previous request.

Govil *et al.* [13] presented malware written in ladder logic called ladder logic bomb that an attacker can insert these malwares into the existing control logic of a PLC. These logic bombs are hard to detect by a control engineer manually validating the control logic running on the PLC. These bombs can either be activated via trigger signals to cause the disruption or can persistently damage the physical operations over time.

Garcia *et al.* [12] presented Harvey, a model aware rootkit that sits in a PLC firmware using JTAG (Joint Test Action Group) [22]. From the legitimate input data Harvey generates fake, real-looking input. The PLC processes this input according to the control logic and generates output commands to actuators. Harvey blocks this output at firmware level and sends the malicious output generated by attackers code to the sensors. This abstraction helps Harvey in deceiving the control engineer monitoring the HMI.

Schuett *et al.* [**?**] evaluated the possibility of modifying the PLC firmware to execute remotely-triggered attacks. They first reverse engineered the PLC framework and added modification. This modified firmware is repackaged and installed on the PLC. Using this compromised firmware, the attacker is able to perform time-based or remote triggered denial of service attacks on the PLC.

3. Attacking Control Logic Engine

3.1 Adversary Model

We assume the adversary is in the ICS network and can communicate with the target controller to launch a control logic engine attack. The attacker can use a real-world IT attack (such as an infected USB stick or a vulnerable webserver) to infiltrate the ICS network and disable

Table 2: Subsets of MITRE ATT&CK utilized on four PLCs in the case studies

PLC	Manipulation of Control	Loss of Availability	Denial of View	Denial of Control	Man in the Middle	Network Sniffing	Unauthorized Command Message
SEL RTAC 3505	✓				✓	✓	
Modicon M221	✓	✓		✓		✓	✓
MicroLogix 1100	✓		✓	✓	✓	✓	✓
MicroLogix 1400	✓		✓	✓	✓	✓	✓

the controller from running the control logic. IT attacks are out of the scope of this paper. While in the ICS network, We also assume that the attacker has the following capabilities:

- Reading the communication between a PLC and an engineering workstation.
- Dropping or modifying any message in the communication by positioning herself as man-in-the-middle.
- Initiating a connection with a PLC to send malicious messages remotely.

3.2 Overview of the Case Studies

We propose two novel case studies to explore IEC-61131 control logic engine attacks on real-world PLCs used in industry settings. A control logic engine attack is defined as "an attack that disrupts or impairs a normal functioning of a control logic engine." The case studies explore cyberattacks that can stop a control-logic engine from executing a control logic. Our methodology to perform the case studies is to employ MITRE ATT&CK [2] knowledge base on real-world PLCs to target the control-logic engine. Specifically, the studies utilize a subset of the following attacks (along with their IDs) from the knowledge base to demonstrate the control-logic engine attacks. Table 2 summarizes the attack subsets used from MITRE ATT&CK knowledge to demonstrate control logic engine attacks on the PLCs.

- *Manipulation of Control (T0831).* The attacker can manipulate physical process control within the industrial environment

- *Loss of Availability (T0826).* The attacker can disrupt some component in order to prevent the operator from delivering the products or services
- *Denial of View (T0815).* The attacker can disrupt or prevent the operator from viewing the status of an ICS environment
- *Denial of Control (T0813).* The attacker can temporarily prevent the operator from interacting with process controls
- *Man in the Middle (T0830).* An attacker in the ICS network can intercept, modify or drop the packets getting exchanged between the engineering workstation and the PLC
- *Network Sniffing (T0842).* An attacker in the ICS network can attempt to sniff the network traffic in order to gain information on its target
- *Unauthorised Command Message (T0855).* Attackers may send unauthorised command messages to industrial control systems devices in order to make them function improperly

The first case study focuses on an SEL RTAC device equipped with security features such as encrypted traffic and device-level access control. The RTAC has a component termed as 'Logic engine', responsible for running the controller's control logic. With this component being the attacker's primary concern, she positions herself as a man-in-the-middle between the engineering software and the PLC to prevent the controller from executing the control logic in two ways. 1) By modifying the packet responsible for starting the logic engine 2) By dropping this packet entirely. Note that the initial communication with the controller is encrypted using transport layer security (TLS). Unencrypted communication begins once a legitimate user has logged in, thereby making the man-in-the-middle attack possible. For details, see Section 1.4.

The second case study focuses on traditional PLCs that have no built-in security features. It involves three PLCs, Modicon M221, MicroLogix 1100, and MicroLogix 1400. These PLCs differ from the RTAC in two ways: Firstly, these PLCs do not have a separate logic engine, and the processor takes up this role. Depending on the controller type, the PLC either needs to be in 'run' mode or given a 'start controller' command to run the control logic. Secondly, unlike the case for SEL-RTAC, all communication done with most of the PLCs is unencrypted. To attack the control logic engine, the attacker can create a well-crafted message and then send it to the PLC to remotely change its state. For details, see Section 1.5.

