1-1-2016

Leveraging PLC Ladder Logic for Signature Based IDS Rule Generation

Drew Jackson Richey

Leveraging PLC ladder logic for signature based IDS rule generation

By

Drew Jackson Richey

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2016

Leveraging PLC ladder logic for signature based IDS rule generation

By

Drew Jackson Richey

Approved:

Sherif Abdelwahed
(Major Professor)

Thomas H. Morris
(Committee Member)

David A. Dampier
(Committee Member)

James E. Fowler
(Graduate Coordinator)

Jason M. Keith
Dean
Bagley College of Engineering

Name: Drew Jackson Richey

Date of Degree: August 12, 2016

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: Dr. Sherif Abdelwahed

Title of Study: Leveraging PLC ladder logic for signature based IDS rule generation

Pages in Study: 80

Candidate for Degree of Master of Science

Industrial Control Systems (ICS) play a critical part in our world's economy, supply chain and critical infrastructure. Securing the various types of ICS is of the utmost importance and has been a focus of much research for the last several years. At the heart of many defense in depth strategies is the signature based intrusion detection system (IDS). The signatures that define an IDS determine the effectiveness of the system. Existing methods for IDS signature creation do not leverage the information contained within the PLC ladder logic file. The ladder logic file is a rich source of information about the PLC control system. This thesis describes a method for parsing PLC ladder logic to extract address register information, data types and usage that can be used to better define the normal operation of the control system which will allow for rules to be created to detect abnormal activity.

DEDICATION

This thesis is dedicated to my wife, Kristen, and my three boys: Hayes, Dillon and Owen. Without your encouragement and support I would have never reached my goal.

ACKNOWLEDGEMENTS

I would like to thank Dr. Thomas H. Morris for being patient with me and giving of his time to assist me with this project. Thank you to Dr. Sherif Abdelwahed for being willing to serve as my Major Professor and for holding me accountable to my timeline. Thank you to Dr. David Dampier for being willing to serve on my graduate committee.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

ACC	Accumulator
ADD	Addition Instruction
AI	Artificial Intelligence
CI	Computational Intelligence
COTS	Commercial Off the Shelf
CPS	Cyber Physical Systems
CTD	Count Down
CTU	Count Up
DCS	Distributed Control Systems
DHS	Department of Homeland Security
DIV	Division Instruction
DOS	Denial of Service
DMZ	Demilitarized Zone
DPI	Deep Packet Inspection
DTMC	Discrete-Time Markov Chains
EMIDS	Embedded Middleware-level Intrusion Detection System
FBI	Federal Bureau of Investigation
HIDS	Host-based Intrusion Detection System
HMI	Human Machine Interface

ICS	Industrial Control Systems
IDS	Intrusion Detection System
IO	Input/output
IP	Internet Protocol
IT	Information Technology
JIT	Just-In-Time
MAC	Media Access Control
MTU	Master Terminal Unit
MUL	Multiplication Instruction
NIDS	Network-based Intrusion Detection System
OPC	Object Linking and Embedding for Process Control
PLC	Programmable Logic Controller
PRE	Preset
RFID	Radio Frequency Identification
RTO	Retentive Timer On
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Acquisition
SCARA	Selective Compliance Assembly Robot Arm
S-IDS	Sequence-aware Intrusion Detection System
SUB	Subtraction Instruction
TOF	Timer off Delay
TON	Timer on Delay
VM	Virtual Machine

CHAPTER I

INTRODUCTION

1.1 Background

Industrial Control Systems (ICS) arose out of the need to monitor and control processes. The first industrial control systems were comprised of simple mechanical switches, racks of electrical relays that created the logic and a human operator was usually the brains of the operation using a board of pilot light indicators and sight glasses as feedback to make important decisions about the operation of the system. These systems proved to be impractical because they were expensive, difficult to wire and making changes required a lot of process downtime. As technology improved, industrial control systems advanced as well.

ICS is a term that encompasses a variety of different control topologies [1]. ICS can be complex SCADA (supervisory control and data acquisition) or DCS (distributed control systems) systems that are distributed across a large area or they can be simple PLC (programmable logic controller) controlled standalone pieces of equipment that only provide local control such as a conveyor line or robot cell.

SCADA systems often provide control for systems that could be separated geographically by miles, yet typically have a central supervisory station. These systems can often be found controlling and monitoring our nation’s critical infrastructure systems like the electrical power transmission and distribution system, water systems and railway services. SCADA systems are made up of master terminal units (MTU) and remote terminal units (RTU). The RTU, also referred to as the remote telemetry unit, is a field device that provides input and output capability for a specific remote location. The RTUs communicate with the MTU through either a wired or wireless connection. The MTU is the master device that takes input from the RTU to make decisions about what needs to happen in the process. It relays this information to the RTU to allow it to carry out the specified function. Since SCADA systems play such an important role in critical infrastructure systems, much research has been done in an attempt to make existing systems more secure and create new systems that are secure by design.

PLC based systems are typically smaller in size and less distributed than SCADA systems. A PLC might be found controlling a manufacturing line in a factory or controlling a single assembly or test station. PLCs have found their way into SCADA systems as well due to the lower cost and versatility of the equipment. In some instances, they are used as field devices instead of the more application specific RTUs [2].

1.1.1 PLC Based Industrial Control Systems

PLCs evolved out of the need to replace large, complicated racks of electro-mechanical relays that were once used to implement binary logic for hardwired control systems. These relay racks were common in large industrial facilities for years, but were difficult to maintain and debug. As technology advanced, PLCs were designed to replace the logic of contacts and coils. The first PLC was created in the late 1960's and was called the Modicon. This name stood for MOdular DIgital CONtroller [3]. The "contact and coil" terminology carried over to the ladder logic that was used to create the logic to control the PLC. The PLC has evolved a great deal from a simple relay replacement and now can control large, distributed systems using logic that mimics some of the most advanced computer programming languages. PLC control systems, at a minimum, consist of a processor, chassis (rack) and IO cards.

1.1.2 PLC Controllers

The PLC Controller or processor is the brain of the control system. Much the same way that a computer processor controls the functions of a personal computer, the PLC processor turns on outputs by making decisions based on inputs. PLC controllers used to be large, bulky systems like the Allen Bradley PLC 2, but now systems like the Allen Bradley Micrologix systems can offer much more processing power and memory in a smaller physical package [4].

The type of processor to use is usually defined by the functionality required and the budget. PLC manufacturers have learned there is a need for low-cost, simple control solutions as an alternative to the higher priced, more complicated and expandable systems. Sometimes a standalone, single station assembly machine only requires minimal control functionality, whereas a large conveyor line or distributed control system would require a processor that is expandable, has advanced communication options and has the memory and processing power to service hundreds of IO (input/output) points.

More complicated control systems sometimes require more than one processor in a PLC rack. The software can be written in a way to take advantage of the multiple processor architecture. In addition, control systems sometimes use multiple processors that are in separate racks. The processors communicate over the appropriate protocol using what is referred to as messaging.

There are many different types of PLC manufacturers throughout the world and each one has its own hardware architecture, ladder logic design software and nomenclature for addressing input and output registers. For instance, Allen Bradley refers to inputs as “I” and outputs as “O”, but Mitsubishi PLCs use “X” for inputs and “Y” for outputs [5]. The internal registers are just as varied. The type of PLC manufacturer to be used is sometimes determined by the geographic region. Allen Bradley has a large presence in the United States, Siemens has a large presence in Europe and Mitsubishi can be found in Asian countries.

1.1.3 PLC IO

The type of IO (inputs/outputs) available on a given type of PLC is even more varied than the selection of processors. Card options range from simple discrete on/off contact cards to highly specialized cards that communicate using a proprietary protocol.

An input card converts a voltage on the input contact and converts it into an electrical signal that can be processed by the PLC processor. A simple discrete input card could be used to monitor a 24VDC signal from a float switch in a water tank or a mechanical limit switch in a silo. More sophisticated input cards might be used to monitor a 0-10 VDC analog signal that corresponds to the flow rate of a liquid through a pipe or a 4-20 mA analog signal that reports the temperature of a chemical tank. Other types of input cards might receive serial data from a barcode scanner.

An output card converts an electrical signal from the processor to a voltage on the contact of the card. Output cards can provide discrete or analog data. A discrete output might provide 24VDC to turn on the coil of a solenoid that in turn turns on a valve of water used to cool a plant. An analog output might provide a 0-10VDC control signal used to proportionally open a control valve. At 0 VDC the valve might be closed. At 5 VDC the valve might be half-opened and at 10 VDC the valve would be fully open. This proportional control is sometimes required when simple on/off functionality won't suffice. More complicated output cards might be used to send out serial data or communicate with a database over a network connection.

Distributed IO can be incorporated when PLC control is required over a large area. Often distributed IO is used when time, budget or system complexity makes it unreasonable to run multiple hardwired signals to the remote location. Distributed IO can provide remote monitoring and control capability at points at a great distance and communicate to the processor via Ethernet or wireless. The ease of installation and implementation has made distributed/remote IO a popular choice for many control system integrators.

1.1.4 HMI

The HMI (human machine interface) of a control system provides the operator with information regarding the system and provides an interface that the operator can use to interact with the control system. A typical HMI is made up of pushbuttons, indicators and often a graphical representation of the process. An HMI might report information such as temperature values or flow values that have been received by an analog signal and scaled to engineering units that are meaningful to the operator. The operator can use this information to make decisions about the process. An HMI might have pushbuttons that allows the operator to turn valves on and off or it might have a dial that allows the operator to incrementally adjust the speed to a motor.

HMIs typically can be found as two types of hardware: dedicated panels and computers. A dedicated HMI panel is a piece of hardware that was created for a specific task. The panel has a processor and operating system that is designed only to run the HMI program. The memory and communications of the panel are usually customized for the runtime application. When a panel is used, a programmer will design an HMI program using the manufacturer's development software and a compiled program will be loaded onto the panel. When the panel boots up, this program automatically executes. The panel is very limited in functionality aside from runtime program.

Computers can also be used as a control system HMI. Computer HMIs can be created using standard programming languages like Microsoft's Visual Basic .NET [7] or it might use a PLC manufacturer specific development environment like Rockwell Software's FactoryTalk View Studio SE [8]. The advantage of using a manufacturer specific software package is that the communication channel is usually already built into the package. Also, these software packages are usually streamlined for machine monitoring and control. However, it is sometimes advantageous to write a customized program using a standard programming language like .NET. Communication with other computers, databases and interaction over the Internet or often more easily accomplished using customized code. When taking this approach, a third-party communication channel is sometimes used like an OPC server (OLE for Process Control). PLC manufacturers usually provide this capability as well. OPC channels are available through Rockwell Software's RSLinx communication package.

1.1.5 Other Control System Devices

There are many other types of control system devices that might be found on a typical control system network.

- Robots – Robotic arms range in size from small pick and place SCARA (selective compliance assembly robot arm) robots to large robots that can pick up payloads of hundreds of pounds like engine blocks in an automotive assembly plant. Many robot manufacturers provide an interface to communicate to PLCs using their native communication protocol. This interface could be a hardware card that slides in the PLC rack along with the processor or the interface might be a software add-on that is imported into the ladder logic program.

- Drives – Drives are used to control all sizes of motors. Drives are sometimes standalone and don't interact with a PLC network, however often they are tightly integrated with the PLC control system. Many PLC manufacturers provide instruction blocks in the ladder logic that are specific to drive communications. Since position feedback is so critical, drive communications is typically done over fiber optic.

- Meters- There are a wide range of meters that can be found on a PLC network. These meters might be scales that provide weight information, thermometers that provide temperature feedback or specialized test equipment like hipot testers or resistance meters.

- Data Collection Devices – Some PLC systems are highly integrated into the plant data collection system. Many auto manufacturers are required by the government to record and store assembly data for several years after the product has been shipped. Since this data is so critical, the collection of data is often interlocked with the manufacturing process. For instance, a seat might be prevented to advance to the next station on an assembly line until the torque data is collected at the current station. Torque data and other traceability data such as barcode information often travels through the PLC network. Torque controllers, barcode scanners, RFID (radio frequency identification) and barcode printers can often be found connected to a PLC. Much like the other devices mentioned, the manufacturer usually provides a method for these devices to communicate with the PLC.

1.1.6 Ladder Logic

Ladder logic is the programming language used to define the functionality of the PLC. As previously mentioned, ladder logic was created to mimic the relay logic electrical diagrams that were replaced by the PLCs. Ladder logic is made up of rungs that contain contacts, coils and instructions. Typically, inputs are represented as contacts (either normally open, NO, or normally closed, NC). Discrete outputs are represented as coils. More sophisticated functions can be achieved using instruction blocks. A ladder logic program is sequential in nature. An output or instruction at the end of a rung will not be executed until the NO and NC contacts in front of it are all closed.

Each PLC manufacturer has its own ladder logic syntax and programming software. Some standardization has been achieved through the IEC 61131-3 standard, however rarely can a ladder logic program be used across two PLC manufacturers. In fact, it is often difficult to use the same ladder program when switching between PLC models of the same manufacturer.

1.1.7 ICS Protocols

Communication between the PLC, HMI and other devices is accomplished through a predetermined protocol that can be sent and received by all devices. A protocol is the defined system for communication that include their own rules, formatting and timing. Protocols are sometimes dependent on the type of physical medium on which they are transmitting. There are protocols specific to wireless, serial, and Ethernet among others. Protocols are also highly manufacturer dependent.

