Proceedings of the Singapore Cyber-Security Conference (SG-CRC) 2016
A. Mathur and A. Roychoudhury (Eds.)

doi:10.3233/978-1-61499-617-0-75

Attacking Fieldbus Communications in ICS: Applications to the SWaT Testbed

David URBINA [a,1], Jairo GIRALDO [a], Nils Ole TIPPENHAUER [b], and Alvaro CARDENAS [a]

[a] *University of Texas at Dallas*
[b] *ISTD, Singapore University of Technology and Design*

Abstract The study of cyber-attacks in industrial control systems is of growing interest among the research community. Nevertheless, restricted access to real industrial control systems that can be used to test attacks has limited the study of their implementation and potential impact. In this work, we discuss practical attacks applied to a room-sized water treatment testbed. The testbed includes a complete physical process, industrial communication systems, and supervisory controls. We implement scenarios in which the attacker manipulates or replaces sensor data as reported from the field devices to the control components. As a result, the attacker can change the system state vector as perceived by the controls, which will cause incorrect control decisions and potential catastrophic failures. We discuss practical challenges in setting up Man-In-The-Middle attacks on fieldbus communications in the industrial EtherNet/IP protocol and topologies such as Ethernet rings using the Device-Level-Ring protocol. We show how the attacker can overcome those challenges, and insert herself into the ring. Once established as a Man-in-the-Middle attacker, we launched a range of attacks to modify sensor measurements and manipulate actuators. We show the efficacy of the proposed methodology in two experimental examples, where an adversary can intelligently design attacks that remain undetected for a typical bad-data detection mechanism.

Keywords. ICS, Critical System, Cyber-attacks, Fieldbus

1. Introduction

In recent years, security threats to industrial control systems (ICS) have received an increasing amount of attention [7,6,15,8,10]. One area that has received particular attention are attacks against the integrity of sensor and control data in the system, also known as false data injection attacks [12,16,19,14]. If the attacker manipulates or replaces sensor data reported from the field devices, the control algorithm will take actions based on an incorrect perception of the world, which will cause incorrect control decisions and potential catastrophic failures.

So far, most of the analysis on false data injection has focused on a wide range of suspected theoretical attacks and simulation studies, and only limited work has been

[1] Corresponding Author: David Urbina, University of Texas at Dallas, 800 Campbell Rd., Richardson, 75080, TX; E-mail: david.urbina@utdallas.edu

published on the practical challenges for launching such attacks as access to real-world ICS is usually hard to obtain for security researchers.

In this work, we discuss practical attacks applied to a room-sized water treatment testbed (the SWaT testbed). The Secure Water Treatment (SWaT) testbed includes a complete physical process, the related industrial communication infrastructure, and a supervisory control network. This paper is an extention of our previous work [9], where we launched Man-in-the-Middle (MitM) attacks at the supervisory control layer of the SWaT testbed. In that work, we attacked the physical system using Ettercap spoofing on the Supervisory Control network level, which enabled us to modify the sensor information. In addition, we noted that Ettercap-based spoofing can easily be detected by more complex networking devices such as SDN-capable switches and their controllers.

In contrast to [9], our proposed methodology attacks fieldbus communications directly, which enables the adversary to have total control of the system operation, without taking into account any command coming from the higher levels such as the Human-Machine Interface (HMI). Such attacks on fieldbus communications are challenging due to the ring topology and the device specific messages used in the fieldbus, requiring more specific knowledge of the network operation. We discuss practical challenges in setting up MitM attacks on a fieldbus communication network using the EtherNet/IP protocol over an Ethernet ring (maintained with the Device-Level-Ring (DLR) protocol). Once established as a MitM attacker, we are able to launch a range of sensor and actuator attacks and demonstrate their efficacy.

We summarize our contributions as follows:

- We study fieldbus communications and implement a prototype Man-in-the-Middle (MitM) attack.
- We show how a MitM attacker can obtain sensor readings from eavesdropped packets, and craft her own spoofed sensor and actuator command traffic. We also provide details on datatypes and conversions required (e.g., from 4-20mA signal to physical measurements).
- We combine the MitM attack and detailed understanding of the EtherNet/IP protocol to demonstrate practical attacks on SWaT, and show their impact on the physical process.

This work is structured as follows: in Section 2, we state the problem setting and attacker model. In Section 3, we present our practical attacks and discuss them in detail. We conclude the paper in Section 4.

2. Background and Problem Formulation

In this section, we introduce Industrial Control Systems (ICS), their fieldbus networks, and the specific testbed we use for our experiments. Finally, we introduce our attacker model.

2.1. The SWaT Physical Process

The SWaT testbed is a water treatment plant which consists of 6 main processes to purify raw water. Each process possesses a PLC that receives the information from the sensors

and compute the control actions to the actuators. SWaT is set up to have two different communication channels for many links: either wired (over IEEE 802.3 Ethernet) or wireless communications (using IEEE 802.11).