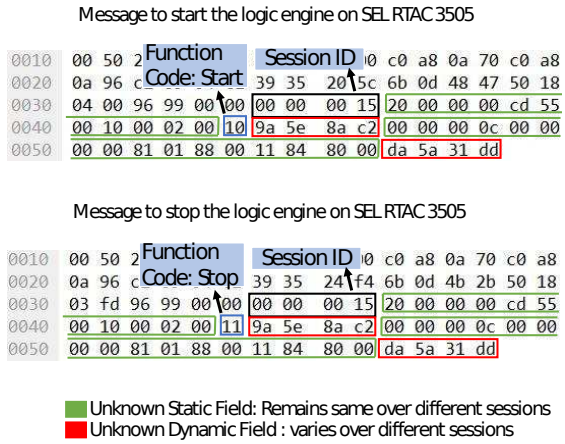


Figure 2: Messages sent by AcSELerator software to SEL RTAC 3505 to start and stop logic engine

4. Case Study I: SEL-3505 RTAC

4.1 Controller Details

SEL-3505 Real-Time Automation Controller is developed by Schweitzer Engineering Laboratories equipped with an IEC 61131 control logic engine. It has a web interface to monitor and configure the network interface, system logs, user accounts, and security settings. The control engineer can write the control logic, protocol communication configuration, read/write projects, and start or stop the logic engine using the AcSELerator RTAC SEL-5033 software [17].

In terms of device-level security, RTAC uses ex-GUARD [10], a whitelist based system to control the execution of different tasks. It blocks any tasks from operation [17], not approved by the whitelist. The RTAC 3505 communicates with the AcSELerator software on port 5432. Most communication, including session establishment, user authentication, and reading and writing project on RTAC, is encrypted using TLS encryption. However, after a user logs in, the controller opens another port, 1217, and starts a second communication channel for sharing the state of RTAC in real-time. Surprisingly, the communication on port 1217 is unencrypted.

4.2 Vulnerability

We explore the RTAC communication internals and find that the RTAC sends the unencrypted commands on port 1217 for starting or stopping the logic engine. We further identify the packets carrying the commands. We also reverse engineer the commands to understand function codes and other fields such as session ID. Figure 2 shows the two request packets sent by AcSELerator to the RTAC to start and stop the logic engine. We find that the session ID is incremented by three in every new session, and the function code for starting and stopping the logic engine is `0x10` and `0x11` respectively. We also find that the rest of the messages remain the same in different sessions (identified as unknown static fields) and do not require semantic knowledge for the attacks.

4.3 MITRE ATT&CK

We utilize the following attacks from the MITRE ATT&CK knowledge base for the case study.

Network Sniffing (T0842). Network Sniffing is the first step in finding the vulnerabilities and launching our final attack. Since the attacker's machine is on the same network as the legitimate engineering workstation and the controller, she can easily sniff the network traffic between the legitimate parties to find ways of attacking her target. Through sniffing, we can find the port on which the communication is unencrypted and the necessary fields for designing our Ettercap filters [1].

Man in the Middle (T0830). After sniffing the network traffic and identifying the packets responsible for starting/stopping the logic engine, the attacker develops Ettercap filters, poisons the ARP cache of the target machines, and positions herself as man-in-the-middle between the AcSELerator software and the SEL RTAC device in order to either modify the content of the packet going from the engineering software to the device or to drop this packet entirely.

Manipulation of Control (T0831). This attack is a continuation of the above-mentioned man-in-the-middle attack. As a result of the steps she takes to launch a man-in-the-middle attack, she can stop the control logic engine from running the control logic on the PLC, which ultimately stops executing the underlying physical process.

Pseudocode for RTAC filters

```
Input: TCP packet
1: if (packet_src ==AcSELerator & packet_dst ==RTAC & packet_port ==1217)
  2: if (packet_payload_contains (static_fields) )
    3: Modify/Drop
```

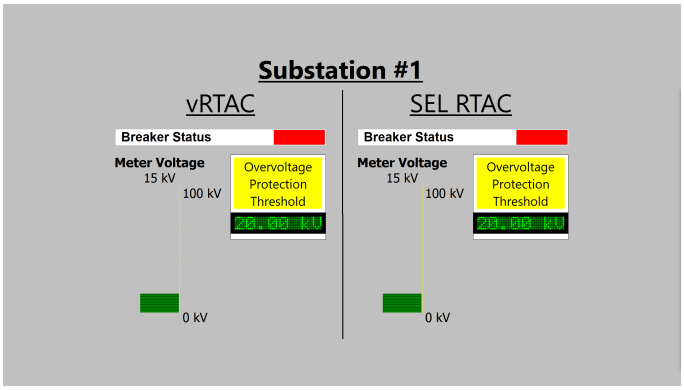
4.4 Attack Implementation

Using the derived information, we developed two Ettercap filters (DropFilter & Start-StopFilter) that can successfully identify the messages containing the logic engine 'start' and 'stop' command, modify them (convert 'start' to 'stop' and vice versa), or drop them all together to stop the logic engine or block the control engineer from accessing the logic engine.

