Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations   Student Graduate Works

9-17-2015

A Feasibility Study on the Application of the ScriptGenE Framework as an Anomaly Detection System in Industrial Control Systems

Charito M. Corvin

Follow this and additional works at: https://scholar.afit.edu/etd

Part of the Computer Engineering Commons

**A FEASIBILITY STUDY ON THE
APPLICATION OF THE SCRIPTGENE
FRAMEWORK AS AN ANOMALY
DETECTION SYSTEM
IN INDUSTRIAL CONTROL SYSTEMS**

THESIS

Charito M. Corvin, Captain, USAF

AFIT-ENG-MS-15-S-010

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

A FEASIBILITY STUDY ON THE APPLICATION OF THE SCRIPTGENE

FRAMEWORK AS AN ANOMALY DETECTION SYSTEM

IN INDUSTRIAL CONTROL SYSTEMS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Charito M. Corvin, B.S.E.E

Captain, USAF

September 2015

AFIT-ENG-MS-15-S-010

A FEASIBILITY STUDY ON THE APPLICATION OF THE SCRIPTGENE

FRAMEWORK AS AN ANOMALY DETECTION SYSTEM

IN INDUSTRIAL CONTROL SYSTEMS

THESIS

Charito M. Corvin, B.S.E.E
Captain, USAF

Committee Membership:

Barry E. Mullins, PhD (Chairman)

LTC Mason J. Rice, PhD (Member)

Timothy H. Lacey, PhD (Member)

AFIT-ENG-MS-15-S-010

Abstract

Recent events such as Stuxnet and the Shamoon Aramco incident have brought to light how vulnerable industrial control systems (ICS) are to cyber attacks. ICS and critical infrastructure is ingrained in modern society, including the electric power grid, water treatment facilities, and nuclear energy plants. Malicious attempts to disrupt, destroy and disable such systems can have devastating effects on the way of life in a modern society, including loss of life. The need to implement security controls in the ICS environment is more vital than ever. ICSs were not originally designed with network security in mind. Today, intrusion detection systems (IDSs) are employed to detect attacks that penetrate the ICS network. This research proposes the use of a novel algorithm known as the ScriptGenE framework as an anomaly-based intrusion detection system or anomaly detection system (ADS). The ADS is implemented between an engineering workstation (EWS) and programmable logic controller (PLC) to monitor traffic and alert the operator of anomalous behavior. Two experiments are performed including an Experimental Validation in which a 'Baseline' model of normal network behavior is established. The experiments are designed to test the effectiveness of the ADS when introduced to three types of network traces: *Normal*, *Malicious*, and *Combined*. The *Normal* and *Malicious* network traces are compared with the 'Baseline' model to determine if the ADS will correctly classify normal network behavior with anomalous network behavior in Experiment 1. The *Combined* network trace is used to determine if the ADS is still able to detect anomalies when the training data also contains anomalous behavior in Experiment 2. The ADS achieves true positive rate (TPR) of 0.9011 and false positive rate (FPR) of 0.054 for Experiment 2. In Experiment 1, the ADS achieves a FPR of 0 and true negative rate (TNR)

of 1 and shows that it is a perfect classifier when trained with network traffic that is free of anomalies. Based on Experiment 1 findings, this research demonstrates the viability of using the ScriptGenE framework.

Acknowledgements

I would like to express my sincere gratitude to my faculty advisor, Dr. Mullins, for his guidance and support throughout the course of this thesis effort and throughout my time at Air Force Institute of Technology (AFIT). His insight, support and experience were most certainly appreciated. In addition, I would also like to thank Stephen Dunlap for his guidance in this research effort. Also, my sincere gratitude to LTC Mason Rice and Dr. Timothy Lacey for their assistance and support in this effort.

Finally, I would like to thank my husband for encouraging me and sustaining me throughout my time at AFIT. I could not have made it this far without him.

Charito M. Corvin

Table of Contents

List of Figures

List of Tables

List of Acronyms

AFIT   Air Force Institute of Technology

ADS   anomaly detection system

AUC   area under the curve

CLAD   clustering for anomaly detection

DARPA   Defense Advanced Research Projects Agency

DCS   distributed control systems

DoS   denial of service

EtherNet/IP   EtherNet Industrial Protocol

EWS   engineering workstation

FN   false negative

FP   false positive

FNR   false negative rate

FPR   false positive rate

FTP   file transfer protocol

FPR   false positive rate

GUI   graphical user interface

HIDS   host-based intrusion detection systems

HMI   human machine interface

HTTP   Hypertext Transfer Protocol

ICS   industrial control systems

CERT   Computer Emergency Response Team

IDS   intrusion detection system

IP   internet protocol

IT   information technology

LAN   local area network

LERAD   learning rules for anomaly detection

MR   mutation rate

MTU   master terminal unit

NIDS   network intrusion detection systems

NIST   National Institute of Standards and Technology

p-tree   protocol tree

PI   protocol informatics

PLC   programmable logic controller

ROC   receiving operating characteristic

SCADA   supervisory control and data acquisition

SUT   system under test

TCP   transmission control protocol

TN   true negative

TP   true positive

TNR   true negative rate

TPR   true positive rate

UDP   user datagram protocol

UPGMA unweighted pair group method with arithmetic mean

VM   virtual machine

WAN   wide area network

A FEASIBILITY STUDY ON THE APPLICATION OF THE SCRIPTGENE
FRAMEWORK AS AN ANOMALY DETECTION SYSTEM
IN INDUSTRIAL CONTROL SYSTEMS

I. Introduction

1.1 Background

Researchers, asset owners, federal agencies, and hackers all have a vested interest
in developing methods to secure supervisory control and data acquisition (SCADA)
systems as these systems infiltrate every aspect of modern society. Today, because of
increased connectivity to the Internet, cyber attacks can reach into homes and vehi-
cles and cause destructive, if not fatal, effects [Sch14]. Even more alarming are the
effects that attackers can inflict on larger populations by targeting ICSs which include
SCADA systems. It is cited that cyber attacks that destroy or manipulate equipment
are far more prevalent than widely believed [Men15]. In 2014, 245 incidents were re-
ported to the ICS-Computer Emergency Response Team (CERT) [DHS14]. Fourteen
of the sixteen critical infrastructure sectors reported their incidents to ICS-CERT,
including the nuclear, healthcare, critical manufacturing and energy sectors. Of the
reported incidents, 59% were reported from the critical manufacturing and energy
sectors [DHS14]. Famous examples, such as Stuxnet, remain in the forefront of secu-
rity professionals' and asset owners' minds as they are motivated to provide novel and
reliable protection mechanisms that address the challenges of the ICS environment.