The main physical processes SWaT can be described as follows (see Figure 1):

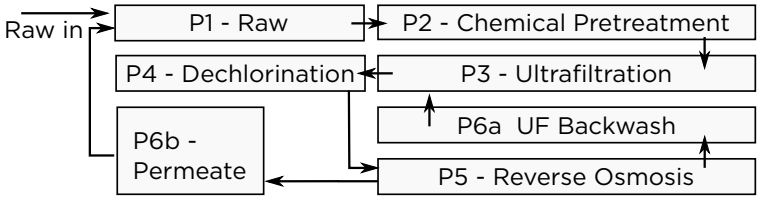

Figure 1. SWaT physical process overview

Raw water (P1) In this process, raw water is stored. P1 acts as the main water buffer supplying water to the water treatment system. It consists of one tank, an on/off valve that controls the inlet water, and a pump that transfers the water to the Ultra Filtration (UF) system's tank.

Pre-treatment (P2) While the water from P1 is pumped to the UF system, water quality properties are evaluated and pre-treated. Conductivity, pH, and ORP are measured to determine the activation of chemical dosing to maintain the quality of the water within desirable limits.

Ultra Filtration (P3) The ultra-filtration process is used to remove the bulk of the feed water solids and colloidal material by using fine filtration membranes that only allow the flow of small molecules. The accumulated contaminants are removed by back-washing away the membrane surface depending on the measure of a differential pressure sensor located at the two ends of the UF.

Dechlorinization (P4) After the small residuals are removed by the UF system, the remaining chlorines are destroyed in the ultraviolet chlorine destruction unit and by dosing a solution of sodium bisulphite.

Reverse Osmosis (P5) The RO system is designed to reduce inorganic impurities by pumping the filtrated and dechlorinated water with a high pressure through semipermeable membranes.

RO final product (P6) The last part of the water treatment process consists on storing the RO product i.e., cleaned water ready to distribute. In the SWaT case, treated water is transferred again to the raw water tank in order to reuse it.

The SWaT testbed features distributed controls among the different process stages. Each stage is controlled by a PLC (with hot-redundant counterpart), and all PLCs and the SCADA system are connected together through a common network (which we call Level 1 (L1) network). In addition, the PLCs are connected to local sensors and actuators through individual fieldbus rings called Level 0 (L0). We now give details of this network architecture.

2.2. Industrial Control Network

A modern industrial control system typically consists of several layers of networks. The SWaT industrial control network is illustrated in Figure 2. The physical process is measured by distributed sensors, and manipulated by actuators. These sensors and actuators in SWaT operate by receiving and sending analog signals (4mA). The analog signals are converted into digital signals by Remote Input/Output (RIO) modules. The digital signals are then encapsulated over *fieldbus* communication protocols (EtherNet/IP in our case over the L0 network Figure 2), and sent back and forth from PLCs. PLCs in turn communicate with a centralized Supervisory Control and Data Acquisition (SCADA) system with the L1 network in Figure 2. This central system contains the HMI and Historian. In this work, we want to show a systematic methodology to deploy cyber-attacks on industrial control systems (ICS), with a focus on fieldbus communication networks.

The reliability of such fieldbus networks is of great concern to the plant operator. For that reason, ring topologies are a popular choice to implement these topologies. The ring topology can be seen in Figure 2 at the L0 network, where there is a ring between the RIO and a primary and a backup PLC. In particular, rings can tolerate faults such as the loss of a single ring segment, without losing connectivity between any of the participating devices. If the communication uses Ethernet as medium, rings can be constructed using the device-level-ring (DLR) protocol.

In the context of an attacker who tries to insert itself as MitM, such ring topologies have interesting properties. In particular, if the attacker cuts the ring to insert her own device, the ring will automatically stop transmitting data through the "lost" segment. As a result, the attacker will not receive any traffic to eavesdrop on. In order to successfully complete the attacker, the attacker must "close" the ring again. We discuss that further in Section 3.2.2.

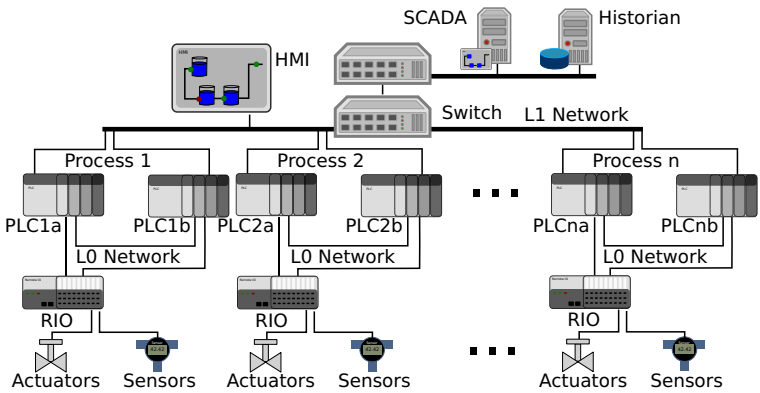

Figure 2. SWaT network architecture.

2.3. Fieldbus Communications

SWaT's ring topology in the fieldbus network contains the following four main devices (see Figure 3).