Some protocols commonly used for industrial control systems include DNP3, PROFIBUS, Modbus and Ethernet/IP. These protocols are not that different than ones found in the IT environment because they just define a way for data to be organized in a packet, however different levels of importance are placed on the reliability, confidentiality, integrity and availability of these protocols when comparing between the control system and IT worlds.

1.2 Motivation

Industrial control systems were once an island unto themselves, isolated from the outside world. In recent years, corporate managers have realized the value of the data that resides in the control systems on the manufacturing floor. Production totals, down-time and alerts are now piped over the network into enterprise systems, into the office environment and out to the Internet. It is not uncommon to find HMIs or computers used for troubleshooting PLC networks connected to both the ICS and the corporate plant network so engineers in the office or at home can view and troubleshoot issues with the control system. As these control systems become more open and connected, they also become more vulnerable.

1.2.1 Problems with ICS Security

Even though great strides are being made to improve upon the security of new control system architectures, there are many PLCs in the U.S. manufacturing sector that are several hardware/firmware revisions old and the likelihood of them being upgraded is low. Typically, the “if it’s not broke, don’t fix it” mentality applies to many manufacturing sectors. There are numerous reasons that lead to this line of thinking, but the fear of causing additional downtime and the lack of financial resources are two reasons that typically rank high. This lack of maintenance results in a large number of vulnerable PLCs that rely completely on the perimeter security of the network.

The task of securing a factory's network perimeter is usually the responsibility of the plant IT (information technology) department. Most IT professionals have a good understanding of what it takes to secure an office network, but they typically lack the control system background necessary to properly secure the manufacturing side of the network. There are many differences between office networks and control system networks that require a different security approach to be applied. One key requirement for a control system network is low latency. In order to control motors and robots in real-time, throughput is critical and the communication lag must be minimal. Proper networking architectures and routing must be employed to ensure network efficiency. Oftentimes, IT personnel make the mistake of not properly isolating the office network traffic from the manufacturing network traffic. This not only introduces additional traffic onto the time sensitive network, but it also increases the risk of unauthorized access to the control system network. It is typically not acceptable to completely isolate the manufacturing network from the outside world because many engineers and managers want the option of remote access to make changes to the system if a problem arises or to have automated alerts of production numbers and system errors [8]. The manufacturing network must have access to the outside world to provide these capabilities.

Another key requirement for a control system network is ease of access. At first glance, this appears to be synonymous with a lack of security, but if handled correctly, access can be granted in both a simple and secure manner. The number one rule in production environments is to keep the product flowing. In order to do this, maintenance must be able to quickly resolve errors as they arise. Maintenance personnel must be able to promptly go on-line with a PLC processor in order to address errors at any hour of the day. Production downtime would increase if complex security measures were put in place that prevented this from happening. Even if complex security measures were put in place initially with good intentions, history has shown that these measures would soon be bypassed after an engineer gets interrupted at home a few times. In an automation environment, simple works best.

1.2.2 Risks of Poor ICS Security

The severity of a control system attack is realized when the financial and safety implications are taken into consideration. An intruder making changes to data tables in the PLC could result in actuators moving when an operator is in an unsafe position or could cause equipment damage. Set-point values can be changed to produce product that must be scrapped because it’s out of tolerance. Any loss time in a production environment takes a financial toll on the organization. Extended production loss can have a cascading effect on the supply chain for the goods being produced. Many companies adhere to a Just-In-Time (JIT) manufacturing philosophy that is a sharp contrast to the warehousing methodology. With JIT, there is no buffer of warehoused components that can be drawn from if the factory production halts. Components are produced and are utilized in the finished product within hours in an effort to be more efficient and save money. Many foreign automakers use this production strategy to gain a financial edge on the competition. For instance, a set of seats rolling off the production line for a vehicle is only a truck ride away from being installed into a vehicle just a few hours later. JIT requires much coordination between scheduling, inventory tracking and production [9]. If any of these pieces of the system fail, then production stops. In the previous example, if seats are not delivered to the automaker, then no vehicles can be produced. This cascading effect results in thousands of people standing around with nothing to do and financial losses on the order of millions of dollars.

In some instances, an attack on a control system might cripple a custom piece of equipment that takes weeks to replace. A more tragic consequence of an attack would be if a human casualty occurred. A safety violation of this magnitude would result in production being shut down until a lengthy investigation could occur. In either instance, the financial ramifications would be extensive.

If the network perimeter security is breeched, the only obstacle standing between an attacker and a disruption of this magnitude are the security mechanisms inherent to the control system. The lack of authentication and encryption in many of today's control system protocols make them an easy target for a malicious attack.

1.2.3 Types of ICS Attacks

Attacks against industrial control systems can be categorized into four major categories: reconnaissance, response and measurement injections, command injection and denial of service [10]. Each of these four categories of attacks require a different level of knowledge about the system under attack and each one have varying degrees of impact on the industrial control system.

Reconnaissance attacks are typically the first step taken when attempting to circumvent the security of an ICS. Reconnaissance is the gathering of information to determine the network architecture in order to learn information about devices on the network such as the IP or MAC address of a PLC to target. This can be done using downloadable software tools such as Nmap and Wireshark to analyze the network. Port scanning is useful in determining what communication ports are available on a network

[11]. Nmap can be used to not only identify open ports and IP addresses, but even some known network vulnerabilities. Wireshark can sniff packets from the network and the packets can be analyzed to find a target of interest. Identifying a PLC is typically not that difficult if some basic vendor information is known. Oftentimes, vendors will have certain ports that they prefer to use and this information can be found in the knowledge base on the vendor's website. For instance, if port 2222 is commonly used for PLC communications [13], then Nmap or Wireshark can be used to identify the presence of this port on the network. Once a potential PLC is identified, packets can continue to be intercepted and analyzed. An attacker might choose to collect information coming to and from the PLC over an extended period of time in an effort to profile the system. Profiling the system allows an attacker to identify which PLC registers are frequently written or read. The values for these registers can also be analyzed for patterns. These patterns can help the attacker identify what type of process the PLC is controlling. Over a period of time, the attacker could become familiar enough with a system to identify critical values and the acceptable limits for those values. This is valuable information that will aid in planning an inconspicuous attack that will disrupt production.

The second category of attacks, response and measurement injection, takes advantage of one of the basic characteristics of ICS. A typical PLC operates by scanning through its ladder logic program, top to bottom, left to right. The amount of time it takes to completely cycle through the ladder logic is called its scan time. The scan time of a program is typically in the milliseconds range. Part of the scan time includes the polling of all local and remote IO. The remote IO can consist of devices such as remote sensors, other controllers on the network or possibly HMI terminals. Devices can either provide information to the PLC when the PLC requests the information, on a specific interval or when a value state change occurs. These devices are polled on an interval, related to scan time, by sending a packet to the remote device requesting current state information. Once the remote device receives the packet, it responds to provide the PLC with information. Devices that are not polled may send information asynchronously when a state change occurs at the device. For instance, a temperature transducer might only send a packet to the PLC when the temperature reading changes from one value to another.

Since many ICS network protocols don't have authentication built in, it is almost impossible for a PLC to validate the source of the remote IO information. Once reconnaissance has been done to identify the typical flow and format of response packets from remote devices on the network, it is not difficult to craft or modify response packets that contain erroneous information. Injecting data can result in critical data being modified or possibly causing the PLC to enter into an error state. A more complicated attack might involve providing the plant operator with bad information via the operator interface which might, in turn, lead him/her to make an uninformed decision that could cause harm to the system. The lack of authentication that is inherent in most control system protocols and the fact that most ICS protocols drop duplicate response packets make them very susceptible to response and measurement injection attacks.

A third category of attack is command injection. This is very similar to the response injection attacks described above however instead of the packets from remote devices to the PLC being altered, with command injection the packets from the PLC to the remoted devices are altered. Again, this is possible in ICS networks because of the lack of authentication that was described above. Through command injection attacks, a hacker has the ability to send erroneous information to a physical device that could cause an actuator to move when it is not supposed to or could cause a variable speed device such as a motor to move at a faster rate than which it was designed. This could pose a danger to the operator, the equipment or the process itself resulting in downtime, financial losses, casualties or fatalities.

Command injection could also alter certain characteristics of a control system such as control set points or recipe files. In some PLCs it is also possible to reconfigure devices or possibly modify the actual PLC ladder logic that controls the system.

The fourth category of attacks, Denial of Service (DOS), attempt to interrupt the normal operation of an industrial control system. DOS attacks can be physical attacks on some portion of the system such as cutting communication wires, forcing a valve to stay in a certain position or physically destroying a device. DOS attacks can also target the communication channels within an ICS. The simplicity of this attack is that it doesn't require taking down the entire network, rather blocking communication with one critical device on the network could result in adverse conditions. This could be accomplished by flooding a device with packets requesting information at a rate faster than the device can respond. The device would be so busy trying to service the false incoming packets that it would be unable to service the legitimate ICS packets. Sometimes flooding a device with packets will overwhelm the device and cause it to go offline until power is cycled. This type of attack requires very little information about the control system itself, therefore denial of service attacks against ICS are very common.

1.2.4 Sources of ICS Attacks

A malicious attack on a control system could be initiated by a variety of people, however three groups should be looked at more closely: Remote hackers, contractors and operators. Each group has very different motives, knowledge of the system and resources to harm the system.

The group that typically comes to mind when computer security is discussed is the outside entity commonly referred to as a hacker. These individuals are driven by many reasons that can range from sport to terrorism. The motivation of the attacker also plays a part in the level of effort invested in information gathering and the goal of the attack. A random hacker is at a disadvantage because their knowledge of the system is typically limited. The attacker might be hundreds of miles away from the target and possibly have no prior information as to the type of facility or manufacturing processes. However, a skilled hacker is more resourceful in the area of attack tools and techniques and typically has previous experience that can assist in compromising security. It is important to remember that a control system is typically protected by only its perimeter. The perimeter security is made up of components such as firewalls that are present in most any type of network. That being the case, it is not uncommon for an experienced hacker to possess the capabilities to breech an inappropriately secured perimeter. Once the outer ring of protection is circumvented, the attacker has free reign over the unprotected control system.

The next group of potential attacker is the contractor. It is common for most manufacturing facilities to have contractors in and out on a daily basis. The larger the organization, the more contractors come and go. Contractors not only have physical access to many critical areas of the plant, but they are sometimes granted remote access as well to support the plant systems. In many cases, contractors have a good understanding of how systems in the plants work and the data that is passed back and forth across the networks. They also are knowledgeable of which areas of the plant are most vulnerable and could cause the lengthiest downtime. Contractors, specifically programmers and machine builders, have the computer skills and equipment necessary to exploit a control system. The contractor's ability to utilize additional tools such as network sniffers is only limited by the amount of effort they choose to put into an attack. A contractor could be motivated to attack a system for a variety of reasons like sport and revenge, like in the Maroochy Water Services incident [14].

The last group of attacker is the plant operator. This group can be motivated to cause harm for several different reasons: disgruntled, want time off, or accidental by bringing in a virus from home. The operator certainly has physical access to the operation and has an extreme understanding of the process. The tools available to the operator would be limited without them drawing attention to themselves. Table 1.1 shows a comparison of these three types of potential sources of attack.

Table 1.1 Three Potential Sources of ICS Attack

Type	**Motivation**	**System Knowledge**	**Skill Level**
Outside Hacker	Sport or Terrorism	Minimal	High
Contractor	Sport, Revenge	Medium to High	Medium
Operator	Disgruntled	High	Low

1.2.5 Attacks on ICS

PLC security received international attention in the summer of 2010 with the discovery of Stuxnet. Stuxnet was a very sophisticated, highly engineered worm with a very small foot print that successfully attacked an Iranian air-gapped ICS responsible for enriching uranium. The worm targeted Microsoft Windows machines running Siemens software and replicated itself across networks and USB drives. Its apparent intended target was the Iranian nuclear program. The worm not only allowed the authors to spy on the industrial control systems it infected, but it also had the potential to adjust the speed of centrifuges in an attempt to cause physical damage and disrupt the nuclear program.

Through elaborate reconnaissance and social engineering, the worm's authors were able to determine that Iran's nuclear program's control systems used the Siemens Simatic S7 PLC and Siemens WinCC HMI software and orchestrate its delivery to the nuclear facility on a USB drive. This allowed the worm to specifically target those systems by taking advantage of a hard-coded database password in the system as well as replace a key communication DLL that intercepted traffic between the WinCC HMI and the Siemens S7 PLC.

Stuxnet was unique because it was the first piece of malicious code that intentionally targeted an industrial control system. Inspection of the code revealed that the authors knew a great deal about control systems and had very specific information regarding the Iranian nuclear program [15].

In 2011 another piece of malicious code that targeted control systems was discovered called Duqu. Duqu was officially labeled a remote access Trojan (RAT) and appeared to share some of the same code as Stuxnet. However, unlike Stuxnet, Duqu was not self-replicating and there didn’t appear to be any ICS-specific attack code contained in the executable. Duqu’s primary function was to steal information regarding the infected system. This information stealing might have been a reconnaissance effort to develop a Stuxnet like attack in the future that targets the system specifically. Once a system is infected, the attacker can control other systems through a peer-to-peer command and control (C&C) protocol [15].