1.2 Motivation

Today, in ICS/SCADA "communication is carried through a variety of media: Ethernet, wireless, shared lease lines, and even the Internet" [RGH07]. Denial of service (DoS) attacks exploit the availability and lack of security often found within an ICS environment. An example of the impact an attack could have is "intentional removal of a system's electrical power as a physical DoS attack" [CKBR06]. According to the National Institute of Standards and Technology (NIST) guide's recommendations, network-based intrusion detection system (IDS), are most effective when deployed on networks and "computers that use general-purpose operating systems (OSs) or applications such as human machine interfaces, SCADA servers, and engineering workstations" [SFS11]. The guide also adds that IDSs "can greatly enhance the security management team's ability to detect attacks entering or leaving the system, thereby improving security" [SFS11]. However, operators and engineers must be able to recognize attacks in addition to organized patterns. The need for an anomaly-based IDS, which distinguishes anomalous network behavior from normal network behavior in an ICS environment is vital.

1.3 Problem Statement

The goal of this research is to assess the viability of a novel framework, known as ScriptGenE, as an anomaly-based IDS. ScriptGenE was originally created to provide automatic configuration of PLC emulators [War15]. Because of the intrinsic clustering algorithms in the ScriptGenE framework, its protocol tree building capabilities, and previous research results, the framework presented an opportunity to be implemented as an IDS that can detect anomalies on an ICS network. It is hypothesized that the ScriptGenE framework can be implemented as an anomaly-based IDS in an ICS environment.

1.4 Approach

This research studies the implementation of an automatic PLC emulator as an anomaly-based IDS or ADS. The ADS is implemented between a PLC and an EWS that are connected via Ethernet to a spanning port switch. Two network traces are created for experimentation using a real PLC: one with simulated malicious network traffic and one with attack-free network traffic. A third trace is produced by merging the malicious network traffic with the attack-free network traffic. The network traces are used in three experiments which are conducted for this research and are labeled Experimental Validation, Experiment 1, and Experiment 2. The 10-fold cross-validation method is used in the Experimental Validation and Experiment 2 to ensure an accurate estimate of the performance of the ScriptGenE framework as an ADS. The result of the Experimental Validation is a 'Baseline' protocol tree (p-tree) and is used to represent normal network behavior between an EWS and PLC. The 10-fold cross validation technique follows machine learning evaluation methods as the ScriptGenE framework closely resembles other machine learning-based ADSs in its ability to build a baseline model of normal behavior for comparison against incoming traffic for anomalous, potentially malicious, activity [Dom12, ZZLJ08, MJ14, YJ13, CMA03, YUH06].

In Experiment 1, network traces are input into the ScriptGenE framework, and p-trees are created using clustering algorithms. The ScriptGenE replay script contains a unique backtracking algorithm that allows for session looping and is used to serve the 'Baseline' p-tree. The two network traces that are generated are then used to create p-trees to be compared with the 'Baseline' p-tree, which models normal network behavior. Output logs from the replay script capture anomalies between the p-trees. Additionally, mutation rates are measured between the 'Baseline' p-tree and experimental network traces mentioned earlier to determine quantitatively if differences

between the p-trees do exist. Experiment 2 addresses a scenario in which an attacker remotely inserts malicious traffic and attempts to train the ADS with the intent to carry out an attack later. The third network trace containing merged malicious and attack-free network traffic is used for this experiment.

A TPR of 0.9 and FPR of 0.1 are selected as goals for this research and are used to demonstrate effectiveness of the ScriptGenE framework as an ADS.

1.5 Assumptions and Limitations

1.5.1 Scope.

Experiments are conducted in an isolated laboratory environment. Real ICS environments are large, complex, and application-specific. This research uses a lab environment which is reduced to one PLC and two dedicated workstations, representing an EWS and the ADS device. The ScriptGenE framework is tested within this small, isolated environment to evaluate performance and the feasibility of the framework as an ADS.

1.5.2 Limitations with ICS network traffic data samples.

Locating real ICS network traffic samples is difficult due to the security implications of analyzing real network traffic. It is in the best interest of asset owners not to release real samples even for research purposes. Therefore, network traffic is generated between the PLC and the EWS for analysis and testing. Additionally, simulating real network traffic between an EWS and PLC also narrows and simplifies the data collected.

1.5.3 ScriptGenE Limitations.

In addition to the documented assumptions and limitations of the current version of the ScriptGenE framework, the ScriptGenE framework is not written to handle more than one transmission control protocol (TCP) session. [LDM06] identifies dependencies that can be interleaved among more than one TCP session as inter-protocol dependencies as is the case with the file transfer protocol (FTP) protocol, for instance. The FTP `recv` command is seen on the control connection to cause traffic to be sent on the data connection [LDM06]. The lack of inter-protocol dependency handling at this moment limits the network traffic to be generated between one field device and one workstation for the ScriptGenE framework to perform [War15]. This research focuses specifically on EtherNet Industrial Protocol (EtherNet/IP) and intrusion detection between an EWS and PLC.

1.6 Thesis Overview

The rest of this document is organized in the following way. Chapter II provides background and related research on intrusion detection and anomaly detection in the ICS environment. Chapter III presents the methodology of this research. Chapter IV covers results and analysis. Finally, Chapter V presents the research conclusions and future work.

II. Background and Related Research

2.1 Overview

This chapter presents an overview of background information on ICS and SCADA systems and related research in IDSs implemented in ICS and SCADA networks. Section 2.2.1 provides an overview of ICSs and SCADA systems. Section 2.2.2 reviews operations and control components. Section 2.2.3 reviews threats to SCADA systems and provides background on previous attacks as well as current vulnerabilities and risk factors. Section 2.2.4 examines IDSs as security instruments, providing a brief analysis of the different types of IDSs. Section 2.3 describes related research in the area of anomaly detection systems as applied to ICS environments. Finally, Section 2.3.2 is provided for an understanding of the ScriptGenE framework internals.

2.2 Background

2.2.1 Industrial Control Systems Overview.

Industrial control systems is a term that comprises different types of control systems, such as SCADA systems and distributed control systems (DCS) as well as control system configurations, such as PLCs. These control systems are often implemented in industrial sectors and critical infrastructures such as electrical, water, oil, natural gas, manufacturing, and pharmaceutical industries, to name a few. The differences between SCADA and DCS systems are often blurred in their implementations. By conventional definition, SCADA systems control geographically-separated assets with a centralized control center that manages data acquisition [SFS11]. On the other hand, DCS are usually confined to a local area and are not as geographically dispersed as SCADA systems. Whereas communications between field devices in a DCS can occur over a local area network, SCADA systems are designed to handle

long-distance communications [SFS11]. Throughout the remainder of this document, SCADA systems and DCSs are referred to collectively as ICSs. ICSs are critical to the operation of critical infrastructure in the US as they provide the framework of daily life for nearly 319 million citizens [Cen].