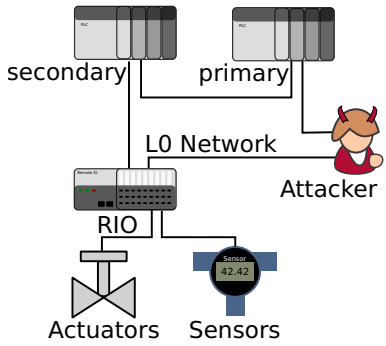

Figure 3. Example ring topology in SWaT, with an attacker inserted as Man-in-the-Middle. In this configuration, the attacker can eavesdrop and manipulate all traffic between RIO and primary PLC.

Programmable Logic Controller (PLC) This is the control device which receives sensors readings and emits control commands to the actuators. SWaT's PLCs correspond to a chassis conformed by 6 components: 1756-PA2 Control Logix AC Power Supply Unit; 1756-L71 Control Logix 2MB Controller; 1756-EN2T Ethernet I/P Module (for communications at the Supervisory Control Network level); 1756-EN2TR Dual Port Ethernet I/P Module (for fieldbus communications at the ring level); 1756-RM2 Control Logix Redundancy Module; and a 1756-RMC1 Control Logix Redundancy Fiber Optic Cable. Each SWaT's ring presents a redundant PLC, and on power up the system selects one as Primary, actively controlling the physical process, and other as Secondary, shadowing the memory state of the Primary. From that moment on, the Primary is responsible for the monitoring and control of the fieldbus communications and the relaying/receiving of data from the SCADA network. If the Primary fails, the system automatically switches over the control to the Secondary.

Remote I/O (RIO) This device is the responsible for the translation of 4-20 mA signals to/from the actuators/sensors to a stream of bytes where each byte (or bit depending on the resolution of the I/O module) corresponds to an I/O signal of the system. For the communications between the PLC and RIO, the stream is encapsulated following the Common Packet Format of the EtherNet/IP specification [18] and transported through a wired connection. The RIO is modular, and in SWaT it consists of 4 components: 1794-AENTR Flex Dual Port Ethernet I/P Adapter (for fieldbus communications at the ring level); 1794-IB32 Flex 32 Points Digital Input Module (1-bit resolution per signal); 1794-OB16 Flex 16 Points Digital Outputs Module (1-bit resolution per signal); and 1794-IE12 Flex 12 Points Analog Input Module (16-bit 2's complement per signal).

Wireless Remote I/O (WRIO) SWaT features a manual switch to turn parts of the system into "wireless mode". If the field communications at the ring level are set to wireless mode, this WRIO takes over the responsibility of scanning the analog sensors and sending updates to the PLC. Its is important to highlight that in SWaT this wireless transmission contains exclusively the analog input signals and not the digital I/O signals. The Digital input and output signals are always transmitted through the wired ring network between the RIO and PLC.

Wireless Access Point (WAP) In wireless mode, the WRIO connects to a wireless access point connect to the EtherNet/IP ring thought a 1783-ETAP 3-ports switch.

2.4. Attacker Model

Objective. The objective of an attacker depends on how much damage she wants to cause to the system. Our proposed methodology can be used to a) manipulate sensor readings that are reported from the sensors to the PLCs, and b) to manipulate control messages that are sent from the PLC to the actuators. Therefore, a wide number of attacks can be deployed, starting with a simple eavesdropping to sophisticated integrity persistent and stealthy attacks.

Resources. The attacker is assumed to either a) being able to physically access the network connecting the PLCs, sensors, and actuators, or b) able to fully compromise one of the devices attached to that network. In both cases, the attacker will have a device attached to the network, and can program the device to transmit arbitrary messages to the network, and to process any message it receives.

3. Attacking Fieldbus Communications in SWaT

Based on the details we provided on SWaT, its fieldbus topologies, and the EtherNet/IP protocol, we now show results of practical MitM attacks. We start by introducing tools we used, and among them our custom *SWaT Assault* tool.

3.1. Tools

We used several tools to launch attacks against the fieldbus communications at the SWaT testbed:

SWaT Assault We developed a command-line interpreter (CLI) application which includes a library of attack modules capable of launching diverse spoofing and bad-data-injection attacks against the sensor and actuator signals of the SWaT testbed. The attack modules can be loaded, configured, and run independently of each other, allowing the attack of sensors and actuators separately. Attack modules also can be orchestrated and assembled in teams in order to force more complex behaviors over the physical process, while maintaining a normal operational profile on the HMI. SWaT Assault consists of 439 lines of Python [3] 2.7 code and its only external dependencies are Scapy and NetFilterQueue.

Scapy Making use of the Scapy[4] packet manipulation program we developed a new protocol parser for the Rockwell Automation proprietary message protocol used for signal communication between the RIO and the PLC, and for the EtherNet/IP Common Packet Format wrapper that encapsulates it. This parser (which we chose to call SWaT message parser) is specific for the SWaT's deployment (the SWaT Ring implementation makes use of User Datagram Protocol (UDP) for the transport of EtherNet/IP I/O implicit messages among ring devices) and its implementation follows SWaT's Control Panels and Electrical Drawings manual. Scapy was also used to sniff sensor readings from the EtherNet/IP Ring and to inject manipulated data on both, sensor readings and actuation commands. Our tool also automatically recomputes the data integrity checksums used by the Transport Layer protocol to match the false-data injection attack values.