In 2012 another variant of Stuxnet surfaced called Flame. Researchers discovered that Stuxnet and Flame shared identical segments of code. Flame was unique because it was considered large for malware. Much like Duqu, Flame’s major purpose was information collection and not destruction. Flame utilized a very effective and sophisticated spreading technique by using Windows Update [16].

Another remote access Trojan that targets ICS is Havex. Much like Duqu, Havex’s primary function is to steal information regarding the infected system. Once the information is collected it communicates with a C&C server. Havex specifically targets systems that use the older DCOM-based (Distributed Component Object Model) version of the Open Platform Communications (OPC) communication protocol. A newer OPC standard was released in 2006 and Havex does not appear to target the newer standard [17] [18].

OPC is often used to establish communications between two devices that don’t share the same native protocol. For instance, some PCs require the use of OPC to communicate with a PLC or other device. An OPC server resides on the PC and each tag specified in the OPC server defines a communication path to a specific address register in the PLC. Havex can profile a system by collecting the tag information which can provide information such as address registers and their possible uses and addresses for connected devices

One of the most recent examples of an attack against a critical infrastructure ICS is the December 23, 2015 hack of Ukraine's power grid. This was a well-crafted plan that ultimately left more than 225,000 residents in the dark. Multiple agencies in the US, including the DHS and FBI, helped Ukrainians investigate the attack. Fortunately, the Ukrainian power distribution companies had multiple system logs in place that helped reverse engineer the attack. The investigation has shown that this was a well-planned attack that started reconnaissance operations many months ahead of time. The initial attempts to gain information about the plant was done through macro-enabled Microsoft Word documents sent via e-mail to various plant employees. This phishing campaign installed a program called BlackEnergy3 onto their systems and gave the hackers backdoor access to the system. Months of reconnaissance allowed the hackers to map out the network and gather passwords; all key to the subsequent attack.

The actual attack consisted of several steps. The operators were locked out of their SCADA terminals to prevent them from interfering with the attack. The hackers then opened up various breakers that cut power to thousands of households. To delay their presence being discovered, the hackers initiated a denial of service attack on the telephone system to prevent customers from calling in to alert the power company that something was wrong. The hackers then replaced the firmware on several serial-to-Ethernet converters to prevent corrective commands from being sent to reclose the breakers. After they had successfully interrupted power distribution, then the hackers used malware called KillDisk to overwrite the master boot record and erase files from the operator workstations to delay recovery. The attackers were also able to disable the emergency backup generators at the power plant which left the plant itself in the dark. In the end, over thirty Ukrainian substations were offline for about three hours [19] [20].

1.3 Contribution

The attacks mentioned in the previous section make it clear that threats against ICS are increasing. The insecure by design components that make up industrial control systems are vulnerable and, until wholesale design changes are implemented, require other security mechanisms to be in place in order to secure our manufacturing facilities and critical infrastructure installations. There is much research in the area of ICS security and great steps are being made to improve our ICS systems. New methods are needed to take information that is currently available and utilize this information to provide better security for industrial control systems.

This thesis includes three major contributions. First, a method for leveraging the information contained in PLC ladder logic code to create signatures for a network intrusion detection system (NIDS) is explored. There are two major categories of signatures that can be derived from analyzing ladder logic code: address register usage and invalid register values. Each of these two categories can be broken down into distinct classes of rules. Second, a Microsoft Visual Basic .NET ladder logic parser program was developed to automatically extract information from an Allen Bradley SLC 500 ladder logic program and create Snort IDS rules [21]. These rules were tested and the parser program's functionality was proven accurate and reliable. Although this software was written to target a specific PLC manufacturer and model, it could easily be modified to accommodate other ladder logic files that are similar in design. Third, the research that led to the development of the ladder logic parser program gives insight into the packet format of the Allen Bradley Ethernet protocol as well as identifies some of the vulnerabilities that still exist in some older, yet highly deployed, models of PLCs.

1.4 Organization

The remainder of this thesis is organized as follows. Chapter II provides a literature review. The literature review is broken into a survey of current ICS security methods, anomaly-based intrusion detection systems, signature-based intrusion detection systems and industrial control system specific intrusion detection system techniques. Chapter III discusses the process for creating Snort IDS rules leveraging the information contained in PLC ladder logic. A brief explanation of address register usage and invalid register values such as division by zero, math overflow, negative values and invalid addressing is covered. After this overview of what can be extracted from PLC ladder logic, then the hardware and software used for testing are discussed. The method for extracting information from the ladder logic and using that information to create IDS rules, along with examples, are discussed in detail. Chapter IV provides an explanation of the testing, evaluation and validation process. Chapter V offers general conclusions and opportunities for future work.

CHAPTER II

LITERATURE REVIEW

As discussed in Chapter I, industrial control systems are the backbone of many of critical infrastructure processes and play an important role in producing the goods and services that stabilize our economy. The ICS vulnerabilities that were discussed along with the ever increasing threats of worms and viruses such as Stuxnet and DuQu have increased awareness toward securing industrial control systems. This chapter takes a look at the current methods that exist for ICS security and a survey of the current state of intrusion detection systems for ICS and current research that is taking place.

2.1 Survey of Current ICS Security Methods

Industrial Control Systems once resided as isolated systems that had no connectivity outside of the local process. With the expanse of the Internet and the wide acceptance of TCP/IP based devices, most ICS hardware vendors have started to abandon some of the early proprietary protocols and communication mediums to transition to more open protocols over Ethernet that allow for interconnectivity between various types of control hardware. This evolution has resulted in ICS systems that are becoming increasing more like traditional information technology (IT) networks. This transition has allowed these once isolated ICS systems to share information with managers in the front office or to the world.

The more ICS become like IT networks, the more applicable standard IT security solutions become. Although the characteristics of ICS and IT are becoming more similar, great care needs to be taken when securing ICS because of what they control. Traditional IT security focuses on protecting data. ICS security ultimately protects lives, financial losses and harm to our environment.

The three basic tenants of information security – confidentiality, integrity, and availability - apply to ICS security; however, they apply in a different order than in the IT realm. These well-known tenants are defined below:

- Confidentiality – guarantee that information is not disclosed to unauthorized entities
- Integrity – guarantee that the information is complete, valid and from a trusted source
- Availability – guarantee that the information is available when needed in a timely manner

In the Information Technology world, confidentiality ranks highest, followed by integrity and availability [23]. In the ICS world, the order of importance reverses as shown in the Table 2.1.

Table 2.1 Order of Importance: Confidentiality, Integrity, & Availability

Order of Importance	**IT**	**ICS**
↓	Confidentiality	Availability
	Integrity	Integrity
	Availability	Confidentiality

The once isolated ICS can no longer be protected with physical security like locked doors and fences. The demands for information flow and external access has resulted in the typical insecure-by-design hardware components of the ICS to be accessible to the outside world. Insiders are no longer the only threat to protect against. Now industrial control systems must protect against the disgruntled insider, but also a plethora of external threats that seek to infiltrate the system [23]. This has resulted in many different approaches to protecting an ICS network. The following sections will give a brief overview of the current ICS protection techniques.

2.1.1 Network Architecture

Network design considerations are integral for providing a secure network for industrial control systems. A common practice is to separate the ICS network from the office network. The office network has a constant flow of information related to internet browsing, email and other in-house communications that take bandwidth away from the low latency dependent ICS network. The management of network hardware is another thing to consider. It is not uncommon for a corporate IT manager to push out updates or other network maintenance routines that could negatively impact ICS devices. Separating the office network from the ICS network provides a clear line of responsibility between the IT department and the control engineering department. This line of responsibility will allow the IT administrators to focus on securing and maintaining the office networks while the control systems engineer or industrial security administrator can protect the ICS network while not affecting the performance of the control system.

As previously discussed, it is not always practical to have the ICS network totally isolated from the IT network. The connection of these two systems must take into consideration a number of things such as how many physical connections will be provided, who or what systems will have access to the ICS, what path does the ICS have to the outside world, what ports will be open, etc. This isolation and access control can be achieved through network architecture design using boundary protection devices such as: gateways, routers, data diodes and firewalls among others.

2.1.1.1 Firewalls and Network Segregation

Firewalls are devices, systems or software that control the flow of packets between different networks. Firewalls are embedded into most internet routers found in homes around the world. Their complexity can range from simple port blocking firewalls found in home internet routers to very expensive systems that give very granular control over the network traffic entering or existing a network. Traditionally these devices can be found separating a network from the Internet, however they are increasingly used more and more to separate sub networks within an organization. Firewalls provide the ability to compartmentalize information within an organization to prevent unauthorized access to sensitive information.

Firewalls can be found in many different forms, however there are two categories that should be mentioned: Packet Filtering Firewalls and Stateful Inspection Firewalls. The most elementary function of a firewall is to filter packets. Packet Filtering Firewalls are routing devices that determine the flow of network packets based on the content and type of the packet. The firewall configuration software allows a user to establish rules about what traffic can enter and exit the network. These rules are based on packet data such as source IP address, destination IP address, port numbers, protocol type and other criteria that might be specific to the type of firewall. The firewall will be configured to allow the specific packet to continue on or block the packet depending on the rule set. These devices are typically low cost solutions that have a very minimal impact on the performance of the network.

The second type of firewall is the Stateful Inspection Firewall. These have the same functionality as the Packet Filtering Firewalls, however they have an extra level of awareness due to the inspection of the Transport Layer (level 4) data. This allows the firewall to keep a record of active sessions and uses that information in determining if a packet should be blocked or forwarded. These types of firewalls are more complex and require network expertise and a more in-depth knowledge of the system to configure.

Firewalls are one of the most utilized security features to help secure an ICS. The boundary protection it provides not only restricts unintended access from the office network, it also prevents unnecessary traffic from taking bandwidth away from the critical operations of the ICS. This separation allows the two separate systems to maintain independent security policies while maintaining a connection that can only be bridged by authorized systems that meet the configured criteria. An added benefit of a firewall is that it can log information flow that can be used for analysis in the event of an intrusion.

Most firewalls are designed with the IT network environment in mind. That being the case, there are a few drawbacks to deploying a firewall on an industrial network. Many ICS protocols have yet to be implemented in commercial off the shelf (COTS) firewalls. ICS vendors have operated with proprietary protocols for years and the variety and complexity of these protocols have prevented them from being implemented by firewall manufacturers. A second concern, as with any packet inspection operation, there could be delays introduced in network traffic by the firewall. This could impede availability and have a negative effect on the overall performance of the ICS.

Deep Packet Inspection (DPI) firewalls are now very common for many IT protocols, unfortunately this technology is very limited for the ICS protocols. DPI for industrial protocols would allow filters to be applied to certain fields within a protocol packet such as commands (write vs read), objects (motor, actuator, etc.), services (get vs. set) and PLC address register ranges. Work is being done in the area of EtherNet/IP and Modbus to implement DPI [24] [25].

2.1.1.2 Network DMZ

A demilitarized zone (DMZ) is a network that buffers access between the corporate or outside network and the ICS network. The DMZ sits between the two networks with both the corporate network and the ICS network only being able to access the shared DMZ, but not each other directly. A DMZ can be created using various router and firewall network configurations. For systems requiring information sharing between an ICS and a corporate network, a DMZ should be implemented. The ICS data would be written to a historian or data collection server within the DMZ and the corporate users would access this information from the DMZ instead of directly from the office network. External access via the Internet could take place through the corporate network for added security or directly from the DMZ with a firewall. A single DMZ zone is rarely sufficient. In most circumstances, it is recommended to use multiple DMZs. Utilizing routers and firewall devices from different manufacturers is another recommended when segregating networks.

2.2 Intrusion Detection Systems

An intrusion detection system (IDS) is software or a hardware device that monitors a system for malicious activity and produces a report or responds to the threat by dropping the suspect packet. There are two major platforms for IDS: Host Intrusion Detection Systems (HIDS) and Network Intrusion Detection System (NIDS). A HIDS runs on a device (host) and monitors the state of that host to determine if malicious activity is taking place. The HIDS can monitor the state of the file system or the network traffic entering or exiting the system. One benefit of using a HIDS is the ease of specifying what is and is not acceptable behavior for the host. Since the behavior is specified at the host level, then it can be uniquely tailored for that host only. A major disadvantage of using a HIDS is the resources it consumes that could have otherwise been used by the host application. Oftentimes, ICS devices have limited resources such as memory and processor power and cannot afford to share those precious resources with a secondary process.

A NIDS monitors the traffic on a network and can analyze packet information to determine if malicious activity is occurring against any device on the network. The HIDS gives protection for one device only whereas the NIDS can protect multiple devices on an ICS. A NIDS could simply look at the frequency of packets in a network or it could perform protocol specific deep-packet inspection. Inspecting the payload of an industrial protocol requires an in-depth understanding of the protocol which can be a research topic in itself. A major advantage of using a NIDS is that the individual nodes (hosts) are not taxed with inspecting logs and traffic for malicious activity. Placement of the NIDS within a network is key and this is often a difficult task due to network architecture. For maximum effectiveness, the NIDS would need to be placed so all network traffic is visible to the IDS.