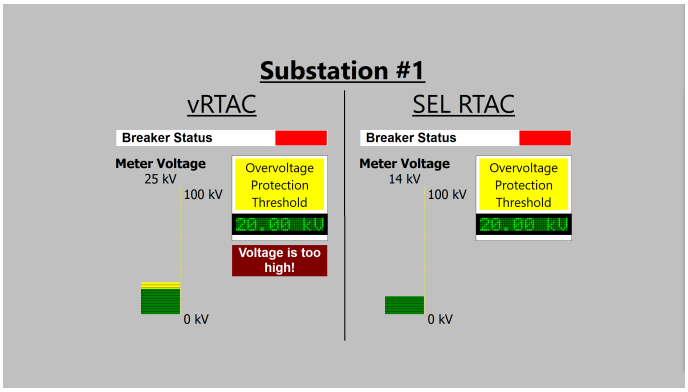
The filters first identify the messages from the AcSELerator to RTAC using the respective IP address and port 1217. As shown in figure 2, a big chunk of the messages containing 'start' and 'stop' commands remains same in different sessions, termed as unknown static field. The filters search these static bytes in the TCP payload of the message to identify the right message. After correctly identifying the message, the DropFilter can use the "drop()" command to drop this message. Similarly, the Start-StopFilter can change the function code located at 15th index in the TCP payload from start (`0x10`) to stop (`0x11`) and vice versa.

4.5 Evaluation

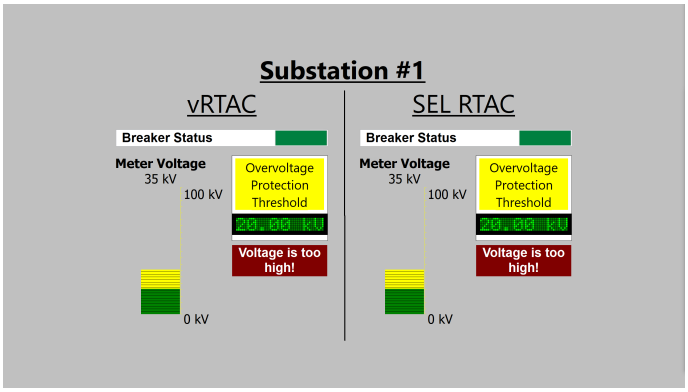
4.5.1 Experimental Settings.. We evaluate the control logic engine attack on a power substation consisting of an SEL-3505 RTAC connected to an engineering workstation, a circuit breaker, and an emulated voltage measurement device designed to behave as a voltmeter. The RTAC is configured to open the circuit breaker when the voltmeter reports a voltage level higher than a given threshold. The operators in the control center configure the threshold. If the circuit breaker does not open quickly after the voltage rises too high, expensive power equipment will be damaged or destroyed. The system is monitored from an HMI shown in Figure 3a. Note that in our evaluation system, the "vRTAC" shows the system's ground truth state even if the SEL RTAC has failed. It shows the real-time dispute between the true state and what the RTAC is reporting during the attack.



(a) HMI showing ground truth (left) and SEL RTAC state (right) for a circuit breaker (red means closed) and voltmeter with a given over-voltage protection threshold



(b) HMI showing SEL RTAC is no longer reporting new values while voltage has surpassed the threshold



(c) HMI showing updated SEL RTAC after the logic engine is enabled and the circuit breaker is now open (green means open)

Figure 3: The figures show the effect of control engine attack on SEL RTAC

4.5.2 Attack Execution and Evaluation.. When the SEL RTAC 3505 first starts up, it automatically enables the logic engine, so the first step was to stop the logic engine. This is a typical operation when system maintenance or reprogramming is needed. At this point, we launch Ettercap's ARP spoofing attack against the RTAC and the engineering workstation along with the developed packet filters. When an operator sends the command to start the logic engine, it is intercepted by the attacker, and the function code is modified or the packet is dropped before it reaches the RTAC, so the logic engine never starts.