NetFilterQueue In order to avoid duplication of packets and/or race conditions between original and injected packets, we employed the NetFilterQueue [2] Python bindings for libnetfilter_queue to redirect all the EtherNet/IP I/O messages between PLC and RIO to a handling queue defined on the *mangle* table of the Linux firewall *iptables*. The queued packets are later modified using Scapy and the previously mentioned SWaT message parser, and finally released to reach their original destination i.e. PLC or RIO. Likewise, this technique allowed us to avoid disruptions on the sequence of EtherNet/IP counters, and injection of undesirable perturbations in the EtherNet/IP connections established between ring devices.

The command we use to queue packets for modification is the following:

```
iptables -t mangle -A PREROUTING -p udp --dport <port> -j NFQUEUE
```

Wireshark We used Wireshark [5] to understand the nature of the communication between devices in the ring. We also used Wireshark together with the SWaT's Control Panel and Electrical Drawings manual, to derive the exact structure of the EtherNet/IP-wrapped messages used in SWaT.

Ettercap We used Ettercap [1], a Man-In-The-Middle attack suite, on our attempts to launch wireless attacks.

3.1.1. Differences Between Parsing Supervisory Network Packets vs. Fieldbus Packets

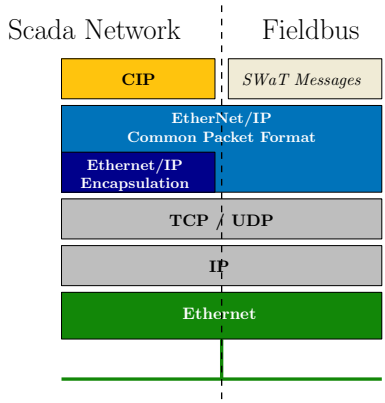

Figure 4. EtherNet/IP encapsulation for CIP and custom *SWaT Messages* for Supervisory Network and Fieldbus communications, respectively.

While the SWaT networks use EtherNet/IP at the supervisory as well as the fieldbus level, the encapsulated protocol is different at each level: the CIP protocol is used as main data payload for device communications at the Supervisory Control Network level, while a device-dependent I/O implicit message payload is employed at the Fieldbus Communications level (see Figure 4).

Parsing and injection of manipulated data at the Supervisory Control level using CIP messages or at the Fieldbus Communications level using EtherNet/IP device-dependent I/O implicit messages introduce different challenges and requirements to the attacker.

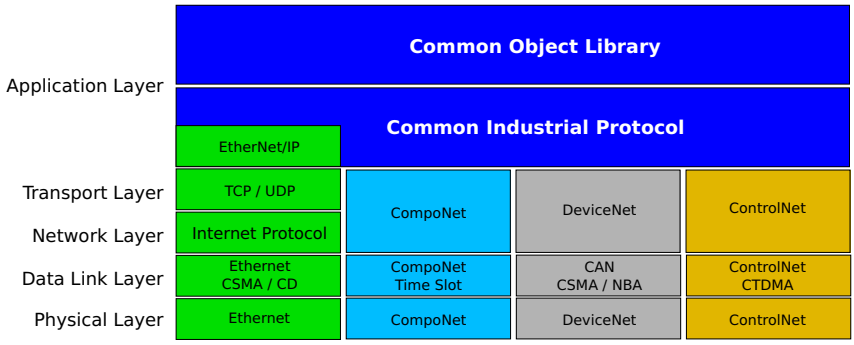

Figure 5. Common Industrial Protocol stack and its different physical layers.

Common Industrial Protocol stack The Common Industrial Protocol (CIP) network specification library was originally developed by Rockwell Automation and finally standardized and maintained by Open Device Vendors Association (ODVA) and ControlNet International. It aims to fulfill the main three needs of ICS systems: control, configuration, and collection of data [11]. It defines the CIP application layer protocol as a encapsulated object-oriented protocol for transmission of connected (I/O implicit) messages between a data producer and one or more data consumer devices, and unconnected (explicit) messages between two devices in the control network. Transmissions associated with a particular connection are assigned an unique *connection ID*. While being an application layer protocol, CIP is independent of the underlying layers, and requires an encapsulation protocol which allows abstraction from different data link and physical layers. It also includes a Common Object library defining commonly used objects, some of which are specific for a particular encapsulation protocol, and allows for extension and definition of vendor specific objects. The CIP specification library includes the definition of 4 different CIP stacks depending of the physical layer in use (see Figure 5): EtherNet/IP (over IEEE 802.3 Ethernet), CompoNet, DeviceNet, and ControlNet.