Figure 1 depicts a general layout of an ICS, which consists of three levels [SFS11]:

- Management level which consists of the control center

- Communications transport level which consists of the wide area network (WAN)

- Field level which consists of field devices such as PLCs
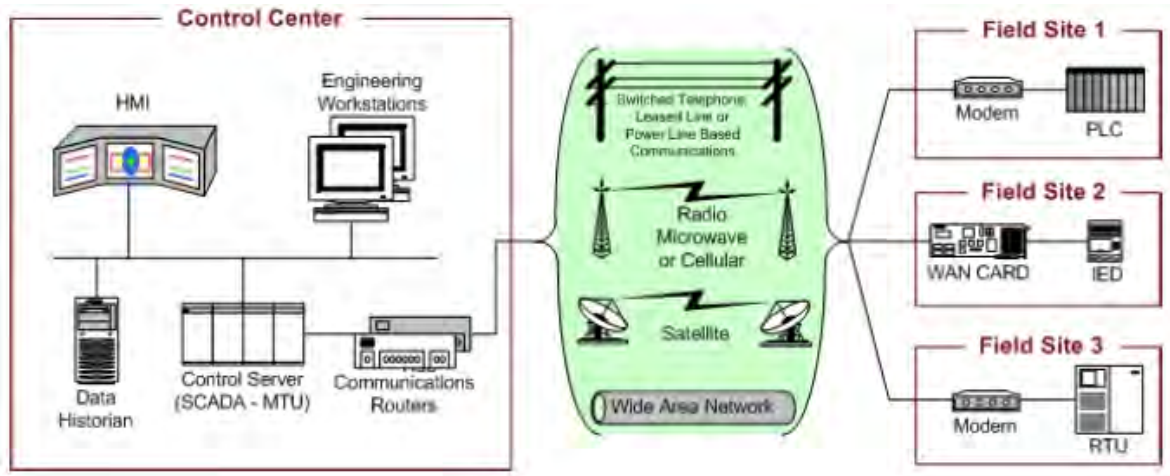
Figure 1. ICS System General Layout [SFS11]

The control center contains the ICS server or master terminal unit (MTU), human machine interface (HMI), EWS, data historian and communications routers. All of these devices are connected by a local area network (LAN) and are used to collect and log information taken from the field sites to be presented to a human operator [Dun13]. System operators are also vital to this level in a ICS as they use the EWS or HMI to remotely control and ensure proper operation of field devices. In addition, the control center handles centralized alarming, trend analyses, and system reporting.

The operator is able to make decisions for the entire process based on the collected and analyzed data [SFS11].

The communication transport level consists of a WAN that connects field devices to the control center. There are various techniques by which data is transported from the field device to the control center HMI or EWS—cable, fiber, radio frequency and/or satellite [SFS11].

Field devices include PLCs and remote terminal units (RTUs) which control local processes and monitoring of sensors. PLCs and RTUs are embedded devices configured by operators to handle specific functions such as arithmetic, data processing, I/O control, and timing. Programming a PLC is accomplished via a user interface located on the EWS, and data is stored in the data historian; these components are also accessible via a LAN [SFS11].

2.2.2 ICS Control Components and Operations.

This research focuses primarily on two control components: the PLC and the EWS. As previously mentioned, PLCs control complex processes at the field level of an ICS. They are configurable, economical, versatile, and typically provide an interface between the cyber-physical components (e.g., sensors) and the information technology (IT) components (e.g., the HMI) [Dun13, SFS11]. The EWS and HMI consist of both software and hardware that allows a human operator to monitor or change processes and control settings [SFS11]. Figure 2 depicts a block diagram of the process that takes place in a SCADA system with all the control components. The EWS and/or HMI give the operator or engineer control over the state of operations and allows the operator to manually override any control process in the event of an emergency. Additionally, the EWS and/or HMI provide the engineer or operator with information on process data which can also be sent to administrators, managers or

business partners for further analysis and action [SFS11]. These two components are critical in ICS operations.
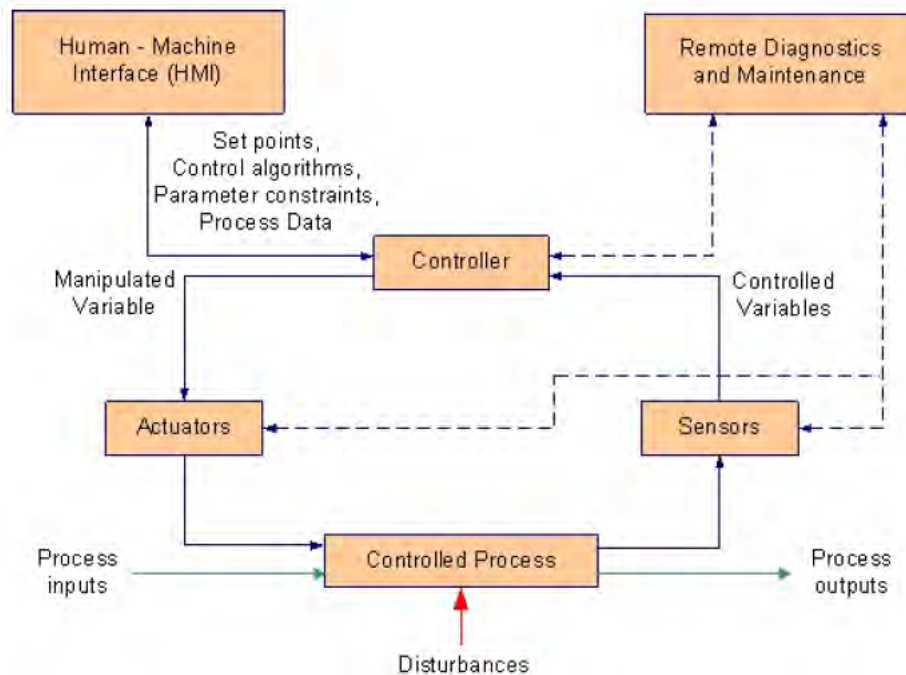
Figure 2. ICS Process Block Diagram [SFS11]

2.2.3   Threats to ICS.

2.2.3.1   Previous ICS Attacks.

In an address to the Brookings Institute, the Chairman of the Joint Chiefs of Staff, GEN Martin Dempsey stated that the first targets in a cyber attack will be civilian infrastructure and businesses. Moreover, GEN Dempsey also added that detected intrusions to US critical infrastructure have increased 17 fold since he assumed the chairman position in 2011 [Dem13]. According to an annual US ICS-CERT report, 245 incidents were reported in 2014; 32% of those incidents were from the energy sector, while 27% originated from the critical manufacturing sector [DHS14].

Stuxnet is perhaps the most well-known and widely referenced attack on ICS. According to the W32.Stuxnet dossier published by Symantec, Stuxnet was deployed to

9

covertly sabotage PLCs by reprogramming them to operate outside of the intended boundaries of the device as used within a nuclear centrifuge system [FMC11]. Stuxnet consisted of a wide variety of attack components to maximize the chance of success for the attacking party. Specifically, Stuxnet contained the first ever PLC rootkit, command and control interface, four zero-day exploits, a Windows rootkit, antivirus evasion techniques, complex process injection and hooking code, network infection routines, and peer-to-peer updates [FMC11]. In September 2010, it was reported that over 100,000 hosts had been infected by Stuxnet around the world. The proliferation of the Stuxnet virus and the success with which it was able to manipulate the EWS, PLCs, and human operators illustrate the vulnerabilities that exist in three components of an ICS.