When the logic engine fails to start due to this attack, even if multiple start commands are sent, the RTAC is not able to control or monitor the power system devices. If the voltmeter detects a high voltage, such as from a short circuit in the system, there is no longer a controller in place to open the circuit breaker. Therefore the power is allowed to flow and can reach critical devices such as transformers and damage or destroy these devices. This requires expensive repairs or equipment replacement and could result in a power outage. This state from our evaluation is shown in Figure 3b. As shown, the SEL RTAC is no longer reporting an up-to-date value for the voltage, and the breaker is not being opened when the voltage is above the threshold.

Once the Ettercap attack is stopped, an operator is able to restart the RTAC's logic engine. The RTAC would then be able to detect the high voltage from the voltmeter and open the circuit breaker, but in an operational power system with real-time requirements this action would be far too late to prevent damage to the system. Figure 3c shows the state after the RTAC logic engine has been enabled. Finally, the breaker has been opened, but this action is unable to prevent the major damage that would have already occurred in an operational power system.

5. Case Study II: Traditional PLCs

The case study involves the PLCs of two vendors: Schneider Electric's Modicon M221, and Allen-Bradley's MicroLogix 1400 & 1100. Unlike SEL RTAC (refer to Section 1.4), the PLCs do not have the security features to protect the communication and PLC device such as encryption and access-control.

5.1 Case Study II (a): Schneider Electric's Modicon M221

5.1.1 Controller Details. Schneider Electric Modicon M221 is a nano PLC made to control manufacturing processes. A control engineer can write the control logic, monitor the physical process and

control state of M221 using the vendor-provided engineering software, SoMachine Basic. SoMachine Basic supports two IEC-61131 languages, i.e., Ladder diagram (LD) and Instruction list (IL), to write the control logic. The engineer can download a control logic to the PLC (write on PLC's memory). The engineer can also start or stop the execution of control logic, i.e., logic engine on M221 via SoMachine Basic. The communication between M221 and SoMachine-basic is unencrypted and uses a proprietary protocol on port 502. The protocol is encapsulated in Modbus/TCP. M221 only allows one connection at a time.

5.1.2 Vulnerability. Other than the fact that the network communication between the M221 PLC and its engineering software is not encrypted, we exploit two more features of the PLC. 1) The PLC's state (which enables or disables it from running the control logic program) can be changed remotely via the engineering software. 2) The PLC only allows one engineering software to connect to it at a time.

5.1.3 MITRE ATT&CK. The case study utilizes the following attacks from the MITRE ATT&CK knowledge base.
Network Sniffing (T0842). This again is the first of the steps that the attacker takes in order to launch her final attack. It gives her the ability to sniff the network traffic between the engineering software, SoMachine Basic, and the PLC itself and then figure out interesting and important features (such as the protocol information) which could help her reach her target.
Unauthorised command message (T0855). Since the communication with the PLC is unencrypted, the attacker can send crafted messages to the PLC remotely. Using the protocol information derived as a result of the above mentioned step, she creates a message and sends it to the PLC that stops it from running the control logic program.
Loss of Availability (T0826). Since M221 allows only one machine to connect and communicate with it, the attacker can make it unavailable for control engineers by not closing the session it establishes as part of her previous attack.
Denial of Control (T0813). Similar to the above attack, in this case, the attacker does not close the session with the PLC which prevents the control engineer from interacting with the process control.
Manipulation of Control (T0831). As mentioned in the case of SEL RTAC, this attack is the ultimate goal of the attacker. She uses network sniffing as well as unauthorised command message attack to stop the controller from running the control logic program.

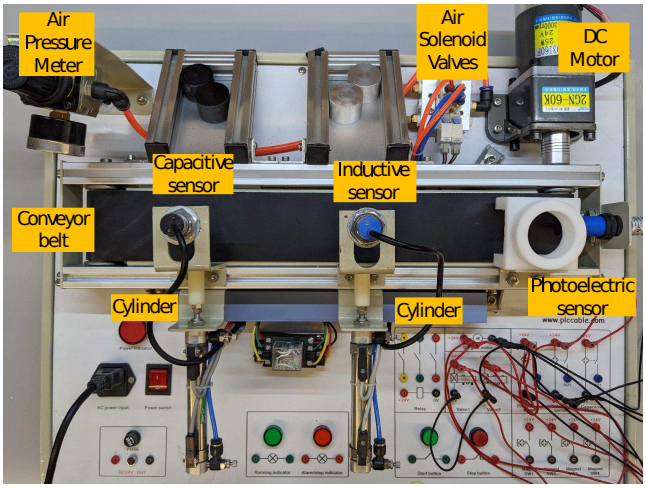
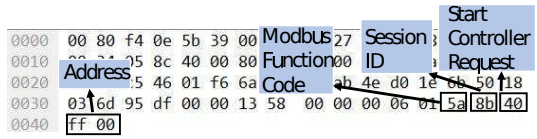