Therefore, CIP messages contain much more rich semantic information about the information being exchanged in the network, which can facilitate the understanding of the process by the attacker. CIP messages follow an object-oriented format, allowing for the transmission of a variable number of data with distinct types, which translates into highly structured packets of variable lengths (see Figure 6). The attacker must dissect each particular packet to extract the CIP object attributes containing the sensor or actuator data, which may be set at different offsets depending on the number and type of objects targeted by the packet.

On the other hand at the fieldbus level of SWaT, EtherNet/IP device-dependent I/O implicit messages follow non-standard formats of fixed lengths, partially defined by the vendor and by the control system designer, and where the analog sensor signals are encoded using 4-20 mA measurements. The attacker therefore must have detailed knowledge and understanding of the system design a priori and implementation decisions, i.e. she must have access to the devices specifications, electrical drawings, and installation layouts in order to understand the information exchanged and manipulate the sensor readings and control commands.

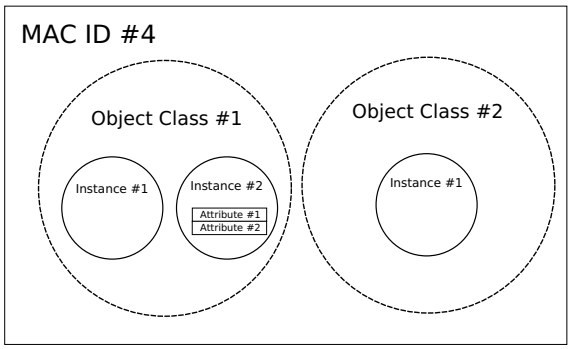

Figure 6. CIP Object-Oriented Packet Structure.

3.2. MitM of Fieldbus Communications

We attempt attacks on the wired and wireless mode fieldbus communications in SWaT. We now present results for both cases.

3.2.1. Wireless MitM

On our first try to sniff the fieldbus communications in the SWaT testbed we attempted a wireless MitM attack between the PLC and the WRIO. We set the fieldbus communications into *wireless mode* and use a laptop we connected to the WAP. Using Wireshark, we realized that we could already see one multicast EtherNet/IP connection, stacking EtherNet/IP over UDP, with the WRIO's IP as source. This multicast connection corresponded to the analog input signal which, as by SWaT's design, is the only signal transported in wireless mode. After switching on and off the wireless mode multiple times, we verified that the multicast address range corresponded to the *Organizational Local Scope* [13], as expected. The assigned UDP destination port was 2222, also defined in SWaT's design.

It is important to highlight that the attacks presented on section 3.4 could be achieved with minimum technical requirements through a *wireless MitM* if the ICS's design accounts for a complete wireless transmission of digital and analog signals. Unfortunately, the digital input and output are not reported via the wireless links, and thus cannot be compromised using the wireless MitM attack. In order to cope with this characteristic of SWaT's design, we resorted to performing a *wired MitM* directly in the EtherNet/IP ring.

3.2.2. Wired MitM

We assume an attacker who is an insider or who is able to set a physical device at any point of the EtherNet/IP ring, between the PLC and the RIO. In our experiments, for the implementation of the wired MitM, we intercepted the fieldbus communications by adding a laptop with two Ethernet ports in a segment of the EtherNet/IP ring.

DLR must be taken into consideration when attempting a MitM in the EtherNet/IP ring. If the attacker uses a DLR-unaware device, as we did in our experiments, she must disable MAC learning and Spanning Tree Protocol when bridging both Ethernet ports. Failing in carefully addressing this requirement will result in the isolation of the EtherNet/IP ring segment, as the DLR supervisor will recognize it as broken, and ultimately, in the inability to sniff and inject data into the ring messages.

Our configuration for the Ethernet ports and bridge for launching the attacks is the following:

```
auto eth0                       # Port 0
iface eth0 inet manual

auto eth1                       # Port 1
iface eth1 inet manual

# Bridge between Port 0 and Port 1
auto br0
iface br0 inet manual
        bridge_ports eth0 eth1
        bridge_stp off          # Disabling STP
        bridge_ageing 0         # Disabling MAC learning
```

3.3. Parsing and injection of manipulated data in EtherNet/IP ring packets

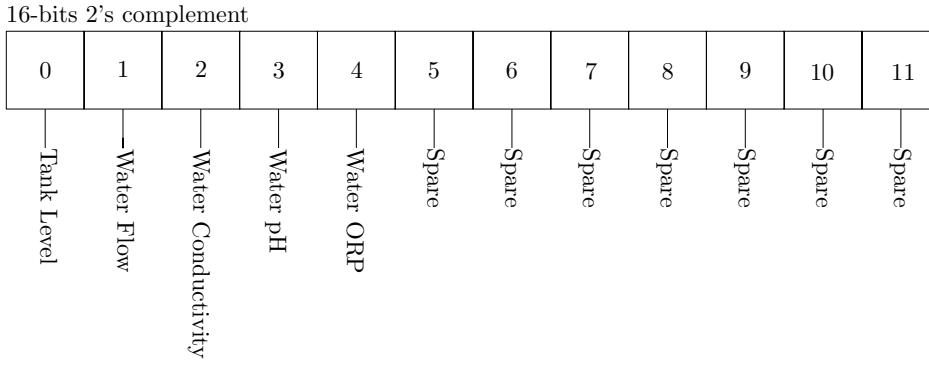

Figure 7. RIO's Analog Input Module 12 input signals (16-bits 2's complement per analog signal)

In SWaT, we identify three different device-dependent I/O implicit messages: one for each I/O module conforming the RIO. Figure 7 shows the I/O implicit message for the analog input module. It consists of a stream of 24 bytes, corresponding to 12 analog inputs channels of 16-bits. The *spare* channels are not in use by SWaT's current deployment. The digital input and output modules emit and receive bit streams, 32 bits and 16 bits respectively, where each bit corresponds to one digital signal. See table 1 for details on RIO modules.