The Shamoon-Aramco incident provides another example of a destructive attack on ICSs. In August 2012 the Saudi Arabian Oil Company, also known as Saudi Aramco, suffered huge data loss that resulted in the disruption of daily operations for nearly two weeks [BTR13]. As it came to be known later, the Shamoon virus was reported to have infected over 30,000 Windows-based computers on the Saudi Aramco company network. The main function of the software was to haphazardly delete data from the company computer hard drives [BTR13]. Although no physical damage occurred, analysis of the attack illuminates the target selection criteria from the attacking party point of view. At the time, Saudi Aramco held 10% of the global supply and was the largest oil producer in the world [BTR13]. Their holdings in the oil market could have been a possible motive of the attacker. In addition, Saudi Aramco sales topped over $200 billion annually, and the attack affected the business processes of the company as well as other oil and gas firms such as Exxon-Mobil [BTR13]. Former Defense Secretary Leon Panetta stated the Shamoon virus is "very sophisticated" and that the US is existing in a "pre-9/11 moment" in which

the US remains unprepared as attackers continue to plot and carry out cyber attacks such as the Shamoon incident [Dem13, BTR13].

2.2.3.2 Vulnerabilities in ICS.

The attacks mentioned in the previous section highlight well known vulnerabilities that exist in ICS. The increasing interconnectedness of ICS devices to the Internet allows attackers to exploit network vulnerabilities typically found in a traditional IT environment. Unlike the traditional IT community, ICS operators place safety and availability above confidentiality and integrity [SFS11]. Because business and operations decisions are centered around safety and availability, attackers can craft malicious software that exploits network vulnerabilities that would normally be patched in a traditional IT environment. The National Institute of Standards and Technology (NIST) cites several network vulnerabilities ranging from configuration vulnerabilities to network hardware, perimeter, communication and wireless connections [SFS11]. Poorly configured security equipment, for instance the practice of using default configurations, allows attackers to exploit open ports or network services. In the case of password management, NIST cites that passwords were permitted to be transmitted in plain text leaving them susceptible to interception via a man-in-the-middle attack [SFS11].

In addition to network device configuration vulnerabilities, NIST documented that no routine security monitoring occurred on ICS networks. This vulnerability leaves an ICS network susceptible to intrusions that could go unnoticed and cause damage to equipment or disruption to services and operations, similar to the Shamoon-Aramco incident [SFS11, BTR13]. No security monitoring also increases vulnerabilities in communication paths accessible externally to an ICS network. Unknown and unverified connections into an ICS network could leave backdoors open for attacks.

Furthermore, once inside the network, attackers could manipulate industrial control protocols without raising suspicion as there is a lack of integrity checking in communications protocols [SFS11].

2.2.3.3 Risk Factors.

The ICS environment presents several challenges in addressing the previously mentioned vulnerabilities that would not normally be an issue in a traditional IT environment. In traditional IT, when information about a vulnerability is released, a software patch is developed and administrators immediately apply those patches to remove the vulnerability from the system and enterprise. An ICS environment, on the other hand, prioritizes system availability and saftey over data integrity. Patching ICS equipment requires taking a control system offline, which requires weeks of planning and the possibility of not providing redundant services in the event the system could not be brought back online in the allotted time [SFS11, Dun13].

First generation, monolithic ICS and SCADA systems were isolated and maintained on a separate network [Sha06]. Control systems are less isolated and separated from the corporate IT enterprise network. Figure 3 depicts a two-layer topology in which the control/field network is connected to the corporate network via a router [SFS11]. Today, corporate networks are connected to the Internet making communication and management of day-to-day business more convenient and accessible. As a result, control systems may be exposed to the Internet via their connection to the corporate networks. This change is due in part to paradigm shifts in operational and information management practices. More recently, decision makers in organizations require access to data to make more informed decisions on the manufacturing and distribution processes [SFS11].
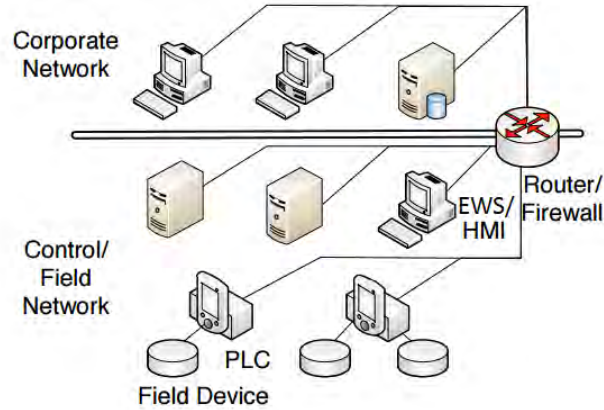
Figure 3. Two Layer Network Topology

Although increased connectivity makes it convenient for asset owners, decision makers, engineers and operators, it has increased the risk of exposing field devices and other critical system components to the Internet and malicious attacks. Field devices have been recorded and can be found on the well-known search engine, SHODAN. SHODAN is an acronym for Sentient Hyper-Optimized Data Access Network and is the first search engine for Internet-connected devices. Project SHINE, which is an acronym for SHODAN Intelligence Extraction, was launch between April 2012 and January 2014. The intent of the project was to find ICS field devices that were connected to the Internet. The 2014 Project SHINE report revealed that 2384 field devices in the US using port 44818 (EtherNet/IP) are accessible via the Internet [RB15]. This accessibility presents a real problem as ICS field devices connected to the Internet are vulnerable to attack.

Shortly after the Project SHINE report was published, another project known as Project RUGGEDTRAX was conducted to provide substantiation that Internet-facing ICS devices are indeed vulnerable to attack. ICS equipment was purchased on eBay, configured with minimal security controls and connected directly to the Internet. Within two hours of being connected to the Internet an attack was de-

tected. Within two days, the ICS device appeared on the SHODAN search engine. The project was conducted over the period of three months. At the conclusion of the project, over 140,000 attacks were detected and recorded from 651 IP addresses [RB15].

In addition to the patching issue, the age of ICS equipment presents another challenge for securing ICS networks. Whereas traditional IT equipment is replaced every three to five years, ICS equipment can remain in operation for up to thirty years [SFS11]. Furthermore, newer equipment needed to replace legacy systems may not be available or even manufactured, and engineers are instructed by vendors to ensure that systems contain the current firmware updates instead of equipment replacement. According to NIST, the mixture of newer hardware and software with legacy systems can make it difficult and/or infeasible to apply their recommended security controls [SFS11].