Figure 4: Top-view of the fully-functional conveyor belt model

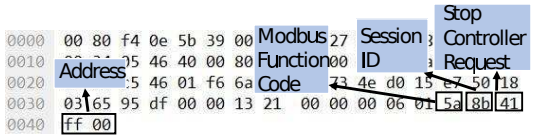
5.1.4 Attack Implementation. We used differential analysis and some manual efforts to reverse engineer the M221 proprietary protocol and identified the packets sent by the SoMachine-Basic software to start and stop the PLC logic engine. Figure 5a and 5b shows the packets. The function code `0x40` and `0x41` is used to start and stop the controller respectively. If the engineering software's request message is successful and accepted by the PLC, it sends back a success message to acknowledge the change as shown in Figure 5c. Using this information, we wrote a python script that first establishes a session with the Modicon M221 PLC and then sends crafted messages to start and stop the execution of control logic running on the PLC.

5.1.5 Experimental Settings. We evaluate the attack on a lab functional model of a real conveyor belt used in an industrial environment. Figure 4 shows the model details. The conveyor belt sorts different types of objects with the help of sensors and handles powers by air solenoid. The system is controlled by Modicon M221 PLC. The SoMachine Basic software runs on a Windows 7 Virtual Machine while the attack scripts run on Ubuntu 16.04 Virtual Machine. We assume the attacker has infiltrated the ICS network so the PLC, the engineering workstation and the attacker's machine are on the same network.

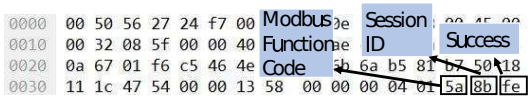
5.1.6 Attack Execution and Evaluation. Through network scanning, the attacker identifies the PLC's IP address and then launches



(a) Request message to "START" the Modicon M221 PLC



(b) Request message to "STOP" the Modicon M221 PLC



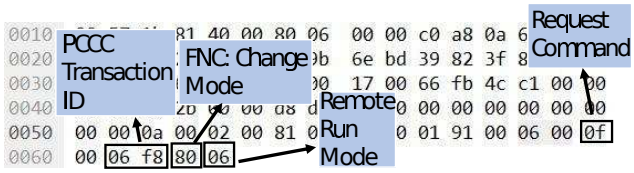
(c) Response from the Modicon M221 PLC with success function code

Figure 5: Messages sent to start and stop the Modicon M221 PLC

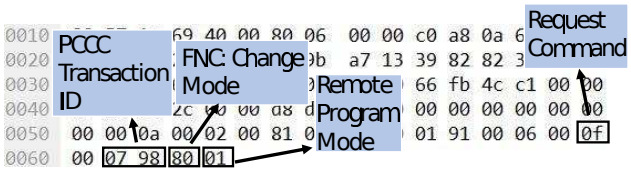
the attack program. The attacker first establishes a Modbus protocol session with the Modicon M221 PLC and then sends the 'Stop controller' request to the PLC. Upon receiving this request, the PLC stops executing the control logic, halting the physical process. The PLC's functionality of allowing only one connection at a time disables the control engineer to communicate with the PLC to run the 'Start controller' command as long as the attacker keeps the Modbus session running.

5.2 Case Study II (b): Allen-Bradley's MicroLogix 1400 & 1100

5.2.1 Controller Details.. Allen-Bradley MicroLogix 1400 and 1100 are from the MicroLogix family and have some similarities. Both the PLCs can be monitored and controlled by RSLogix 500 engineering software and use unencrypted PCCC protocol encapsulated in EtherNet/IP to communicate with RSLogix [24]. RSLogix supports ladder diagram (LD) to write the control logic. After connecting with the



(a) Request message to set the PLC to Remote-Run Mode



(b) Request message to set the PLC to Remote-Program Mode

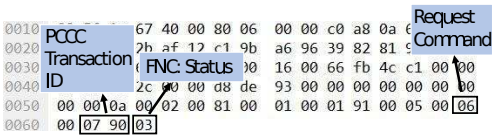
Figure 6: Messages sent to change the Mode of Micrologix 1400 and 1100 PLCs

PLC, the control engineer can upload the control logic (read the control-logic running on the PLC) or download the newly created logic on the PLC (write on PLC's memory). Both PLCs have three modes of operation; Run, Program, and Remote. In 'Run' mode, the PLC executes the control logic and controls the physical process. To do the maintenance or change the control logic, the user has to put the PLC in 'Program' mode to make any changes. The PLC's logic engine pauses while it is in 'Program' mode. Hence, it does not execute any control logic as long as it remains in this mode. The user can physically change the mode from the CLI interface provided on both of these PLCs. However, generally, for operational ease, it is placed in 'Remote' mode, which allows the control engineer to change the mode from 'Run' to 'Program' and vice versa remotely from the engineering software. Both the PLCs use unencrypted PCCC protocol encapsulated in EtherNet/IP to communicate with the engineering software.