The architecture and design characteristics of ICS lend itself to be a good candidate of IDS protection [26]. [28] identifies four major differences between industrial control system IDS and traditional I.T. system IDS:

- Physical Process Monitoring – unlike regular IT networks, an ICS has data being exchanged that represents physical characteristics of the processes that it is controlling. Therefore, the laws of physics apply which provides definable behaviors for certain data sets. For instance, a water storage tank has certain level bounds that it could never exceed and a certain level that it could never be below. A chemical flowing through a pipe has certain flow values that would be physically impossible to reach. These physical restrictions help better define what is acceptable ICS behavior.

- Closed Control Loops – ICS networks lend themselves to IDS protection because the network topology is typically static and the routine polling of IO devices is somewhat predictable. PLC programs are very cyclic by nature and this time-based execution makes it easier to define what is normal and what is abnormal. This differs greatly than network that may be seen on a typical IT network where traffic is dependent on what the users on the network might be doing. A person checking their e-mail or surfing the web can produce unpredictable and very intermittent network traffic that is more difficult to profile.

- Attack Sophistication – Since ICS typically control Cyber Physical Systems (CPS), the ramifications of an attack are quite high. An attack on an ICS could leave millions without water or power. It could result in irreparable damage to physical systems that could shut down entire processes and have extreme economic consequences. Circumventing security of an ICS could damage the environment or even result in human casualties or fatalities. The payoff of a successful security breach is high, therefore the sophistication and amount of effort is great. Infiltration of an IT network comes with its own set of motivating factors: corporate espionage, ransom of data or deletion of data; however, the sophistication of attacks against ICS have proven to be impressive and will only continue to increase with the rise of state sponsored cyber warfare.

- Legacy Technology – In the IT world, hardware upgrades come around almost every year and the system admins are pushing out software updates and policy changes on a weekly basis. This is not the case in the static world of ICS. Control system hardware is very expensive to replace and usually requires downtime. Downtime means no profits. This legacy hardware makes it a good target for hackers that are seeking to infiltrate a system that has known vulnerabilities and no firmware updates or patches applied.

There are several metrics that are commonly used to characterize IDS performance. The time it takes for and IDS to alert following malicious activity, packet sampling efficiency and the hardware resource requirements all factor into an IDS system's success rate. IDS systems traditionally use the following three terms when defining the accuracy of a system to detect malicious activity:

- False Positive – A false positive occurs when an IDS identifies malicious behavior when none is present.
- False Negative – A false negative occurs when an IDS fails to identify malicious behavior when malicious behavior is present.
- True Positive – A true positive occurs when an IDS successfully identifies that malicious activity is occurring.

A perfect system would have 100% true positives every time, however this is not a practical expectation. The balance between false positives and false negatives is a delicate one. If a system has too many false negatives, then the system would consistently fail to alert on malicious activity. If a system has too many false positives, then a system administrator might become numb to the alerts because of "The Boy That Cried Wolf" effect. Many security administrators would prefer the latter simply because it makes for a more secure system. This would require a secondary process for evaluating each security alert to determine whether or not it is valid. This might be done via human intervention or some secondary automated security alert log file inspection routine [28].

2.2.1 Anomaly-based IDS

Anomaly-based IDS systems, also known as behavior-based systems, flag activity that appears abnormal by comparing the activity to an established baseline. When the deviation between the observed activity and the established baseline exceeds a predefined threshold, then an alarm occurs. An anomaly-based IDS has the potential to detect malicious activity that has never been observed before and studied. Due to the nature of anomaly-based systems, they tend to result in a higher occurrences of false positives than other IDS techniques. They can also become very complex which can be taxing on the performance of a system and the setup time is also often lengthy. Another disadvantage of some of the anomaly-based IDS systems is that an intruder can train the system over time to convince the system that the malicious activity is normal.

There are many methods used to define the baseline for “normal” behavior in an anomaly-based system. Statistical based approaches profile normal behavior using the frequency of events over time, standard deviation, statistical mean and other mathematical correlations. Statistical techniques [29] [30] [31] [32] [33] include: Markov process/Marker Model, Operational, Multivariate, Statistical Moments, Times Series, Univariate, among others. The statistical based approach does not require prior knowledge about normal activity.

Cognition based approaches, also known as knowledge-based, analyze audit data, but are also influenced by a set of predefined rules and input from a training dataset. Knowledge-based approaches include: finite state machines, description scripts and expert systems. This approach can be very flexible, robust and scalable, however defining valid data can be time-consuming [34].

Machine learning involves both supervised and unsupervised learning. A machine learning algorithm creates models by monitoring the system over time to create patterns for behavior. Machine learning approaches include: Baysian networks [35], generic algorithms, neural networks, fuzzy logic and outlier detection [34]. Zamani and Movahedi [36] break down machine learning into two categories: Artificial Intelligence (AI) and Computational Intelligence (CI). AI techniques refer to the statistical modeling approaches and CI methods are the biological-inspired models that address problems that statistical approaches cannot. CI models include fuzzy logic, artificial neural networks, evolutionary computation and artificial immune systems.

2.2.2 Signature-based IDS

Signature-based IDS systems, also known as knowledge-based, analyze information and compare that information to predefined signatures that are indicative of malicious activity. A signature is a pattern that identifies a known attack. After a known attack is identified, the pattern of that attack can be studied and payload patterns that help identify that attack are noted and a signature is created. Because of this process, signature-based attacks have a very low occurrence of false positives because, by definition, only patterns that match a known attack signature will create an alert.

Unfortunately, the same characteristic that makes this IDS attractive is also a major disadvantage to this technique. Since the attack pattern behavior is for known attacks, the system can be circumvented fairly easily by the slightest attack modification. Because of this, the signatures must continually be updated just like virus protection software must continually update the virus definitions. Great effort must be put into writing signatures that match the pattern for malicious behavior, but also allow for variance in the known attack.

The signatures of attack are defined through a number of different methods. Log files from known attacks can be analyzed to develop a characteristic signature for that attack. Known vulnerabilities in a system can be studied and signatures can be created to thwart an intrusion. An in-depth understanding of various protocols within a network must be gained in order to develop meaningful signatures. This is often a very expensive and time intensive process.

Because signature-based approaches are less susceptible to generating false alarms, this approach is very attractive and much research had been put into countering the negatives of this technique by finding ways to minimize its vulnerability to variance of known attacks and to reduce the amount of processing power and time it takes to pattern match against known signatures. In [37], Uddin et al. propose a multi-layer signature database model that uses mobile agents to transfer signatures from a large complementary database to smaller signature databases at the point of inspection. The key to this method is that there are multiple IDS inspection engines analyzing the code so the workload is distributed. This proposed model is an attempt to improve the time and computational requirements for signature pattern matching. The results of this method were promising. The researchers were able to reduce the size of the signature database at each inspection node which resulted in a significant decrease in the number of dropped packets.

In [38], the author questions the universal claim that signature based IDS cannot detect zero-day attacks. This theory is tested using Snort IDS and presenting it with 183 zero-day attacks and 173 known attacks. This is done by using a current Metasploit Framework and a Snort rule set that was older than the vulnerabilities to be tested. After testing, it was noted that the zero-day attacks had a mean detection rate of 17% compared to 54% with the known attacks. After false positive reduction is taken into consideration the result of the paper indicates that a Snort signature based IDS has a zero-day detection rate of around 8.2%.

The novel approach of this thesis involves the automated creation of IDS signatures. Many times, signatures for possible attacks are created through painstaking analysis of network logs. [39] takes a look at automated signature creation for signature based IDS with an Automated Signature Creator called Pancakes. This proposed method uses a virtual machine mirror system where network traffic is logged by both the Snort system and the anomaly-based IDS on the virtual machine system. Since anomaly-based IDSs are often subject to false positives, a supervised verification of flagged traffic is necessary. Using machine learning, the authors can determine which machine learning algorithms are best for detecting malicious activity. The researches discuss the following machine learning algorithms that were studied: Naïve-Bayes Classification Algorithm [40], C4.5 Algorithm [42], Random Tree [43], Random Forrest [43] and Logistic Regression [44]. They found that the Random Tree Algorithm correctly classified malicious activity best. Once the malicious traffic is identified, signatures can be created through an automated process.

2.3 Defense-in-Depth

The defense mechanisms discussed in the previous sections are only individual bricks in the wall of defense-in-depth architecture. IT systems and ICS systems alike can be very large and complicated systems. There are multiple points that could be targeted by an attacker. Guarding the entry points with firewalls might aid in protection against an outside attack, however it would do nothing for the disgruntled insider or a well-intentioned employee that brings in malicious code on a USB drive or opens a properly placed e-mail. For example, in 2010 the Mariposa virus was brought into a utility company on a laptop by an employee that was unaware that his laptop was infected. The employee's company did not detect the virus on their system and it was not detected until the second utility company reported malicious activity. The virus was tracked back to a USB drive that was shared among participants at an industrial conference. Before it was detected, it had already infected multiple systems [45]. In order to effectively protect an ICS, security must be enforced at multiple levels using a variety of the security prevention and detection techniques discussed in previous sections. This multi-layered defense technique is referred to as defense-in-depth architecture and requires a holistic approach that utilizes all defense resources in an attempt to deter all malicious activity.

In order to establish a defense-in-depth architecture, an organization must first know the threats that face them as well as the exposure and vulnerabilities they have. A vulnerability assessment will identify the current state of the security architecture and help identify where improvement is needed. For a vulnerability assessment to be meaningful, it must be done on all levels of an organizations. The computer network must be looked at in its entirety from the IT side and the ICS side. Such an assessment would not only take into account electronic resources, but it would also look at physical security, such as gates and locked rooms, etc.

The assessment also extends to personnel. Social engineering is one of the weakest links in any cyber security master plans. Most people are inclined to be helpful, hackers know this and this makes employees a good target. When a help desk employee receives an e-mail requesting a password be changed or a receptionist at a front desk of a large corporation is presented with an opportunity to help someone out of a tight situation, without the proper training, they might hand over a critical piece of information that could result in a cyber security breach. Because of this, employee training and a well-established security policy is a key component to any effective defense-in-depth architecture.

In a SANS Institute whitepaper [46], Kevin Mitnick emphasized the importance of comprehensive employee training over just a well-defined security policy by stating: “The methods that will most effectively minimize the ability of intruders to compromise information security are comprehensive user training and education. Enacting policies and procedures may not be effective: my access to Motorola, Nokia, ATT(sic), Sun depended upon the willingness of people to bypass policies and procedures that were in place years before I compromised them successfully.”

No one security mechanism alone will effectively protect an ICS. ICS systems are insecure by design and when connected to the external world through the corporate network it exposes the network to a number of threats. An effective security strategy will include network segmentation using firewalls/routers, good physical security, strong security policies and well trained employees.

2.4 ICS Specific IDS Techniques

Since ICS security is a relatively new field, many attempts have been made to apply traditional IT approaches to ICS. Since many of the same hardware components (routers, Ethernet, etc.) are used in the ICS world, this was an obvious assumption. Unfortunately, this assumption has been proved to have been made in error due to the inherent differences between IT and ICS. The availability that is required in an ICS puts strong demands on any security feature that must analyze network traffic in real-time without causing a noticeable delay. Since real-time events are taking place on ICS networks, such as actuators moving, the timeliness of packets being received is of the utmost importance. On an ICS network, the validity of the data in a packet can sometimes have a lifespan of milliseconds. Also, the insecure by design hardware and protocols found on a typical ICS lack some of the basic authentication features that are helpful when securing a typical IT network. Because of this, different approaches must be taken when applying intrusion detection techniques to ICS security. The following paragraphs will look at some work that is being done specific to ICS specific intrusion detection systems [47].

In [48], three model-based techniques were developed and were implemented specifically for the Modbus TCP protocol. This work takes advantage of the static network topology and regularity of network traffic. The first of the models described is a protocol-level model for Modbus TCP. This model defines what a normal Modbus TCP packet should look like and the acceptable values contained within. This model allows for the IDS to alert if an undefined function code is present in the packet. An undefined function code could be an indication that reconnaissance activity is taking place. The second of the models described in this work is based on expected communication patterns. Some devices on a network always communicate with specific devices. Some devices always listen and some devices might always send information. Careful modeling of this routine behavior can help identify when abnormal communication occurs on the network. The last model employed uses a machine learning approach for detecting changes in server or service availability. The system can be monitored to alert if a device that normal services a particular response fails to do so.

[49] presents a model for integrating an IDS into an embedded system using middleware called EMIDS (embedded middleware-level intrusion detection system). They considered this approach because the middleware framework had access to both the communication streams and application layer for the embedded device. The middleware can then forward communication information to the IDS and alleviate the embedded device from processing information that it might not have the processing power or resources to handle. EMIDS uses sensors embedded into the middleware framework that are classified as either interval-based, procedural-based or misuse-based. Interval-based sensors monitor the frequencies of certain communications taking place and report this information to the IDS. Procedural-based sensors collect data based on the execution patterns and typically are positioned at the entry and exit points of an applications functions. Misuse-based sensors are located in the applications' source code at locations were known vulnerabilities exist.

[50] discusses the implementation of a SCADA power-grid testbed for intrusion detection and event monitoring. Their testbed leveraged the facilities at the University of Idaho's Electrical Engineering Power Laboratory in order to produce real-world results.