2.2.4 Intrusion Detection Systems Overview.

To address the vulnerabilities and risk factors listed in previous sections, NIST recommends a defense-in-depth architecture strategy that includes implementing firewalls, demilitarized zones, training programs for personnel, incident response mechanisms and IDS [SFS11]. NIST defines an IDS as the process of monitoring events in a computer system or network and the analysis of such events looking for intrusion traces [GUZ11, SFS11]. An IDS can also be defined as a system that can detect malicious or inappropriate actions of a system within a computer or within a whole network [DOS06]. The intrusion detection process can be divided into three components [GUZ11, JY13]:

- Information sources

- Data analysis strategy

- Threat assessment and response

Information can originate from several sources; [GUZ11] classifies three: information obtained from a host, obtained from monitoring a network, and data obtained from the execution of applications. From this classification of information origination, IDSs can be separated into two main groups: host-based intrusion detection systems (HIDS) and network intrusion detection systems (NIDS). HIDS monitor and analyze information relating only to the host, such as application logs. NIDS analyze activities on a network and are more commonly used in ICS networks [GUZ11]. A network event is a collection of traffic data, such as internet protocol (IP) or TCP packets [JY13]. Since NIDS are more frequently used in ICS environments, consideration of the data analysis strategy further classifies IDSs into two main categories: misuse- or signature-based intrusion detection and anomaly-based intrusion detection [GUZ11]. Threat assessment and response involves the human operator as part

of the process. According to [Zan04], the IDS process loop closes on a human, which is an essential part of the system. When an alert is generated in the HMI or EWS, it is up to the human operator to decide on the best course of action for the control system.

2.2.4.1 Signature-based IDS.

A signature-based, also known as misuse detection, IDS monitors and analyzes activities on a network and compares those events with signatures of attacks stored in a database [GUZ11]. Signature detection methods are widely used because they have high accuracy rates. If an event matches a signature, an alert is generated. However, as new attacks are discovered every day, rule sets must be manually updated. Therefore, the signature-based IDS is only as effective as the current rule sets which include the most current attack signatures [JY13].

2.2.4.2 Anomaly-based IDS.

Anomaly detection systems depend on the activity on the network to build a model of normal behavior with which to compare anomalous behavior [GUZ11]. Furthermore, anomaly detection is used in control components of critical infrastructure to detect tampering of communications among those components [Hgv06]. Anomaly detection systems look for deviations from normal behavior and can include detection of novel attacks [CMA03, Hgv06]. This characteristic is an advantage of ADSs in ICS environments. Anomaly detection systems have the capability of classifying deviations from learned normal behavior as intrusions in a system [GUZ11].

One of the drawbacks in this system is the inability to discern intent, meaning the system is unable to determine whether a deviation from normal behavior is malicious or not. The system is designed to trigger an alert simply due to a deviation [CMA03].

This problem presents itself in a high FPR [GUZ11]. An asset owner defines the acceptable threshold for false positives. An ADS with a high FPR could potentially serve no purpose in an ICS network [CMA03, GUZ11]. Another cited problem is a lack of clarity in the detection process. The lack of clarity works to the advantage of the attacker as he could work slowly within the network to develop a new model of acceptable behavior so that the IDS does not register his malicious activity as a deviation [GUZ11].

2.2.4.3 Shadow Honeypot.

The Quickdraw SCADA IDS is one of many ongoing projects at Digital Bond to encourage the protection of critical systems [?]. Within this project are nearly 150 attack signatures written for common ICS protocols, such as ModBus TCP, Ether-Net/IP, and DNP3, to be implemented in a misuse- or signature-based IDS. Although these signatures are being shared throughout the ICS community, signature-based IDSs require routine updating to ensure the IDS is able to detect current attacks. An ADS is useful in separating anomalous behavior from normal behavior, however it does not discriminate malicious, anomalous behavior from benign, anomalous behavior. [SS12] and [ASA+05] proposed a novel concept known as a shadow honeypot. The shadow honeypot combines features of an ADS with the honeypot. Anomaly detectors monitor all traffic to a protected network [ASA+05]. Once the ADS detects anomalous behavior, traffic is routed to a honeypot where it is investigated and replayed for the attacker. It is recommended a IDS or honeypot be paired with any ADS to form a hybrid IDS, however, this is beyond the scope of this research.

2.3 Related Research

There is much research in the area of anomaly detection in ICS networks. [VCD$^+$07] claims that anomaly detection is better for ICS environments than signature-based intrusion detection. ICS networks are more predictable than traditional corporate enterprise networks as they must operate in a predictable manner continuously performing the same operations [GUZ11]. The predictable nature of ICS networks makes anomaly detection a better fit as an IDS; normal operations do not and cannot vary daily therefore the learned behavior will follow a fairly strict pattern and deviations from that pattern can be logged for an operator to analyze.

2.3.1 Machine Learning Methods Applied to Intrusion Detection.

2.3.1.1 Learning Rules for Anomaly Detection (LERAD).

Machine learning techniques have been applied to anomaly detection in ICS networks. In their work, [CMA03] developed and examined two methods that build models of ICS network traffic from past behavior. LERAD is an algorithm that characterizes normal behavior on the network then builds a model to compare using probability to estimate the likelihood of an instance under consideration is anomalous. learning rules for anomaly detection (LERAD) is designed to be an efficient algorithm that uses a minimal set of rules to concisely characterize training data. Afterward, the probability that an instance occurs outside of this rule set is estimated. Finally, an anomaly score based on an instance, x, is calculated by finding the p-value of a rule in the rule set. A small p-value indicates a low likelihood of the instance, x, being anomalous.

The researchers used the 1999 Defense Advanced Research Projects Agency (DARPA) intrusion detection dataset and assumed the training data was free of attacks, thereby enabling the algorithm to develop a model based on normal behavior. Based on their

experiments, resultant rule sets consisted of 50 to 75 rules, and the algorithm detected 117 attacks out of 201 with at most 10 false positives [CMA03]. They concluded that LERAD was successful in finding highly predictive normal patterns and was able to detect 58% of attacks not detected by the original participants who attempted to identify intrusion in the DARPA dataset [CMA03].

Machine learning when applied to anomaly detection shows potential in providing a layer of security in an ICS environment. The main drawback to machine learning as a method of anomaly detection is a high rate of false positives due to the difficulty in obtaining attack-free data [PS10]. Although this presents a challenge many researchers must overcome, it does not take away from the overall advantage anomaly detection has over signature detection: the potential ability of the system to detect novel attacks as they are occurring on the network [GUZ11, MJ14, YJ13, YUH06, ZS10].

2.3.1.2 Clustering Algorithms.

In addition to LERAD, [CMA03] developed another algorithm based on clustering and aptly named the algorithm, clustering for anomaly detection (CLAD). The clustering approach allowed for the identification of "outliers" or data points outside of clusters of selected network features, to be classified as anomalous. Though the algorithm is related to k-NN (Nearest Neighbor), the key differences in this algorithm are that clusters are allowed to overlap and clusters have a fixed width [CMA03].