Table 1. RIO I/O modules.

Module	Signal size (bits)	# signals	Avg. Freq. (ms)
Digital Input	1	32	50
Digital Output	1	16	60
Analog Input	16	12	80

The I/O implicit messages representing the analog signals are sent by the RIO to the PLC with an average frequency of 80 milliseconds. They transport the numeric represen-

tation of the 4-20 mA signals measured by the analog sensors. In order to scale back and forth the 4-20 mA signal to the real physical measurement we use Equation (1), which is a typical linear transformation (scaling and bias shift) to change the analog signal (4-20mA) into a physical meaningful quantity (e.g. the height of the water level in a tank being 0.5m). The constant values (*RawMin, RawMax, EUMax, EUMin*) depend on the deployment and the physical property being measured (we obtained the specific values for each constant from the HMI software of the testbed). Figure 8 shows an example for the scaling of the water flow in SWaT.

$$Out = (In - RawMin) * \frac{EUMax - EUMin}{RawMax - RawMin} + EUMin \tag{1}$$

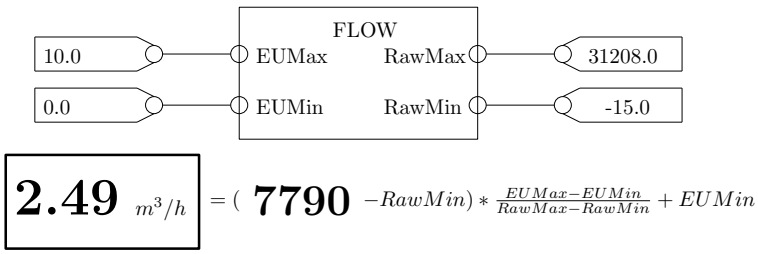

Figure 8. Scaling from 4-20 mA signals to water flow. The 4-20mA signal is scaled by the RIO to another value (7790 in this case) and this value sent over the network is the one we capture and convert to physical observations using Equation (1) with the respective constants for each signal.

3.4. Example: Stealthy Sensor Attack

We illustrate the feasibility of our proposed methodology by deploying a stealthy sensor attack in the first stage of the plant i.e., raw water storage. The raw water process consists of a storage tank with its water level sensor h_1, one valve that opens when $h_1 < 0.5\ m$ and closes when $h_1 > 0.8\ m$, and one pump whose action depends on the UF process. As a safety mechanism, if the water level in tank 1 is below $0.25\ m$, the pump is immediately Off. The attacker's goal is to overflow the water without being detected by a typical behavior-based detection mechanism using the "physics" of the system under to control to identify anomalies. Using the methodology described above, we gain access to the ring and we are able to modify the sensor and actuator information by constructing appropriate packets.

3.4.1. Attacker Action

Figure 9 depicts how an attacker can gain access to the sensor measure $h_i(k)$ and actuator command $u_i^{nom}(k)$ packets and modify them. In this example, the attack consists on injecting false information to the level sensor in tank 1. In particular, data injected is computed using $h_1^a(k) = h_1(k) + \delta(k)$. As result of this attack, the level of the water is decreasing all the time, i.e., $\delta(k) - \delta(k-1) = \Delta < 0$ is the rate at which the sensor information is modified.

3.4.2. Detection Mechanism

The detection mechanism also known as bad-data detection uses the residuals $r_i(k)$, which consists on the difference between the sensor measure $h_i(k)$ and its estimated $\hat{h}_i(k)$, such that $r_i(k) = |h_i(k) - \hat{h}_i(k)|$. For a sensor attack occurring at a time instant k^*, the residuals are then $r_i(k) = |h_i^a(k) - \hat{h}_i(k)|$ for all $k \geq k^*$. In our work, the estimated states are obtained by obtaining a mathematical approximation of the system behavior with a Luenberger observer (we refer to [16,17] for more details on system estimation and the bad-data detection method, which are out of the scope of this paper). When $r_i(k) > \tau_i$ for $\tau_i > 0$, an alarm is triggered indicating the presence of an attack. The main property of this type of detection is that it is based on the physical properties of the system and it can detect changes that violate those physical properties. For instance, Figure 10 depicts how the detection mechanism is able to trigger alarms for $\tau_1 = 0.1$ when an attack induces a sudden change in the sensor measurements, $h_1^a(k^*) = 0.1\ m$, which yields to $r_1(k) > 0.1$. However, an intelligent adversary can remain stealthy by causing small changes in the sensor information.