In the prototype system a method for updating device settings was created using the Perl programming language. This program would facilitate communications from the operator terminal to the telnet program that communicates with the SCADA devices. The Perl program was used to provide more secure communications between the two devices via SSL and ssh and it allowed for the logging of commands and settings modifications. Another component that was added was uptime monitoring using ping and telnet. The uptime monitoring was very effective at detecting faulty devices and network paths. Information about device access, settings modifications, uptime monitoring and other static values such as IP address, telnet port and legal commands were detailed using XML. A Perl program was then used to parse the XML files to generate Snort IDS signatures. Not all of the IDS signatures could be created automatically. Rules for failed login attempts had to be manually created. The prototype discussed in this work proved to be useful, however the current prototype only automatically generated rules for RTUs. The functionality discussed is currently being expanded to other devices.

Morris, Vaughn and Dandass [51] take a look at retrofitting serial based industrial control systems that use the MODBUS protocol to allow for IDS implementation using Snort. Some security professionals might consider serial connections to be secure, however a compromised PLC that has serial connectivity could compromise the serial devices that it communicates with. The researchers used MODBUS RTU/ASCII Snort, a software developed to allow Snort to monitor MODBUS traffic, to capture serial traffic. The MODBUS RTU/ASCII Snort can run in passive mode (bump in the wire) where it only listens or as an active device (inline) that intercepts packets and retransmits them to their final destination. Once the serial traffic is intercepted it is converted to MODBUS

TCP/IP and transmitted to the Snort IDS via a virtual Ethernet with two virtual machines. (The paper details the conversion from MODBUS RTU/ASCII to MODBUS TCP/IP.) In passive mode, the first virtual machine (VM1) captures the data from the MTU upstream direction and the second virtual machine (VM2) captures the data from the RTU downstream direction. Since only the RX pin of the serial connection is monitored, the passive configuration can only monitor; therefore, it is unable to block or drop packets. In passive mode, only VM1 runs Snort to analyze the information. The inline configuration still uses two virtual machines but each has the ability to transmit. VM1 sends and receives information from the MTU and VM2 sends and receives information from the RTU. When traffic is captured by either VM, it is processed by Snort, then forwarded on to the second VM so it can be passed along to the endpoint. The inline configuration provides for the added functionality of dropping suspect packets, but at a cost of increased processing/transmission time.

Caselli, Zambon and Kargl [52] explore the concept of a "semantic attack" and propose a sequence-aware IDS (S-IDS) method as a countermeasure. Some cyber-attacks attempt to steal proprietary information or cause a disruption in the process, but a semantic attack is classified as an attack that tries to maximize the damage to the physical pieces of the control system. This might be causing a centrifuge to tear itself apart [53] or cause a furnace to burn out of control [54]. Semantic attacks rely on knowledge about the control system, the protocols used and the physical components that it controls. Many times these attacks are carried out using valid commands that would be impossible to detect if the IDS was only looking for malicious activity. Oftentimes, a system can be damaged by issuing valid commands in the wrong sequence. This type of attack is called a sequence attack. In an effort to better detect sequence attacks, the S-IDS is based on a layered structure. The lowest layer is called the Reader. The Reader captures raw packets, filters corrupt or redundant packets and generates a formatted input stream that is used by layer 2 of the S-IDS architecture. Layer 2 is the Sequencer. This is the core of the architecture. A predefined set of rules are applied to allow the Sequencer to organize the information from the Reader into a sequence of events. The third layer is the Modeler. This layer takes the sequences from the Sequencer and uses that information to build models that represent how the system should behave over time. This work uses discrete-time Markov chains (DTMC) to create a model for the system. The last functional layer of the process is the Detection layer. The Detection algorithm compares the sequence models that are being monitored with normal system behavior models that were trained when no malicious activity was present. This S-IDS approach was tested using real-world data from a water treatment facility and was shown to correctly identify sequence attacks as well as keep the number of false positives low.

This literature review surveyed some of the current approaches to securing an industrial control system. It also highlighted the fact that no single security measure is enough and many different approaches must be taken. The importance of IDS security was made clear with the various papers discussed in this section and the pros and cons of each type were discussed thoroughly. As seen in this section, much work has been done in the area of IDS research and continued focus has been put on creating a dependable IDS that is accurate and easy to implement in an ICS. As stated, signature-based IDSs lend themselves well to control systems because of the static nature of the system [55].

Because of that, this thesis provides a novel approach to signature generation through ladder logic analysis.

CHAPTER III

IDS SIGNATURE CREATION USING LADDER LOGIC

3.1 Overview

A PLC ladder logic program contains a great deal of information that can be leveraged to produce a set of signature based IDS rules to help identify abnormal activity on an industrial control system network. The ladder logic is made up of contacts, coils and instructions that instruct the PLC how to orchestrate a control system. Each contact, coil and instruction has parameters that can either be a constant value or an address register that acts as a variable. By analyzing this information, there are two main categories of IDS signature sets that can be identified: address register usage and invalid register values.

3.2 Address Register Usage

The first key piece of information that can be obtained by inspecting a ladder logic program is the address register usage. Knowing what address registers are used in a ladder logic program can quickly provide a lengthy set of IDS rules identifying registers that should never be written or accessed. An attacker that had no knowledge of the industrial control network would need to blindly guess which address registers to write using a brute force method. If the IDS was setup to identify attempted writes to a list of unused address registers, then this attack could quickly be identified. A simple scan of the PLC ladder logic can provide this information.

3.3 Invalid Register Values

The second key piece of information that can be obtained by inspecting a ladder logic program is invalid register values. An invalid register value is a value that, if written into a certain PLC address register, would result in a PLC processor fault. On some PLCs if the PLC processor faults, then all control will cease and the processes it controls will stop or be in an uncontrolled state. There are many different types of invalid register values that can be discussed and these will vary from manufacturer to manufacturer and product type to product type. Some examples of invalid registers values include: division by zero, math overflow, negative values and invalid addressing.

3.3.1 Division by Zero

A division by zero fault occurs when a variable address register is located in the denominator of a division math instruction and that address register is inadvertently set to a zero. Once this occurs, some PLC processors will fault and control functionality will cease. PLC division instructions are widely used in some of the most basic process control ladder logic programs. Their frequency of use makes them a good target for malicious activity. It is not possible to write an IDS rule for every division instruction, only the ones that have a constant for a numerator and an address register for the denominator. When an address variable appears in the numerator, as well as the denominator, it quickly becomes difficult to write an IDS rule for what would constitute a division by zero since the numerator is no longer static. For the purposes of this thesis, only division instructions that have a constant in the numerator are used to create an IDS rule.

3.3.2 Math Overflow

A math overflow fault occurs when a value outside of the computational range of the processor results in a register. The acceptable range is defined by the type of processor. The Allen Bradley SLC 500, used for testing, has a 16-bit architecture, therefore the allowable range for any integer math register is -32767 to 32767. Any value outside that range will result in a math overflow fault. Math overflow faults can occur as a result of a value being written directly to the register from an external source over the network or written to a register as the result of an internal math function. There are several commonly used math instructions that can result in a math overflow fault. Since addition instructions and multiplication instructions are the most susceptible to overflow, these two instructions are of focus in this paper. An attacker could easily write large values to random address registers and would eventually cause a math overflow fault. An IDS rule that monitored for this condition could easily prevent or draw attention to suspicious or malicious activity.

3.3.3 Negative Value

A negative value fault occurs when a negative integer appears in an address location that limits its values to positive integers only. Two examples of this are the Rockwell Software Logix 500 counter and timer instructions. There are two major types of counter instructions: count up (CTU) and count down (CTD). Both of these instructions have custom parameters called preset (PRE) and accumulators (ACC). The preset value is the value the counter needs to reach before the instruction becomes true. The accumulator value is the current value of the instruction. If either of these values contain a negative integer, the processor will fault. Similarly, the preset and accumulator parameters can be found for the various types of timer instructions. There are three types of timer instructions that are focused on in this paper: timer on delay (TON), timer off delay (TOF) and retentive timer on (RTO). A negative value in any of the preset or accumulator values for these instructions will result in a processor fault.

3.3.4 Invalid Address

An invalid addressing fault occurs when a reference is made to an address that doesn't exist. There are two types of addressing that can be found in most Rockwell Software programming platforms. Direct addressing is when an address is explicitly stated and constant (e.g. N7:0). Indirect addressing is when the word portion of the address is a variable, for example N7:[N10:0]. In the example provided, the value in N10:0 will determine what word of N7 is addressed. If N10:0 contains a value that is outside the defined range of N7, then an invalid addressing value would occur and the processor would fault. An error would also occur if a negative value was written to N10:0.

3.4 Testbed: SLC 500 and RS Logix 500

The testing and vulnerabilities described throughout this paper are specific to the Allen Bradley SLC 5/05 processor platform and Rockwell Automation's Ladder 500 programming language. The SLC series of hardware is modular and chassis-based. A typical system usually consists of a rack (multiple slot size options), a power supply, IO cards and a processor. Allen Bradley offers a wide array of IO cards ranging from discrete IO, to analog IO to specialty IO. The SLC series of hardware has several different processors to choose from with memory and communications being the determining factor as to which one to us for an application. The SLC 5/05 processor is the only processor in the SLC line that offers Ethernet/IP communications. Ethernet communication occurs at 10 Mbps or 100Mbps. The SLC processor is a 16-bit architecture with memory options ranging from 16k to 64k. The SLC systems can be configured with a maximum of 3 independent chassis for a total of 30 slots and it can have up to 4096 individual I/O points. For testing purposes, the following hardware was used: SLC 5/05 processor, 3 IO cards, power supply and 4-slot rack.

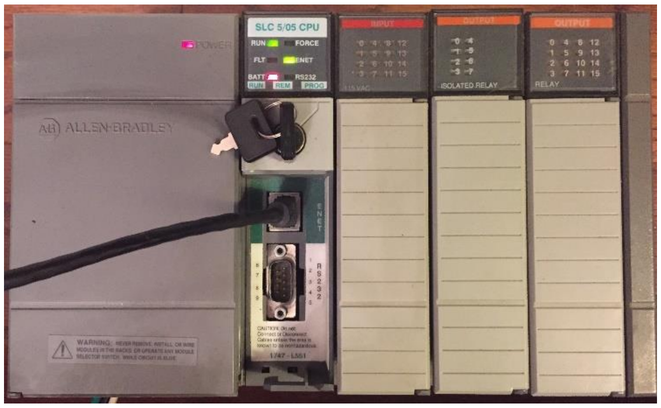


Figure 3.1 SLC Hardware

The SLC family of hardware was introduced in the 1990's. Since that time Allen Bradley has introduced more sophisticated platforms such as the ControlLogix processor, however due to the reluctance of industry to replace hardware and the "if it's not broke don't fix it mentality," the SLC 500 market share is still quite large and responsible for controlling large manufacturing and processing operations throughout the world. Because of this large presence, it is still necessary to expose vulnerabilities and provide methods for protecting industrial control systems that still use this aging hardware.

There are two types of faults that occur in this type of processor: minor faults and major faults. Minor faults are typically just a nuisance fault indicating that something in the system needs attention. This might be a warning that the processor battery is low or something similar to that. These fault indications are usually passed along to an HMI or a signal light to alert the operator that something needs attention. A major fault, on the other hand, has a more dramatic impact on the process. A major fault occurs when an error condition is present that prevents an instruction from executing. When this happens, the current instruction is aborted and the fault is reported using status registers. A major fault will halt all logic execution and the controller will switch from "Run Mode" into a "Fault Mode." Once the processor is faulted, all output functionality ceases and the outputs are left in whatever state they were in when the error occurred. Depending on the process it is controlling this could put workers in danger or have serious health, economic or environmental implications. There are measures that can be put in place to control what happens under certain instances, but this does not account for all major fault conditions.

The SLC family processors are programmed using Rockwell Software's Logix 500 programming package. This package is an IEC-1131 compliant ladder logic program that offers a graphical interface for modifying and monitoring the PLC program. Although the newer software packages have migrated to a tag based ladder, the Logix 500 programs still use an address based approach. The newer tag based ladder editors allow a programmer to name inputs, outputs and internal registers with a meaningful name. For example, an input from an emergency stop pushbutton might be named "EmergencyStopPB." The Logix 500 interface does allow a programmer to add comments to provide better readability, but the basic addressing follows the format of file type, file number, word number and bit number. The following chart shows the basic types of files.

Table 3.1 Logix 500 Address Types

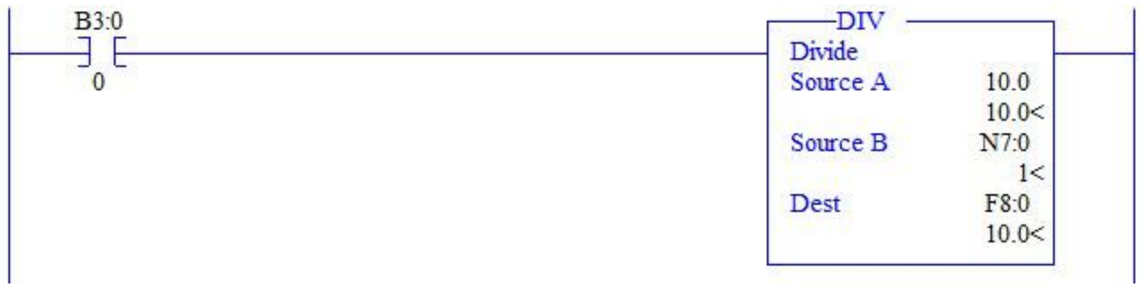
File Type	Identifier	Numerical File Type	Example Address
BINARY	B	85	B3:0/0
TIMER	T	86	T4:0
COUNTER	C	87	C5:0
INTEGER	N	89	N7:0
FLOAT	F	8a	F8:0

Although this programming package has been around since the debut of the SLC 500 series of processor, it continues to be used widely today as a programming platform for the Micrologix family of processors and other newly released Allen Bradley processors.