CLAD operates in two phases. The first phase creates clusters of fixed width, W, while the second phase assigns data points to the created clusters. During this phase, if a data point is farther away than the fixed width, W, from a cluster, then the data point becomes the center of a new cluster. If the data point is not farther away than W, then the data point is clustered with the existing cluster. Determining

whether a cluster is an outlier requires analysis using two properties of a cluster: the density of the cluster and the distance from other clusters [CMA03]. Density is calculated by counting the number of data points within a cluster. Inter-cluster distance is calculated by the Euclidean distance function. After these calculations are performed, a determination is made as to whether the cluster is distant or "sparse." A "distant" cluster, considered to be a global outlier, is more than one standard deviation away from the average of inter-cluster distances. A cluster is considered "sparse," a local outlier, if it is more than one *median absolute deviation* smaller than the median number of data points in the cluster, labeled as "Count." Alerts signifying anomalies are generated for clusters that are sparse or dense and distant [CMA03].

The algorithm was implemented by learning a model for each TCP port and 10 application protocols, including FTP, Hypertext Transfer Protocol (HTTP) and POP3 to name a few. The researchers found that density was not as reliable as inter-cluster distance in determining an anomaly score. Their model learned 11 application protocols from the DARPA dataset and detected 76 attacks. Although the baseline of attacks detected from this dataset is 85, it must be noted that 85 detected attacks were achieved by a signature-based detection system. The anomaly detector, CLAD, on the other hand, detected 76 attacks [CMA03]. Although it detected nine less attacks than the signature-based IDS, CLAD detected the attacks with no a priori knowledge of the attacks, which is another appealing feature of anomaly detection systems.

2.3.1.3 Pattern Matching.

Pattern matching is another method used in anomaly detection. [YUH06] examined network traffic by creating profiles using specific feature vectors, then classifying the vectors based on temporal variables (e.g., time of day, day of week). The IDS

predicts correct behavior using predefined features that are meant to represent normal network behavior using an auto-associative kernel regression (AAKR) model. Finally, a sequential probability ratio test (SPRT) is applied to the residuals in the model to determine if the residuals are generated from normal behavior or an anomalous distribution. If the new traffic data fails to fit within the profiles, an alert is generated [GUZ11, YUH06, ZS10].

2.3.2 ScriptGenE Framework.

The ScriptGenE framework was originally designed for automatically configuring PLC emulation. The ScriptGenE framework consists of two Python scripts: ScriptGenE.py and ScriptGenEreplay.py; the former generates and manipulates protocol trees (p-trees) and the latter replays the p-trees as an emulator. ScriptGenE is based on ScriptGen (an automated script generation tool for honeyd) and protocol informatics (PI), which allows for protocol stream analysis by using bioinformatics algorithms [LMD05, War15]. PI is used in the ScriptGenE framework and provided alignment functions, such as Smith-Waterman, Needleman-Wunsch and tree creation via unweighted pair group method with arithmetic mean (UPGMA) clustering.

ScriptGen extends PI by using Region Analysis. Regions are bytes with the same type, similar mutation rates, the same "kind" of data, and the same gap presence, which are all calculated and displayed when bytes in messages are aligned [LMD05]. Regions are used to determine whether intra-protocol dependencies, or links, between client and server messages exist within a session [LDM06, War15].

The ScriptGenE framework begins with the ScriptGenE script. Packet capture (Pcap) files are input to ScriptGenE.py with build parameters specified. After a series of consolidations and clustering operations, SciptGenE.py generates an initial p-tree that represents valid TCP connections based on observed traffic. Next, ScriptGenE.py

identifies intra-protocol dependencies and the resulting p-trees are exported to be used in the replay script for emulation [War15].

The replay script, ScriptGenEreplay.py, uses a set of build options to define the server to be emulated. ScriptGenEreplay.py loads a p-tree, and the user specifies the server IP and port. To test the replay functionality, test client makes a connection with the server. The server replays the loaded p-tree based on messages that are sent from the test client. Specifically, the TCP connection that is established with the server provides data in a stream of bytes from the client, and every "chunk" of data received is checked for a match in the serving p-tree [War15]. If a match is discovered, the corresponding stored server message is retrieved and sent back to the client if it is data. If no match is found in the p-tree, or the corresponding server response contains no data, then an unknown transition is handled. The replay script offers a novel algorithm known as *backtracking*, that allows the emulator to backtrack to a different state with a matching transition to send valid data back to the client [War15]. The goal of the backtracking algorithm is to find the earliest edge in the p-tree that matches the client request and then return the corresponding server response and data. The backtracking algorithm enables session looping, and provides flexible emulation [War15].

Implementing the ScriptGenE framework as an ADS requires a combination of approaches to test and evaluate the scripts. As an ADS, ScriptGenE most resembles pattern matching ADSs. Recall that pattern matching analyzes deviations from normal behavior by establishing patterns in normal data sets and identifying outliers as anomalous [YUH06]. Additionally, ScriptGenE utilizes several clustering algorithms to group TCP message sequences and build protocol trees [War15]. In anomaly detection, clustering algorithms characterize messages, align bytes, and select features to classify byte streams from messages, similar to the algorithms within ScriptGenE.

Finally, to evaluate the effectiveness of the ADS, a machine learning testing technique known as cross-fold validation is used to ensure that the entire data set is utilized.

2.4 Chapter Summary

ICS is an important part of daily life and is ingrained in modern societies. The United States has 16 critical infrastructure sectors that are as susceptible to cyber attack as any traditional corporate IT enterprise. This section provides background on the ICS environment as well as threats, vulnerabilities and risk factors to the ICS. This section also examines IDSs and specifically provides information on current research on anomaly detection systems. Machine learning and pattern matching are just two methods in the research community that are being investigated for detecting anomalous behavior in the ICS environment. An overview of the ScriptGenE framework is provided to provide background on the framework as a set up to its application as an anomaly detection system.

III. Framework Design

3.1 Overview

This research focuses on the feasibility of implementing an ADS within an ICS environment based on the ScriptGenE framework. The goal in testing and implementation are both in line with current research methods in anomaly detection and is described in the following sections.

3.2 Approach

Python-based ScriptGenE is implemented as an ADS. Much of its original functionality is preserved to prove its versatility as an ADS. The ADS is implemented in a simulated ICS environment designed to specifically monitor traffic between an EWS/HMI and PLC. The link between the EWS and a PLC is chosen because of the potential for a successful man-in-the-middle attack to occur between the two devices and manipulation of either the controller, HMI, or EWS [SFS11]. If an adversary can take control of the controller or field device, the adversary could send modified network traffic to cause "undesirable events" to occur unbeknownst to the human operator [SFS11]. The rest of this section describes the components of the intrusion detection process and its applicability to this research. Additionally, a brief description of the experimental approach is introduced.