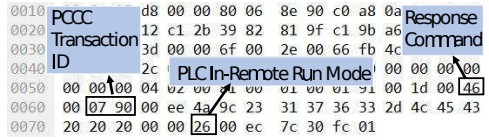
5.2.2 Vulnerability. We exploit an inherent functionality in MicroLogix PLCs to change operational modes to 'Run', 'Program', and 'Remote', where the PLCs in 'Program' do not execute control logic and wait for an operator to update their control logic and configurations.

5.2.3 MITRE ATT&CK. We employ the following attacks from the MITRE ATT&CK knowledge base for the case study on MicroLogix PLCs.
Network Sniffing (T0842). The attacker uses network sniffing to find the communication protocol between the PLC and its engineering software. She determines the packets responsible for changing the PLC state in order to halt the physical process from running.
Unauthorised command message (T0855). Using the information derived, she creates a well crafted message that can remotely change the PLC state from 'run' to 'program'. This would stop the control logic program from running.
Manipulation of Control (TO831). As a result of the above attack, the attacker is able to disrupt the control logic program from getting executed on the PLC which halts the physical process it controls.
Denial of Control (T0813). Since the attacker changes the state of the PLC from 'run' to 'program', she temporarily prevents the control engineers from interacting with the process controls.
Man in the Middle (TO830). Along with changing the state of the PLC and halting the physical process, the attacker also wishes to hide this change of state from the control engineer. Hence, she poisons the ARP cache of the engineering software and the PLC and then positions herself as man-in-the-middle between her targets to modify the PLC state from 'program' to 'run' when the engineering software requests to read the state.
Denial of View (T0815). The above Man in the Middle attack deceives the control engineer who now assumes the PLC is still in 'Run' mode controlling the physical process while, in reality, the process has been halted.

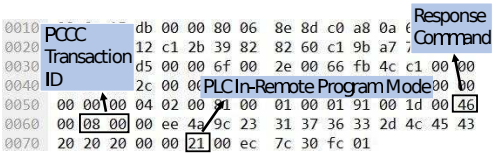
5.2.4 Attack Implementation. Through manual reverse engineering, we successfully identified the request messages sent by RSLogix to change the mode of the PLC to Remote-Run or Remote-Program. Figure 6a and 6b show the messages sent to put the PLC in Remote-Run and Remote-Program mode. As shown in the figures, the function code `0x80` is used to change the mode. The function code is followed by `0x01` for Remote-Run mode or `0x06` for Remote-Program. We also found that the RSLogix software periodically inquires about the status of the PLC. As shown in figure 7a, the engineering software sends a request message with function code `0x03` to inquire about the status of the PLC. The differential analysis of the response messages during Remote-Run and Remote-Program mode shows that a function code of `0x21` is used for Remote-Run mode (fig. 7b) while `0x26` is used for Remote-Program



(a) Request message to sent the PLC to inquire current status



(b) Response message from the PLC when in Remote-Run mode



(c) Response message from the PLC when in Remote-Program mode

Figure 7: Periodic status inquiry and response from MicroLogix 1400 and 1100

mode (fig. 7c). Using this information, we developed a program that initiates an ENIP session with the target PLC and sends the mode-change messages to put the PLC in Remote-Program mode, which stops the execution of control logic on the PLC. Since the RSLogix software periodically inquires about the status, the change in mode can be detected by the control engineer. Thus, to deceive the RSLogix software, we developed an Ettercap filter that detects the status response message and changes Remote-Program's function code to Remote-Run.

5.2.5 Experimental Settings. Since both Micrologix 1400 and 1100 use the same communication protocol and function codes, we tested the logic engine attack on Micrologix 1400 connected with the lab functional model of an elevator. The elevator model has four floors and operates like a real elevator as shown in figure 8. A user can select a floor both from inside and outside the elevator as an input to the PLC. In response, the PLC moves the elevator to the desired floor. RSLogix 500 can communicate with the PLC. It runs on a Windows 7 Virtual

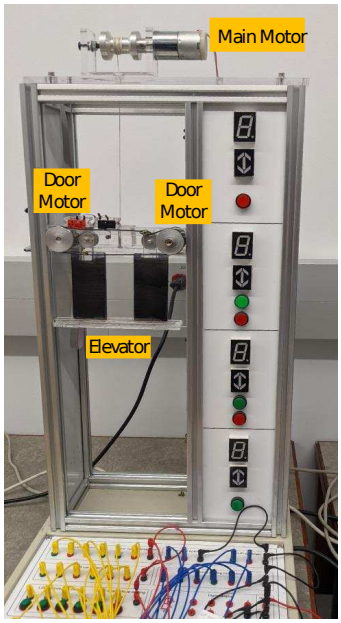


Figure 8: Front-view of the fully-functional elevator model

Machine (engineering workstation). The attacker uses a Ubuntu 18.04.3 LTS machine. Like the previous experiments, the PLC, engineering workstation, and attacker machine are in the same network.