The IDS signatures generated for this thesis are developed for Snort. Snort is an open-source, lightweight intrusion detection system that is based on the libpcap packet sniffer and logger. Snort can be on many different types of operating systems and can easily be deployed on most any node of a network because it has very little impact on the host system. Snort has a built in packet decoder that can analyze multiple layers of the TCP/IP stack. The detection engine uses pattern matching to compare the decoded raw network traffic to a set of known signatures. Once a match has been found the packet can either be logged for further analysis or an alert can be generated [56].

3.5 Converting Graphical Ladder Logic into Text

Most ladder logic programs are graphical in nature which makes them difficult to parse; however, some ladder logic programs can be copied and pasted into a text format that is easily searched for key information. For instance, Rockwell Software's Logix 500 programs can be converted to a text file by highlighting the rungs of the graphical ladder program and using the Windows copy/paste commands to paste that information into a text file. Each ladder element is represented by a specific keyword and constant/variable information. The keyword identifies the type of instruction for the ladder element and the parameters that follow the keyword give more detail as to how the instruction is used. For example, the following figure shows a single rung of ladder logic containing a normally open (NO) contact and a division (DIV) instruction along with its corresponding text representation.



SOR XIC B3:0/0 DIV 10.0 N7:0 F8:0 EOR

Figure 3.2 PLC Ladder Rung and Text Representation

The text representation in Figure 3.2 contains the following information:
SOR – Start of rung
XIC B3:0/0 – Examine if closed (normally open) contact with address B3:0/0
DIV 10.0 N7:0 F8:0 – Division instruction, 10 divided by N7:0 = F8:0
EOR – End of rung

3.6 IDS Rules Identified via Ladder Logic

There are many types of PLC instructions and they vary from manufacturer to manufacturer. There are many basic instructions that are common across most types of PLC ladder logic. Basic math instructions such as addition, subtraction, multiplication and division are very common instructions that can be found in most off the shelf ladder programming packages. The Allen Bradley SLC 5/05 processor uses these basic functions on a routine basis in even the most elementary ladder programs. As mentioned earlier, the SLC 500 series of processors have several vulnerabilities that can be exploited mainly due to its 16-bit architecture and its lack of intrinsic error handling. These major faults can result in the total halt of all control functions of the processor.

The following vulnerabilities can easily be detected by parsing through ladder logic and used to create Snort IDS rules:

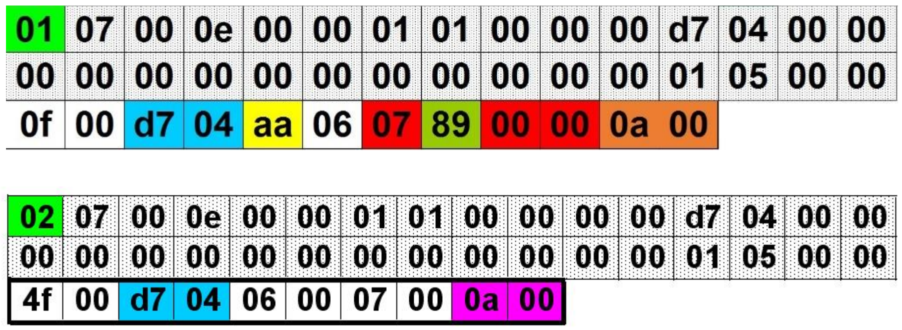
- Division by Zero Fault
- Math Overflow Fault
- Negative Value Fault
- Invalid Addressing Fault

These potential faults along with the detection of unused registers can provide a significant list of signatures to be used as input to a IDS in an effort to monitor the network and protect the industrial control system.

3.7 SLC 500 Ethernet Packet Format

In order to understand the IDS rules that are discussed in the next section, it is important to understand the structure of a typical SLC 500 Ethernet packet. The SLC 500 Ethernet packet encapsulates an older protocol called DF1. The DF1 protocol has been around for a number of years and was primarily used for serial and Allen Bradley's DH+ physical layers. Once Ethernet emerged onto the industrial control network scene, Allen Bradley encapsulated the already proven DF1 protocol into the Ethernet packet. The documentation that is published for the DF1 protocol is very abstract, therefore much of what was learned for the purposes of this paper was learned through experimentation using a mock control system network.

In an effort to reverse engineer the Ethernet encapsulated DF1 protocol, a laptop running Wireshark and Nmap was connected to a mock control system network consisting of a PLC and HMI. Data was then recorded and analyzed. The figure below shows a sample Ethernet command and reply packet. This packet depicts an integer write (aa) of the value 10 (hex 0a) to PLC integer register N7:0.



```
01 07 00 0e 00 00 01 01 00 00 00 d7 04 00 00
00 00 00 00 00 00 00 00 00 00 00 01 05 00 00
0f 00 d7 04 aa 06 07 89 00 00 0a 00

02 07 00 0e 00 00 01 01 00 00 00 00 d7 04 00 00
00 00 00 00 00 00 00 00 00 00 00 00 01 05 00 00
4f 00 d7 04 06 00 07 00 0a 00
```

Figure 3.3 Example Ethernet Command (top) & Reply (bottom) Packet

The Ethernet and TCP header portions of these packets have been omitted and only the data portion is shown above. The first item to be noted is the repetition of the shaded portion between the command and reply packet with the exception of the initial byte (highlighted in green). This repetition and simplicity of the packet, make it easily susceptible to a man in the middle attack or response injection. The unique portion of each packet is highlighted with a bold border in the figure above. Since this sample packet is representative of an integer value, only two bytes are shown at the end of the packet (orange). A floating point value would have two additional bytes at the end of the packet to hold the four-byte IEE 754 Floating Point value. The table below (Table 3.2) gives a brief explanation of each byte as observed through various tests.

Table 3.2 Command/Response Packet Byte Descriptions

Unique Seq. Number	This value links the command and reply packet.
Command	Specifies the process to be performed (a2 = read, aa = write, etc.)
Address	Address to be inspected. In this example the address is N7:0
Address Type	Specifies the type of value located at the address. (89=N=integer)
Write Value	The value to be written (10 or hexadecimal 0a)
Return Value	Return value at specified location. In this example N7:0 = 10 (or hex a)

The unique sequence number (light blue) is retrieved from the PLC (host) when a client (laptop, HMI, etc.) establishes communication. This number increments with each command/response packet exchange. This is a simple method for matching up command and response packets.

The command portion (yellow) specifies the action that is to take place. The DF1 protocol has a lengthy set of commands, but most are very specialized commands like for uploading and downloading programs to the PLC.

After exhaustive testing, the most typical commands used for routine industrial control are the following:

a2 – protected typed logical read

aa – protected typed logical write

ab – bit level write

The address bytes (red) refer to the file number and word number of an address register. The left most value is the file number and the two bytes grouped to the right are the word number. In the example above, the file number is "07" and the word number is "00 00". This packet is addressed for N07:00. Each file number is unique and can represent any type of value (integer, float, etc.) This is discussed in more detail below. Since this value is limited to one byte, then the largest file address for a SLC500 is 255.

The address type (green) indicates the data type of the value stored in the target register. This numeric representation could indicate the target register is an integer, float, counter, timer, string, etc.. This was explained in detail in a previous section, Table 3.1.

The write value (orange) is found at the end of the command packet and its byte length varies depending on the data type being written. An integer write will have a maximum of two bytes for the value at the end since the maximum 16-bit integer is 32,767 (7F FF hex). However, a floating point write would have four bytes for the value at the end in order to hold the IEE 754 Floating Point Standard value.

The return value (violet) in the response packet is confirmation of that the command was executed successfully. If a read command is issued, the return value will contain the bytes representing the value, of the specified type, in the address register being read. If a write command is issued, the return value will contain the bytes representing the value that was written. It is important to note that the read and write values are stored in little endian format.

3.8 Ladder Logic Parser Program and Snort Rule Creation

For the purposes of this paper a Microsoft Visual Basic .NET program was developed to parse through ladder logic code to create Snort IDS rules for the vulnerabilities identified in the previous section. The ladder logic code was exported to a text file to present the VB.NET program a format that was easily traversed and Snort IDS rules were created for each possible vulnerability.

Snort is very versatile and offers many different options for what to do with a packet once a signature is identified. The testing done for this paper uses the alert functionality in order to log that a rule was successful in matching a known signature. In a real world scenario, the IDS might be setup to drop the packet to prevent a failure from occurring. Each Snort rule was written using the payload detection rule option, byte_test.

The byte_test option tests a field against a specific value. The format of the byte_test is as follows:

```
byte_test:<bytes_to_convert>,<comparison operator>,<value>,<offset>,
                    <relative>,<endian>
```

The first parameter, <bytes_to_convert>, is the number of bytes to inspect. This varies from ladder instruction to ladder instruction depending on what is being inspected. This value is converted to an integer value and compared to the value directly after the comparison operator. The comparison operator is the operation to be performed (>,<,=, etc.). The <value> parameter is what the first parameter is tested against using the comparison operator. The <offset> parameter allows the user to specify the starting point of the <bytes_to_convert> in relation to the Content portion of the rule. The <relative> option indicates that this offset is relative to the Content. The last parameter, <endian>, indicates the byte order of the <bytes_to_convert>.

The header information for each Snort rule in the following section indicates that the rule will alert when any TCP/IP traffic, with payload information matching the defined pattern, is directed to port 2222 on the PLC's hardcoded IP Address (192.168.0.109). Port 2222 is the standard communication port for many Allen Bradley hardware products with Ethernet communications. The rule header information is left out of the following section for neatness and referred to as {rule header info}. The following substitution can be made whenever this placeholder is found throughout this paper.

{rule header info} = alert tcp any any -> 192.168.0.109 2222

The next section describes each IDS rule creation scenario that was explored for this paper and detail how to transition from a graphical ladder element to a Snort IDS rule.

3.8.1 Division by Zero Rule Creation

The division (DIV) instruction has three parameters: Source A, source B and Dest. Source A is the numerator and can be either a constant value or an address register. Source B is the denominator and can also either be a constant value or an address register. Address registers can be of type integer or float for both the numerator and the denominator. The Dest. parameter, or destination, is required to be an address register. This register can either be an integer or a float, however if an integer value is specified then the quotient is truncated after it is calculated. The graphical representation of the Logix 500 DIV instruction is below.

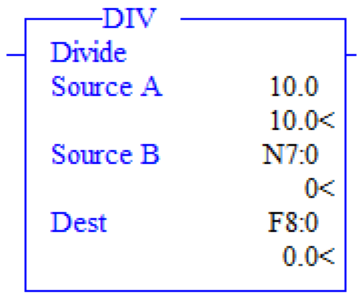


Figure 3.4 Logix 500 DIV Instruction

Once exported to a text file, this instruction is represented as DIV 10.0 N7:0 F8:0

The Visual Basic parser program searches through the exported ladder text looking for DIV instructions that contain an address register in the denominator parameter. All DIV instructions with a constant value in the denominator are ignored. An address register in the denominator acts as a variable that could be intentionally set to a zero through malicious activity on the network. As previously discussed, if a division by zero fault occurs all PLC control functions will cease. In order to protect against this, the ladder parsing program automatically creates the following Snort IDS rule.

```
{rule header info}  (content:"|aa02078900|";byte_test:2,=,0,1,relative,little;sid:100000;)
```

3.8.2 Math Overflow Rule Creation

A math overflow fault occurs when a register exceeds the allowable range for that register type. The SLC 500 16-bit registers are limited to (+/-) 32,767 for integer registers and 3.4e+38 for floating point registers. There are three common types of instructions where a math overflow could occur: addition (ADD), multiplication (MUL) and subtraction (SUB). The graphical representations of each of these Logix 500 commands are shown in the figures below.