3.2.1 Intrusion Detection Process Components.

This section elaborates on the components of a standard intrusion detection process as the method with which the ScriptGenE framework is tested and evaluated. A standard intrusion detection process is listed below [JY13]:

- Information Sources – *Where does the information or data originate?*

- Data Acquisition Tools – *What tools are needed to acquire the necessary data?*

- Data Pre-processing – *What component of the ScriptGenE framework will be used to pre-process the data?*

- Intrusion Detection – *What component of the ScriptGenE framework will be used for intrusion detection?*

- Data Analysis – *How will the intrusion detection method be evaluated? What are the performance metrics?*

- Threat Analysis and Response – *How will threats be addressed?*

The components of the intrusion detection process are collectively used an an approach for evaluating the viability of the ScriptGenE framework as an ADS. For this research, each component must be addressed in order to proceed utilizing the ScriptGenE framework as an ADS. Figure 4 depicts the proposed process flow for implementing the ScriptGenE framework ADS and serves as a reference visual in addressing the standard IDS process components listed above. The following sections address each component of the intrusion detection process.

3.2.1.1 Information Sources.

For this research, information or data originates from an EWS and a PLC. Information or data is defined as the network traffic that is generated between the EWS and the PLC and is the workload for the system under test (SUT) described in more detail in Section 3.3. Network traffic between the EWS and PLC is narrowly generated to model normal behavior found in a real ICS system. The recorded
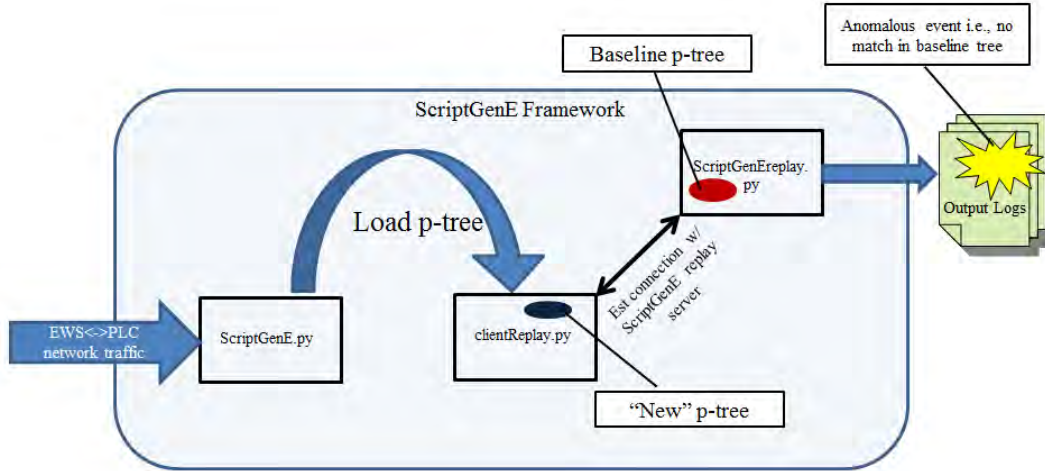
Figure 4. Proposed ScriptGenE ADS Process

network traffic is saved as a Pcap file. Referring to Figure 4, this is portrayed as "EWS ↔ PLC network traffic" in the ScriptGenE ADS process.

3.2.1.2 Data Acquisition Tools.

Data acquisition tools are used to capture events from network-based information sources [JY13]. An experimental script was developed for this research to initiate the packet capture between an EWS and PLC. The network analyzer `tcpdump` is used in the experimental script to collect network traffic generated between the EWS and PLC via a spanning port switch. Section 3.7 provides a detailed description of the hardware used to collect the data.

3.2.1.3 Data Pre-processing.

Data pre-processing refers to data cleaning/transformation techniques used to support analysis of the IDS [JY13]. Referring to Figure 4, the data pre-processing component is depicted by the ScriptGenE.py block within the ADS. The Script-GenE.py script is used for data pre-processing as it transforms a packet capture of the generated network traffic between the EWS and PLC to a p-tree to be used for

26

intrusion detection. A Pcap of the captured network traffic is input to ScriptGenE.py.

3.2.1.4 Intrusion Detection.

For this research, the intrusion detection component of the ADS consists of using a clientReplay.py script and the ScriptGenEreplay.py script, as shown in Figure 4. 'New' p-tree refers to new network traffic introduced to the system to be verified by the 'Baseline' p-tree loaded into the ScriptGenEreplay.py script. The 'Baseline' p-tree represents normal, acceptable network behavior between the EWS and PLC. An anomalous event is defined as one to many instances where there is no match found between the 'new' p-tree and the 'Baseline' p-tree. Output logs generated by the ScriptGenEreplay.py script are to represent network logs that would be reported to a human operator at the EWS.

3.2.1.5 Data Analysis.

Data analysis for this research is the performance evaluation of the ADS, which includes performance metrics of the SUT, depicted in Figure 5. Performance metrics are described in further detail in Section 3.6. In addition to performance metrics, data analysis also occurs with Wireshark. Wireshark is a network packet analyzer and is used to inspect anomalous network packets that are reported by the ADS.

3.2.1.6 Threat Analysis and Response.

Threat analysis and response refers to the human element of the system in which a human operator inspects the anomalous events reported by the output logs and makes decisions in response to the activity. The response is based on the security policy of a specific ICS environment [SFS11].

3.2.2 Experiments.

Three experiments are developed and performed for this research. A brief description of each experiment is provided here. Chapter IV goes into more detail on the implementation of each experiment.

Experimental Validation. An experimental validation is used to test the effectiveness of the approach and methodology described in this chapter. This experiment also establishes a baseline p-tree that is used in Experiment 1 and is referred to as 'Baseline' p-tree throughout the rest of this document. The 'Baseline' p-tree is used as the model of normal network behavior between an EWS and PLC and contains no anomalous or malicious data.

Experiment 1. Experiment 1 assesses the effectiveness of the ADS in being able to distinguish anomalous activity from normal activity. Two independent test traces (labeled *Normal* and *Malicious*) are introduced to the system separately and are compared with the 'Baseline' p-tree for matches. Anomalous behavior is reported if there is no match in the 'Baseline' p-tree.

Experiment 2. Experiment 2 assesses the ADS's ability to identify anomalous activity even if anomalies are learned with normal network traffic. This experiment addresses the problematic scenario in which a threat actor slowly trains an ADS to accept anomalies as normal behavior to gain access to an ICS network.