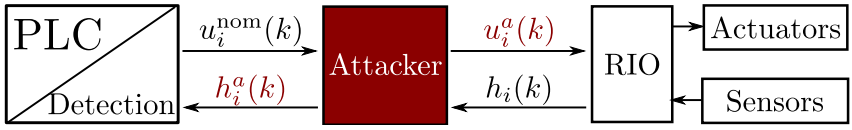

Figure 9. Detection mechanism and man in the middle attack for SWaT. $h_1^{(k)}$ is the attacked water level measure, $u_1^{nom}(k), u_1^a(k)$ are the real and attacked control command, respectively, for time instance k.

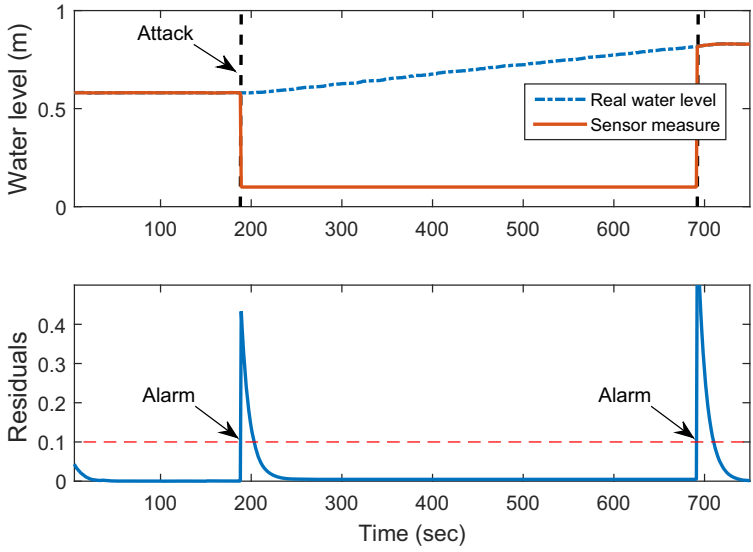

Figure 10. Sensor attack in the water level $h_1^(k)$. The attack is detected when $r_1(k) > 0.1$.

Figure 11 illustrates the effect of the stealthy sensor attack. Before the attack is launched, the valve was closed and the pump was ON, so the water level was decreasing.

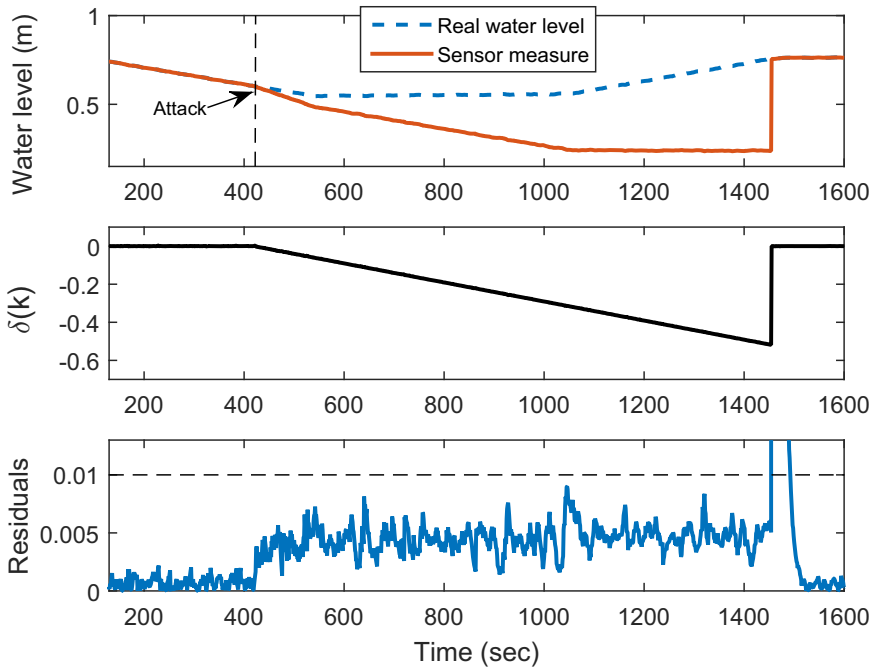

Figure 11. Effects of a stealthy sensor attack in the SWaT testbed. The PLC is connected to an intelligent verifier that analyzes the sensor measure and detects attacks based on sudden changes in the physical behavior. The attack was designed such that the bad-data detector with $\tau_1 = 0.01$ could not detect the attack.

As soon as the attack starts, the sensor measurement received keeps indicating that the water level is always decreasing (a little bit faster because the pump is ON), such that the valve opens when it reaches its minimum level $h_1^a(k) = 0.5\ m$. At that instant the pump continues pumping water such that the real water level remains constant for some time. When the pump stops ($h_1^a(k) < 0.25\ m$), the tank starts filling up with water (the water level $h(k)$ starts increasing) but the false sensor reading keeps indicating low water levels. The valve never closes and eventually it will yield to an overflow. Due to the small rate of change, the bad-data detection with a threshold $\tau_1 = 0.01$ never detects the attack. Smaller thresholds can detect the attack but also increase the number of false alarms. Besides, even if τ_1 is smaller, the attack can be designed to remain stealthy for small Δ.