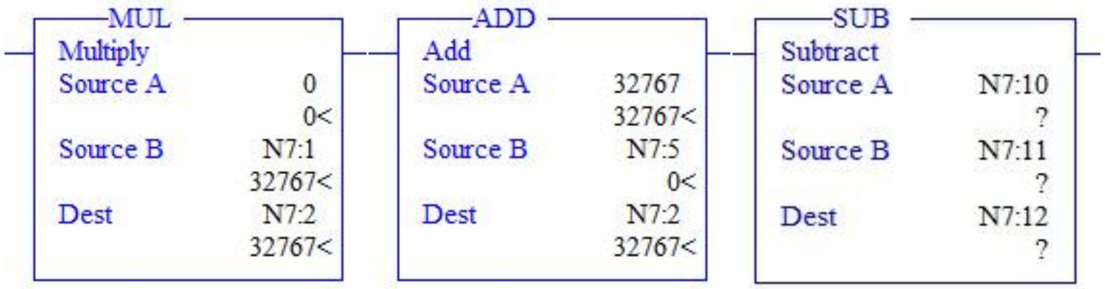


Figure 3.5 Logix 500 MUL, ADD & SUB Instructions

Once exported to a text file, each of these instructions are represented as

MUL 0 N7:1 N7:2 ADD 32767 N7:5 N7:2 SUB N7:10 N7:11 N7:12

Each math command has the same parameters as the DIV function discussed earlier. In order to identify which math command could result in a math overflow condition, one of the Source parameters must be a constant and the other must be a variable address register. If both of the parameters are address registers, it might be possible to create a rule to detect a math overflow, but that is beyond the scope of this paper. If both parameters are constant or both are address registers, then the instruction will be ignored by the parser program. Once a math instruction is found that meets the required criteria, then the value that creates an overflow condition is calculated and the rule is written. For example, in the ADD instruction above Source A is a constant value set to 32767. That means anything greater than zero in Source B will create a math overflow fault. The ladder parsing program would automatically create the following Snort IDS rule based on this information.

```
{rule header info}  (content:"|aa02078905|";byte_test:2,>,0,1,relative,little;sid:100000;)
```

3.8.3 Negative Value Rule Creation

A negative value fault occurs when a negative value is present in a register reserved for positive numbers. Two instances where this might be a problem is in the accumulator (ACC) and preset (PRE) registers of timer and counter instructions. The graphical representation of each of these Logix 500 instructions are shown in the figures below.

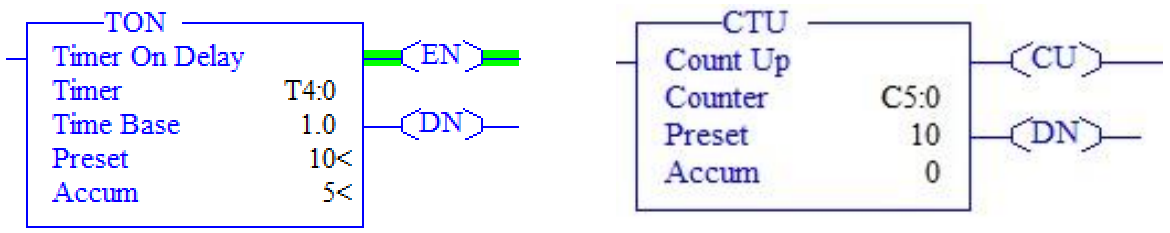


Figure 3.6 RS Logix 500 TON and CTU Instructions

Once exported to a text file, each of these instructions are represented as
TON T4:0 1.0 10 5 CTU C5:0 10 0

Timer and counter instructions are very similar. The first parameter specifies the file number of the timer or counter. By default, timers are found in file number four (T4) and counters are found in file number five (C5), however other file numbers can be added if required. The second parameter found in the timer instruction is not present for counter. On timers the second parameter is the time base. This can have a value of 1.0, 0.01 or 0.001 (seconds). The time base parameter defines the units for the timer Preset parameter. Both timers and counters have a Preset parameter. The Preset is the target for the instruction. As long as the rung containing a timer instruction is true, the timer keeps counting until the Preset value is reached. Each time a rung containing a counter instruction transitions from false to true, the counter increments by one until the Preset value is reached. For both the timer and counter instructions, the Accumulator value is the current value of the timer/counter. Once the Accumulator value equals the Preset value, then the target is reached and the instruction's DN (done) bit will transition to logical high.

To prevent a negative value fault, the ladder parser will create a rule for all timer preset and accumulator registers and counter preset and accumulator registers that appear in the ladder logic. An alert is generated if a negative value is written to any of these prohibited registers. For the timer and counter instructions shown above, the following four rules would be created.

Timer Preset:

```
{rule header info} (content:"|aa0204860001|";byte_test:1,>,128,1,relative,
                    little;sid:100000;)
```

Timer Accumulator:

```
{rule header info} (content:"|aa0204860002|";byte_test:1,>,128,1,relative,
                    little;sid:100000;)
```

Counter Preset:

```
{rule header info} (content:"|aa0205870001|";byte_test:1,>,128,1,relative,
                    little;sid:100000;)
```

Counter Accumulator:

```
{rule header info} (content:"|aa0205870002|";byte_test:1,>,128,1,relative,
                    little;sid:100000;)
```

Three things to note about these rules that differ from other examples to this point:

1. The Content portion of the rule has an additional byte listed at the end. This byte is a “01” to address to Preset portion of the register and a “02” to address the Accumulator portion of the register.

2. The file types for these rules, the fourth byte in the Content portion, is “86” for timer registers and “87” for counter registers.

3. The value that is compared against in the byte_test portion is “128.” This is because a negative number is present when the 16th bit (sign bit) of the two bytes is a “1”. Therefore “1000000000000000” is equal to hex “80 00.” If the most significant byte only is inspected, then 0x80h is 128 in decimal.

3.8.4 Unused Address Rule Creation

The PLC ladder logic program defines all of the addresses that should be in active use for controlling a system. Any read or write attempts to address registers that are not defined in the PLC ladder program should be flagged as suspicious. An attempt to write unused registers don’t present any harm to the system, however a brute force attack attempted by someone with little knowledge of the system could be identified by these unexpected write attempts. An attack or reconnaissance attempt would undoubtedly attempt to read or write random registers.

The ladder parser program logs every address register that is referenced in the PLC ladder logic. Once this information is gathered, the address registers can be organized so IDS rules can be written for entire file numbers that are not used, entire words within a specific file that are not used or specific bits within a word that are not used. These unused addresses can be written at the word level or at the bit level so it is important to write rules that identify both types of write attempts. The following rules are examples that might be generated from a typical PLC ladder program for unused address registers.

Example 1: Alert if a word level write attempt is made on any integer address greater than N77:22

```
  {rule header info} (content:"|aa027789|";byte_test:1,>,22,0,relative, little;sid:100000;)
```

Example 2: Alert if a bit level write attempt is made on any integer address greater than N77:22

```
  {rule header info} (content:"|ab027789|";byte_test:1,>,22,0,relative, little;sid:100000;)
```

CHAPTER IV

TESTING AND EVALUATION

4.1 Test Environment/Equipment

Testing and evaluation of this topic required a specific hardware setup consisting of an Allen Bradley SLC 500, an HMI interface/client PC to serve as an attacker and a PC running the IDS software to monitor the network traffic.

4.1.1 PLC Configuration

The PLC used for testing was an Allen Bradley SLC 500 system with a 5/05 CPU (part number 1747-L551B/C) and 16K of memory. The processor was housed in a 4-slot rack with the power supply. A simple RS Logix 500 ladder logic program was created to simulate each of the rule creation opportunities discussed in this paper.

4.1.2 HMI Interface

A Microsoft Visual Basic .NET program was developed to serve as a client HMI to communicate with the PLC. The HMI had various numeric indicators to display information read from the PLC and input boxes to allow information to be written to the PLC registers. This interface was used to compromise the PLC by writing values to address registers that would halt the PLC processor by creating a division by zero, math overflow, or negative value fault to occur.

4.1.3 IDS Monitoring PC

The IDS Monitoring PC was a laptop running Snort version 2.9.2. The signatures were created for the IDS using the Microsoft Visual Basic ladder logic parser program described in previous sections. The number of rules that were being actively monitored were dependent on the size and type of ladder logic program analyzed by the parser program. Each time an IDS alert was received it was logged for further inspection.

4.1.4 Network Layout

The network consisted of the three devices mentioned above connected to a simple 4-port network hub. A hub was used to ensure that all traffic between the PLC host and the HMI client could be inspected by the IDS PC. A network layout diagram is shown below in Figure 4.1.

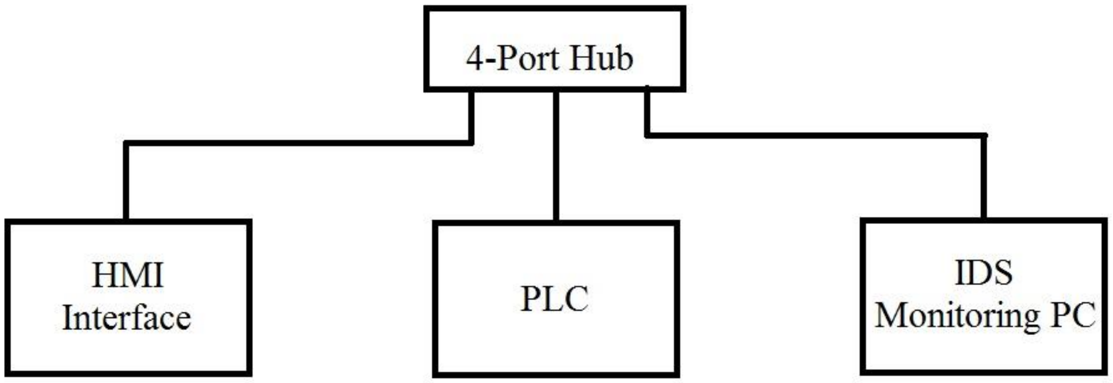


Figure 4.1 Test Network Diagram

4.2 Analysis of Ladder Logic

The ladder logic parser program was developed and tested using a very simple ladder logic program that was written to demonstrate one of each type of vulnerability described in this paper. This proved to be very effective for development purposes. The penetration testing that was done using the IDS rules created for the small test program demonstrated that the ladder parser was very effective at creating Snort signatures to alert if any of the known attacks were made against the PLC.

Although a simple PLC program was sufficient for development and testing purposes, it is not typical of what would be found in a manufacturing environment today. In order to get a better idea of the potential effectiveness of the ladder logic parser rule creation process in a real world environment, seven different ladder logic programs were analyzed. Each of the seven ladder logic programs are actual real world examples of programs that are in execution today at various manufacturing facilities around the United States. For confidentiality purposes, the name of each company is not mentioned, but referred to as Plant 1-7 instead. To show the wide variety of industry reflected, a brief description of manufacturing facility is as follows:

- Plant 1 produces air conditioning component, compressors and entire heating/cooling units. The ladder logic program is from a large conveyor assembly line that assembles compressors.
- Plant 2 produces home appliances. The ladder logic program is from a robotic work cell that cuts holes in a blow molded part.
- Plant 3 is a tier 1 supplier to the automotive industry. The ladder program is from an assembly cell that produces transmission components.
- Plant 4 is a tier 1 supplier to the automotive industry. The ladder program is from a conveyor line that produces seating components.

- Plant 5 is a tier 2 supplier to the automotive industry. This company produces various metal components associated with vehicle suspension.
- Plant 6 is a tier 1 supplier to the automotive industry. The ladder program is from a seating line that controls multiple assembly stations along the conveyor line and several stand-alone assembly stations.
- Plant 7 is a tier 1 supplier to the automotive industry. The ladder program is from an assembly machine that folds airbags before they are place into the protective sleeves.

The data in Table 4.1 shows the various fault related IDS signatures that can be created by parsing the ladder logic of each of the real world examples. There are a few things that can be noted by looking at the data. First, none of the sampled ladder programs had a division (DIV) instruction that matched the criteria required to be vulnerable to a divide by zero fault. This is not surprising given that no process control related ladder programs were available for parsing. A chemical plant or other process control related system would certainly lend itself to more equations and math functions that would probably include the use of more division operations.

Table 4.1 Rule Count for PLC Major Faults in Sample Ladder Logic Programs

Plant	Divide by Zero	Math Overflow	Negative Error	Indirect Address	Program Size (Total Rungs)
Plant 1	0	92	90	0	338
Plant 2	0	101	98	0	214
Plant 3	0	64	64	0	174
Plant 4	0	1140	990	1326	2353
Plant 5	0	378	378	0	989
Plant 6	0	1376	1360	33	2468
Plant 7	0	306	306	0	658

The second item to note is that not all ladder programs use indirect addressing. This is something that is left to the programmer's personal preferences. It is certainly possible for a control systems engineer to go his entire career without using indirect addressing, however indirect addressing has its advantages in certain applications. Out of the seven programs sampled for this dataset, only two included indirect addressing. As the data shows, one ladder program uses indirect addressing extensively.

The data in Table 4.2 shows the number of unused address register IDS signatures that can be created by parsing the ladder logic of each of the real world examples. Each of the rules can be written to cover a range of addresses at the file level, word level or bit level. The level on which the rule is written varies depending on how the address register is used in the ladder logic program. For example, if no individual bit references are used in the ladder logic program, then a rule can be written at the word level to cover that range. How the programmer chooses registers is also a big factor on the number or unused address rules that are created. If the unused registers are consecutive, then larger blocks of unused addresses can be covered in a single rule. On the other hand, if the address usage is very sporadic, then multiple rules would need to be written to cover all of the gaps.

Table 4.2 Rule Count for Address Faults in Sample Ladder Logic Programs

Plant	Unused Address Word	Unused Address File	Unused Address Bit	Program Size (Total Rungs)
Plant 1	42	22	142	338
Plant 2	88	44	124	214
Plant 3	82	32	10	174
Plant 4	346	102	1744	2353
Plant 5	292	36	564	989
Plant 6	478	62	1856	2468
Plant 7	94	32	195	658

4.3 Validation

Because of the large number of rungs in each of real-world sample ladder logic programs and the likelihood of human error, it would be impractical to do a manual verification of each signature that was generated. Therefore, validation of this rule generation technique was done on a smaller scale. A PLC ladder file was created that simulated each of the vulnerabilities mentioned in this thesis. The PLC file was downloaded to a SLC 500 processor and each vulnerability was verified as valid by sending a value to the address register that would cause a fault to occur.