5.2.6 Attack Execution and Evaluation. The attacker first performs a man-in-the-middle using ARP poisoning and establishes an ENIP session with the Micrologix 1400 PLC controlling the elevator. The attacker then sends a request message to the PLC to change the mode to Remote-Program, due to which the PLC stops executing the control logic, resulting in halting the elevator operation. To prevent the control engineer from knowing the current status of the PLC, the attacker launches the Ettercap filters explained in the previous section. The filter changes the Remote-Program function code to Remote-Run. In this way, the control engineer is unaware of the attack while the elevator is no longer operating.

6. Mitigation

The main issue for most of the PLCs is that the communication is unencrypted. While RTAC still employs TLS for most of its communication, the one that occurs on port 1217 is still not encrypted which makes it easier for attackers to reverse engineer the protocol and launch an attack of their choice.

PLCs like Modicon M221 and Micrologix 1400 have some level of security in the form of passwords. These PLCS use password authentication schemes to protect the control logic from being read and, in some cases, written by unauthorised users. Unfortunately, the attackers can change the PLCs' state to prevent the control logic from running without the need for authentication. It is, therefore, suggested to employ passwords for changing the state of the PLC as well.

Moreover, M221 PLC also has an intrinsic default feature that allows unauthorised users to connect to it without the need for authentication. It also only allows one user to connect to it at a time. This leaves a room for attacks that require initiating a successful connection with the PLC such as the one we discussed in our study. The attacker connects with the PLC, stops the controller from running the control logic program and then keeps the session active to disallow legitimate field engineers from taking control of the PLC. In order to prevent these kinds of attacks, password protection should be employed for connecting to the PLC as well.

Man in the middle attacks can be prevented by techniques like DHCP snooping and ARP inspection [3].

7. Conclusion

We presented a new dimension of control-logic attacks on the PLCs. Instead of injecting a malicious control-logic into a PLC, our attacks targeted the IEC-61131 control-logic engine responsible for executing a PLC control logic. We successfully employed the attacks from MITRE ATT&CK knowledge base to demonstrate our control logic engine attacks as case-studies on four real PLCs, i.e., SEL-3505 RTAC, Modicon M221, and MicroLogix 1400 and 1100. The case studies showed a real-world impact of halting three physical processes during operation, i.e., power substation, conveyor belt, and elevator, and facilitate the ICS research community and industry to understand the control logic engine's attack vectors.

Acknowledgement

This work was supported, in part, by the Virginia Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development. For more information, visit `www.cyberinitiative.org`.

This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article

References

[1] Ettercap (www.ettercap-project.org), 2021.

[2] ATT&CK® for Industrial Control Systems,The MITRE Corporation, (collaborate.mitre.org/attackics/index.php/Main_Page),2021

[3] H. Adjei, T. Shunhua, G. Agordzo, Y. Li, SSL Stripping Technique "(DHCP Snooping and ARP Spoofing Inspection)", *Twenty Third International Conference on Advanced Communication Technology (ICACT)*, pp. 187–193, 2021.

[4] I Ahmed,S. Obermeier, S. Sudhakaran and V. Roussev, Programmable Logic Controller Forensics, *IEEE Security Privacy*, vol. 15(6), pp. 18–24, 2017.

[5] I. Ahmed, S. Obermeier, M. Naedele, G. Richard III, SCADA Systems: Challenges for Forensic Investigators, *Computer*, vol. 45(12), pp. 44–51, 2012.

[6] I. Ahmed, V. Roussev, W. Johnson, S Senthivel and S. Sudhakaran, A SCADA System Testbed for Cybersecurity and Forensic Research and Pedagogy, *Proceedings of the Second Annual Industrial Control System Security Workshop*, pp. 1–9, 2016.

[7] A. Ayub, H. Yoo, I. Ahmed, Empirical Study of PLC Authentication Protocols in Industrial Control Systems, *Fifteenth IEEE Workshop on Offensive Technologies (WOOT)*, 2021.

[8] S. Bhatia, S. Behal and I. Ahmed, Distributed Denial of Service Attacks and Defense Mechanisms: Current Landscape and Future Directions, *Versatile Cybersecurity*, M. Conti, G. Somani and R. Poovendran (Eds.), Springer International Publishing, Heidelberg Germany, pp. 55–97, 2018.