4. Conclusions

In this work, we discussed practical MitM attacks on ICS Fieldbus communications. In particular, we provided details on the typical topologies (in particular, DLR) and protocols used in such a setting (EtherNet/IP). We also discussed practical challenges in setting up MitM attacks and how to overcome them, and demonstrated results of our practical attacks. We have shown that, although EtherNet/IP can be used as the overall encapsulation protocol, the protocol selection, and therefore, parsing and injection of manipulated data on the encapsulated message depends on the control system layer. SWaT

presents CIP for Supervisory Control Network communications and a device-specific I/O messages for Fieldbus communications.

The MitM of an EtherNet/IP ring also presents several challenges: The attacker must verify that the attacking device incorporated into the ring does not break the ring permanently, as these events could potentially be monitored, i.e. the attacking device must not interfere with the DLR protocol. An attacker who successfully deploys a MitM device into a EtherNet/IP ring established for Fieldbus communications must be assumed to have access to all digital and analog signals. She could inject manipulated data in all sensors and actuators monitored and controlled by the ring's PLC at any time. Therefore, the adversary could effectively isolate and manipulate the physical process disregarding control actions sent by the control room or the PLCs. Using the proposed methodology to launch attacks, different detection mechanisms can be tested and improved. In addition, more robust communication infrastructures can be designed in order to decrease an attacker's impact over the system.

5. Acknowledgments

We thank SWaT's lab engineer Kaung Myat Aung for his valuable advise and help during the experimentation process. In addition, we thank Professor Aditya Mathur for giving us access to the SWaT testbed deployed at SUTD. The work from UT Dallas was supported by NIST under award 70NANB14H236 from the U.S. Department of Commerce.

References

[1] Ettercap Project. https://ettercap.github.io/ettercap/, October 2015.
[2] Python bindings for libnetfilter_queue. https://github.com/fqrouter/python-netfilterqueue, October 2015.
[3] Python Language. Version 2.7.10. https://docs.python.org/2/, October 2015.
[4] Scapy Packet Manupulation Program. Version 2.3.1. http://www.secdev.org/projects/scapy/doc/, October 2015.
[5] Wireshark Network Protocol Analyzer. https://www.wireshark.org/, October 2015.
[6] M. Abrams and J. Weiss. Malicious control system cyber security attack case study–maroochy water services, australia. Technical report, The MITRE Corporation, 2008.
[7] D. Albright, P. Brannan, and C. Walrond. Did stuxnet take out 1,000 centrifuges at the natanz enrichment plant? Technical report, Institute for Science and International Security, 2010.
[8] S. Amin, A. Cárdenas, and S. S. Sastry. Safe and secure networked control systems under denial-of-service attacks. In *Hybrid Systems: Computation and Control. Proc. 12th Intl. Conf. (HSCC '09), LNCS, Vol. 5469, Springer-Verlag*, pages 31–45, 2009.
[9] D. Antonioli and N. O. Tippenhauer. MiniCPS: A toolkit for security research on CPS networks. In *Proceedings of Workshop on Cyber-Physical Systems Security & Privay (CPS-SPC), co-located with CCS*, Oct. 2015.
[10] A. Banerjee, K. Venkatasubramanian, T. Mukherjee, and S. Gupta. Ensuring safety, security, and sustainability of mission-critical cyber-physical systems. *Proceedings of the IEEE*, 100(1):283 –299, Jan 2012.
[11] P. Brooks. EtherNet/IP: Industrial Protocol White Paper. Technical report, Rockwell Automation, 2001.
[12] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry. Attacks against process control systems: risk assessment, detection, and response. In *Proceedings of the 6th ACM symposium on information, computer and communications security*, pages 355–366. ACM, 2011.
[13] I. N. W. Group. Administratively Scoped IP Multicast. http://tools.ietf.org/html/rfc2365, October 2015.
[14] O. Kosut, L. Jia, R. Thomas, and L. Tong. Malicious data attacks on smart grid state estimation: Attack strategies and countermeasures. In *Proc. of the IEEE Conference on Smart Grid Communications (SmartGridComm)*, pages 220–225, Oct 2010.

[15] M. Krotofil and D. Gollmann. Industrial control systems security: What is happening? In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 670–675. IEEE, 2013.

[16] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[17] D. G. Luenberger. Observers for multivariable systems. *Automatic Control, IEEE Transactions on*, 11(2):190–197, 1966.

[18] ODVA. The CIP Networks Library Volume 2: EtherNet/IP Adaptation of CIP, 2007.

[19] L. Xie, Y. Mo, and B. Sinopoli. False data injection attacks in electricity markets. In *Proc. of the IEEE Conference on Smart Grid Communications (SmartGridComm)*, pages 226–231, Oct 2010.