Once each fault condition was validated by a manual attack, the sample ladder logic program was processed by the Ladder Logic Parser program. The Ladder Logic Parser program not only generated a set of Snort rules that directly addressed the vulnerabilities represented in the sample test program, but also generated rules related to the unused address ranges. The generated rules were copied over to the IDS Monitoring

PC for testing. Once Snort was put into monitor mode, each manual attack was initiated one-by-one to verify that Snort alerted to indicate the rule was successful in recognizing the packet. Other network traffic was introduced to the test network to simulate 'normal' traffic by using the RS Logix 500 programming software to go online with the PLC to actively monitor the ladder file. By doing this, multiple PLC registers were being polled in real time which would simulate traffic on a PLC network. Since each rule was written to specifically catch each individual vulnerably, the success rate of each rule was 100%. It is also important to note that the other polling traffic did not result in any false positives.

Once the small test program was validated, then portions of each of the real-world ladder logic files were checked for validity. Some of the address range rules are written to cover large blocks of registers, therefore it is not difficult to check those by manual inspection. Other address rules are more scattered. A sampling of the vulnerability rules was selected and checked for accuracy. All signatures in the sampling were deemed valid.

4.4 Implementation

There are several key items regarding implementation of the system described in this paper. First, Snort must be operating with a current rule set. If the ladder logic file is ever modified, then the Ladder Parser program must be executed again with a text export of the new ladder file and those rules must replace the old Snort rule file. If the rule file definitions are not kept up to date, then a significant number of false positives would certainly occur and malicious activity might go unnoticed.

As with any IDS system, placement on the network is of the utmost importance. The computer running Snort must be located where it can analyze all the incoming network traffic to the PLC. This is not always practical where managed switches are used.

The proposed method has a minimal cost in terms of processing power and memory usage. Both of these resources would be impacted as the number of signatures to compare against increase, however testing for this thesis project showed a very minimal increase in memory and CPU utilization when executing Snort with the rule sets generated from the ladder files described in this paper.

CHAPTER V

CONCLUSION

5.1 Summary

Industrial control systems oversee our world's critical infrastructure and manufacturing environments. These systems are under ever increasing threat from a variety of sources. Whether it be state sponsored terror, a weekend hacker just trying to have some fun or a disgruntled employee, the insecure by design hardware that make up these systems are an easy target. The defense-in-depth architecture is the only way to adequately protect against such treats and the IDS plays an important role.

As discussed in this paper, IDS has great potential in the future of ICS security however there are many improvements to be made. Signature-based IDS is only as good as the rules that define it. A security administrator should use everything as his disposal to help define what is to be considered malicious activity. The ladder logic used to program the PLCs offer a world of information regarding the registers that should be accessed over a network and the potential vulnerabilities that are present due to the structure of the program and the inherent traits of the hardware. It is important to leverage the static topology of ICS networks and those programs that define them to enhance the IDS's knowledge of the environment in which it is deployed.

Analyzing the ladder logic and creating these rules by hand would be a futile task due to the magnitude of most PLC programs. This thesis shows that it is possible to automate the process of rule generation for certain types of PLC ladder logic programs. In this paper a Ladder Logic Parser program was created to do just that. Although this thesis only applies to ladder logic and functionality associated with the Allen Bradley SLC 500 and its associated Rockwell Software's Logix500 programming language, the principle can be investigated for other systems.

The testing shows that this method is not only applicable to a controlled test environment, but can also create a significant number of Snort rules that define abnormal behavior using real-world ladder files. Using a smaller test case ladder file, the functionality of this method was proven accurate and a sampling of the larger real-world files were found to be thorough and valid.

5.2 Future Work

There is much work that could be done to build upon this research. The first item to note is that the Snort IDS in this work only had the ability to alert when a packet was detected that would trigger a fault. Further work in this area would work to setup Snort to drop packets that would cause the PLC processor to fault. To do this some work would have to be done to intercept packets, process them and only forward the ones that are deemed valid.

As larger ladder files are parsed and more rules are generated, the increase in the number of rules might have an impact on Snort's ability to perform efficient and timely pattern matching. Further work could be done to improve the rule definitions to make them as efficient as possible to counter the increase in rule quantity.

Another area that has merit for continued investigation is the adaptation of this work to other, newer, ladder logic interfaces and PLC hardware models/manufacturers.

This work specifically looked at one type of aging hardware. Rockwell's newest ladder logic interface, RSLogix 5000, has the same flexibility that allows the graphical ladder file to be copied and pasted into a text format for easy parsing. The addressing is drastically different with a new format because they no longer use the file/word/bit format, but have transitioned to a tag based format that represents addresses as text. Using database files associated with the ladder file, it would still be possible to identify the tags within the ladder file by type to allow the same kind of rule generation to occur. The newer processors may not have the same vulnerabilities that the SLC 500 has, but with experimentation it may be discovered that they have their own inherent set of vulnerabilities that can protected against via automated rule generation.

REFERENCES

[1] Krotofil, Maryna, and Dieter Gollmann. "Industrial control systems security: What is happening?." *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*. IEEE, 2013.

[2] Stouffer, Keith, Joe Falco, and Karen Scarfone. "Guide to industrial control systems (ICS) security." *NIST special publication* 800.82 (2011): 16-16.

[3] GICSP, Ernie Hayden, Michael Assante, and Tim Conway. "An Abbreviated History of Automation & Industrial Controls Systems and Cybersecurity." (2014).

[4] http://www.rockwellautomation.com/en_IN/news/allen-bradley-history-pages.page

[5] Mitsubishi Programming Manual. Manual number: JY992D48301. Rev J, 1999.

[6] https://www.visualstudio.com/en-us/visual-studio-homepage-vs.aspx

[7] http://www.rockwellautomation.com/rockwellsoftware/products/factorytalk-view-me.page

[8] Byres, Eric, and Justin Lowe. "The myths and facts behind cyber security risks for industrial control systems." *Proceedings of the VDE Kongress*. Vol. 116. 2004.

[9] Swanson, Christine A., and William M. Lankford. "Just-in-time manufacturing." *Business Process Management Journal* 4.4 (1998): 333-341.

[10] Gao, Wei, and Thomas H. Morris. "On Cyber Attacks and Signature Based Intrusion Detection for MODBUS Based Industrial Control Systems." *The Journal of Digital Forensics, Security and Law: JDFSL* 9.1 (2014): 37.

[11] Wilhoit, Kyle. "Who's Really Attacking Your ICS Equipment?." *Trend Micro* (2013).

[12] Rockwell Automation Support Center. "29402-TCP/UDP Ports Used by Rockwell Automation Products."

[13] Abrams, Marshall, and Joe Weiss. "Malicious control system cyber security attack case study–Maroochy Water Services, Australia." *McLean, VA: The MITRE Corporation* (2008).

[14] Singer, P. W. "Stuxnet and Its Hidden Lessons on the Ethics of Cyberweapons." *Case W. Res. J. Int'l L.* 47 (2015): 79.

[15] https://ics-cert.us-cert.gov/jsar/JSAR-11-312-01

[16] Bencsáth, Boldizsár, et al. "The cousins of stuxnet: Duqu, flame, and gauss." *Future Internet* 4.4 (2012): 971-1003.

[17] Piggin, Richard. "Industrial systems: cyber-security's new battlefront." *Engineering & Technology* 9.8 (2014): 70-74.

[18] https://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-176-02A

[19] Murphy, Dennis, and I. C. S. Senior. "Lights out! Who's next?." (2016).

[20] Lee, Assante and Conway. *Analysis of Cyber Attack on the Ukrainian Power Grid*. Washington: E-ISAC. 2016.

[21] http://www.snort.org

[22] Cheminod, Manuel, Luca Durante, and Adriano Valenzano. "Review of security issues in industrial networks." *Industrial Informatics, IEEE Transactions on* 9.1 (2013): 277-293.

[23] Harp and Gregory-Brown. *The State of Security in Control Systems Today*. Maryland: SANS Institute. 2015.

[24] Byres, Eric and Eng, P.. "Understanding Deep Packet Inspection for SCADA Security." Tofino Security. 2012. https://scadahacker.com/library/Documents/White_Papers/Tofino%20-%20Understanding%20Deep%20Packet%20Inspection%20for%20SCADA%20Security.pdf

[25] Byres, Eric et al. "Securing EtherNet/IP Control Systems using Deep Packet Inspection Firewall Technology. Tofino Security. 2014.

[26] Hadeli, Hadeli, et al. "Leveraging determinism in industrial control systems for advanced anomaly detection and reliable security configuration." *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*. IEEE, 2009.

[27] Mitchell, Robert, and Ing-Ray Chen. "A survey of intrusion detection techniques for cyber-physical systems." *ACM Computing Surveys (CSUR)* 46.4 (2014): 55.

[28] Liao, Hung-Jen, et al. "Intrusion detection system: A comprehensive review." *Journal of Network and Computer Applications* 36.1 (2013): 16-24.

[29] Roosta, Tanya, et al. "An intrusion detection system for wireless process control systems." *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*. IEEE, 2008.

[30] Valdes, Alfonso, and Steven Cheung. "Communication pattern anomaly detection in process control systems." *Technologies for Homeland Security, 2009. HST'09. IEEE Conference on*. IEEE, 2009.

[31] Valdes, Alfonso, and Stephane Cheung. "Intrusion monitoring in process control systems." *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. IEEE, 2009.

[32] Rrushi, Julian, and Kyoung-Don Kang. "Detecting anomalies in process control networks." *Critical Infrastructure Protection III*. Springer Berlin Heidelberg, 2009. 151-165.

[33] Gao, Wei, et al. "On SCADA control system command and response injection and intrusion detection." *eCrime Researchers Summit (eCrime), 2010*. IEEE, 2010.

[34] Jyothsna, V., VV Rama Prasad, and K. Munivara Prasad. "A review of anomaly based intrusion detection systems." *International Journal of Computer Applications* 28.7 (2011): 26-35.

[35] Wang, Y., Statistical Techniques for Network Security, Modern Statistically-Based Intrusion Detection and Protection. IGI Global. October 2008.

[36] Zamani, Mahdi, and Mahnush Movahedi. "Machine Learning Techniques for Intrusion Detection." *arXiv preprint arXiv:1312.2177* (2013).

[37] Uddin, Mueen, et al. "Signature-based Multi-Layer Distributed Intrusion Detection System using Mobile Agents." *IJ Network Security* 15.2 (2013): 97-105.

[38] Holm, Hannes. "Signature based intrusion detection for zero-day attacks:(not) a closed chapter?." *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014.

[39] De Ocampo, Frances Bernadette C., Trisha Mari L. Del Castillo, and Miguel Alberto N. Gomez. "Automated signature creator for a signature based intrusion detection system with network attack detection capabilities (pancakes)." *International Journal of Cyber-Security and Digital Forensics (IJCSDF)* 2.1 (2013): 25-35.

[40] "An Introduction to Bayes' Theorem." *Bayes' Theorem: Introduction*. Web. May 2016.

[41] Ruggieri, Salvatore. "Efficient C4. 5 [classification algorithm]." *Knowledge and Data Engineering, IEEE Transactions on* 14.2 (2002): 438-444.

[42] LaValle, Steven M., and James J. Kuffner Jr. "Rapidly-exploring random trees: Progress and prospects." (2000).

[43] Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 5-32.

[44] Bishop, Christopher M. "Pattern Recognition." *Machine Learning* (2006).

[45] https://ics-cert.us-cert.gov/advisories/ICSA-10-090-01

[46] McGuiness, Todd. "Defense in Depth." SANS Institute. 2001. https://www.sans.org/reading-room/whitepapers/basics/defense-in-depth-525

[47] Zhu, Bonnie, and Shankar Sastry. "SCADA-specific intrusion detection/prevention systems: a survey and taxonomy." *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*. 2010.

[48] Cheung, Steven, et al. "Using model-based intrusion detection for SCADA networks." *Proceedings of the SCADA security scientific symposium*. Vol. 46. 2007.

[49] Naess, Eivind, et al. "Configurable middleware-level intrusion detection for embedded systems." *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. IEEE, 2005.

[50] Oman, Paul, and Matthew Phillips. "Intrusion detection and event monitoring in SCADA networks." *Critical Infrastructure Protection*. Springer US, 2007. 161-173.

[51] Morris, Thomas, Rayford Vaughn, and Yoginder Dandass. "A retrofit network intrusion detection system for MODBUS RTU and ASCII industrial control systems." *System Science (HICSS), 2012 45th Hawaii International Conference on*. IEEE, 2012.

[52] Caselli, Marco, Emmanuele Zambon, and Frank Kargl. "Sequence-aware intrusion detection in industrial control systems." *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM, 2015.

[53] Falliere, N., Murchu, L.O., and Chien, E.. W32.Stuxnet Dossier, Symantec Tech. Rep. 1.4, 2011.

[54] https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf

[55] Wei Gao, “Cyberthreats, attacks and intrusion detection in supervisory control and data acquisition networks,” Mississippi State University, Starkville, PhD Thesis UMI Number: 3603432, 2013.

[56] Roesch, Martin. "Snort: Lightweight Intrusion Detection for Networks." *LISA*. Vol. 99. No. 1. 1999.