[9] T. Chen and S. Abu-Nimeh, Lessons from Stuxnet, *Computer*, vol. 44(4), pp. 91–93, 2011.

[10] Exe-GUARD (www.energy.gov/sites/prod/files/2017/04/f34/SEL_Exe-guard_FactSheet.pdf)

[11] N. Falliere, L. Murchu and E. Chien, W32.Stuxnet Dossier, Version 1.3 ,Symantec Corp, Cupertino, CA, USA, 2011.

[12] L. Garcia, F. Brasser, M. Cintuglu, A. Sadeghi, O. Mohammed, S. Zonouz, Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit, *Twenty Fourth Annual Symposium on Network & Distributed System Security(NDSS)*, pp. 8:1–8:15, 2017.

[13] N. Govil, A. Agarwal, N. Tippenhauer, On Ladder Logic Bombs in Industrial Control Systems, *Computer Security*, S. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis, C. Kalloniatis, J.Mylopoulos, A. Anton and S. Gritzalis (Eds), Springer International Publishing, Heidelberg, Germany, pp. 110–126, 2018.

[14] R. Johnson, Survey of SCADA security challenges and potential attack vectors, *2010 International Conference for Internet Technology and Secured Transactions*, pp. 1–5, 2010.

[15] S. Kalle, N. Ameen, H. Yoo, I. Ahmed, CLIK on PLCs! Attacking Control Logic with Decompilation and Virtual PLC, Binary Analysis Research(BAR) Workshop, *Workshop on Binary Analysis Research (BAR), Proceeding of Twenty-Sixth Network and Distributed System Security Sym-posium*, 2021.

[16] N. Kush, E. Foo, E. Ahmed, I. Ahmed, A. Clark, Gap analysis of intrusion detection in smart grids, *Second International Cyber Resilience Conference(ICR 2011)*, pp. 38–46, 2011.

[17] Schweitzer Engineering Laboratories Incorporation, SEL-3505/SEL-3505-3 Real-Time Automation Controller, Pullman, WA, USA, (`selinc.com/products/3505/docs/`), 2020.

[18] S. Mclaughlin, P. McDaniel, SABOT: Specification-based Payload Generation for Programmable Logic Controllers, *Proceedings of the 2012 ACM conference on Computer and communi-cations security*, 2012.

[19] S. Qasim, J. Lopez, I.Ahmed, Automated Reconstruction of Control Logic for Programmable Logic Controller Forensics, in *Information Security*, Z. Lin, C. Papamanthou, M. Polychronakis, Cham, pp. 402–422, 2019.

[20] S. Qasim, J. Smith, I. Ahmed, Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems, *Forensic Science International: Digital Investigation*, vol. 33, pp. 301013, 2020.

[21] M. Rais, Y. Li, I. Ahmed, Spatiotemporal G-code Modeling for Secure FDM-based 3D Printing, in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, Associ-

ation for Computing Machinery, New York, NY, USA, 177–186, 2021.

[22] M. Rais, R. Asmar, J. Jr, I. Ahmed, JTAG-based PLC Memory Acquisition Framework for Industrial Control Systems, *Twentieth Annual Digital Forensics Research Conference*, 2021.

[23] C. Schuett, J. Butts, S. Dunlap, An evaluation of modification attacks on programmable logic controllers, *International Journal of Critical Infrastructure Protection*, vol. 7(1), pp. 61–68, 2014.

[24] S. Senthivel, I. Ahmed, V. Roussev, SCADA network forensics of the PCCC protocol, *Digital Investigation*, vol. 22, pp. S57–S65, 2017.

[25] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, V. Roussev, Denial of Engineering Operations Attacks in Industrial Control Systems, *CODASPY '18: Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pp. 319–329, 2018.

[26] R. Sun, A. Mera, L. Lu, D. Choffnes, SoK: Attacks on Industrial Control Logic and Formal Verification-Based Defenses, *ArXiv*, vol. abs/2006.04806, 2020.

[27] H. Yoo, I. Ahmed, Control Logic Injection Attacks on Industrial Control Systems, in *ICT Systems Security and Privacy Protection*, F. Karlsson, K. Hedström, A. Zúquete, Springer International Publishing, Cham, pp. 33–48, 2019.

[28] H. Yoo, S. Kalle, J. Smith, I. Ahmed, Overshadow PLC to Detect Remote Control-Logic Injection Attacks, in *Detection of Intrusions and Malware, and Vulnerability Assessment*, R. Perdisci, C. Maurice, G. Giacinto, M. Almgren, Springer International Publishing, Cham, pp. 109–132, 2019.