3.3 System Boundaries

Figure 5 is a depiction of the SUT: the ScriptGenE framework implemented as an ADS. The components under test (CUT) consist of two Python-based scripts, ScriptGenE.py and ScriptGenEreplay.py. The workload consists of network traces and are applied to the system; the traces consist of both attack-free and malicious network data, which is designed to simulate anomalous traffic. The network traces were generated between the EWS and PLC performing two different tasks. With the input network traces, ScriptGenE.py builds a p-tree based on build options. The output p-tree is then imported to ScriptGenEreplay.py replay script, and then used to detect anomalies in the incoming 'new' network traffic. The ADS is simulated as being placed between the EWS and PLC. Performance metrics used to evaluate the SUT are TPR, FPR, TNR, and false negative rate (FNR). More detail about each section in the SUT are given as this chapter progresses.
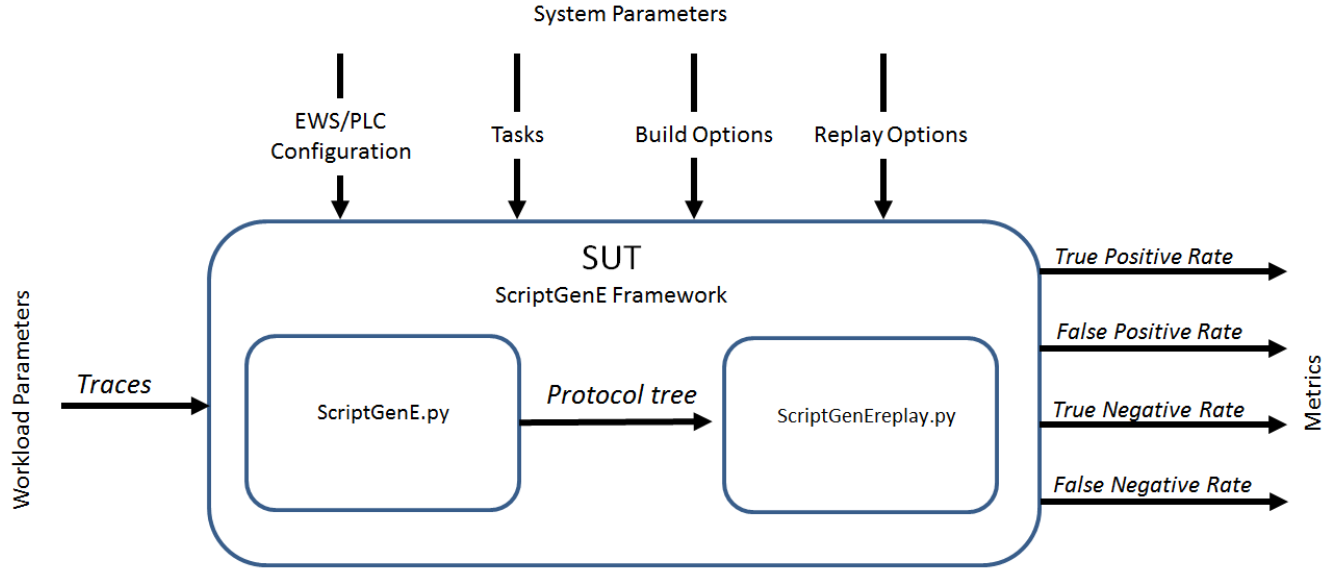
Figure 5. System Under Test: Anomaly Detection System

3.4   Workload

The first part of the intrusion detection process is identifying the information source. For this research, the information source is the workload for the SUT mentioned in Section 3.2. Simulated ICS network traffic is generated between a PLC and EWS. The workload for the ADS consists of network traffic generated by performing normal application tasks expected between a PLC and EWS, specifically, the tasks are 'uploading' a ladder logic program to the PLC and establishing communication with the PLC from the EWS, respectively. Application tasks are discussed with system parameters in Section 3.5.

The network traffic/traces for this research are characterized into three types:

1. *Normal* – consists of attack-free network traffic

2. *Malicious* – consists of simulated malicious, anomalous network traffic

3. *Combined* – consists of merged *Normal* and *Malicious* network traffic

30

Normal network traffic represents normal network behavior between the EWS and PLC and contains no anomalous or malicious packets. An experimental script developed for this research is configured to collect 10 reference Pcaps based on network traffic generated from performing the two tasks previously mentioned. Of the 10 reference Pcaps, five traces are chosen randomly to create one experimental Pcap which is used as the input trace to the ADS. Based on previous research, five traces are determined to be adequate in accurately emulating a PLC communicating over EtherNet/IP because five traces provides byte variability in TCP header fields that are supposed to change (i.e., sender context field) [War15]. Furthermore, five input traces overcomes sampling issues and achieves variability allowing the backtracking algorithm to handle more transitions during replay [War15]. An accurate model of normal network behavior provides a good baseline p-tree with which the other types of network traces in the workload can be compared for intrusions or anomalies. The experimental Pcap representing *Normal* network traffic is used for Experimental Validation and Experiment 1.

Malicious network traffic is created to represent anomalous, malicious packets sent to the PLC. The *Malicious* network Pcap is modeled after a denial of service (DoS) SYN flood and created by a third party, open source TCP/IP packet generator called HPING2 [San09]. One Pcap consisting of 400 extraneous SYN packets represents the SYN flood DoS attack generated for use as an input to the ADS for Experiment 1.

The *Combined* network traffic consists of merged *Normal* and *Malicious* traces. Ten reference Pcaps are generated containing *Normal* network traffic, from which five traces are randomly selected to form one experimental Pcap. The *Malicious* Pcap described previouly is merged with the experimental Pcap using Wireshark's 'merge' function to create the *Combined* network trace used as input to the ADS for Experiment 2.

3.5 System Parameters

As shown in Figure 5, the system parameters consist of EWS/PLC configuration, build options, replay options and tasks. This section provides details on the system parameters of the ScriptGenE framework ADS.

3.5.1 PLC Configuration.

The PLC configuration for this research is outlined below.

Allen-Bradley ControlLogix5561 (L61) PLC

- Firmware Version 19.015

- Slot 0 – EtherNet/IP ENBT

- Slot 1 – L61 Controller

- Slot 6 – DC Output Module

3.5.2 EWS Configuration.

A description of the EWS software configuration for the PLC is outlined below.

Rockwell Automation RSLogix5000 Software Suite

- Version 19.01.00 (CPR 9 SR 3)

- Ladder Logic Sample Program: LightingDemo.ACD

- RSLinx Classic Lite, Version 2.59.02 CPR 9 SR 5

3.5.3 Tasks.

Two tasks are chosen to simulate normal network traffic behavior between a PLC and EWS. The first task establishes communication between the PLC and EWS.

RSLinx is used to establish this connection using EtherNet/IP since the PLC and EWS are connected via Ethernet to a switch. RSLinx is a Windows-based software package that enables communication between all Rockwell Industrial Control hardware and an HMI and EWS [Roc14]. RSLinx uses the application RSWho to browse the network for a PLC and populates the RSLinx graphical user interface (GUI) with available modules. The second task is 'uploading' a program to the PLC. The program is a simple ladder logic program executed on the PLC to perform a lighting demo by providing power to lights in a pattern via the DC output module.

3.5.4 ScriptGenE.py Build Options.

The ScriptGen.py build options for the experiment are:

```
./ScriptGenE.py $IP -p 44818 -M 0.2
```

where, M, is the macroclustering threshold, -p is the target port, which is port 44818. In the ScriptGenE framework, 0.5 is the default value for the macroclustering threshold, which defines the minimum distance between two sequences and it controls how clusters are created. The threshold can be set between 0.1 and 1.0 in the command line or in a configuration file when ScriptGenE.py is run. If M is set to a small value, many small clusters are created. If M is set to a big value, then fewer, larger clusters are created. Based on findings conducted during pilot testing, a threshold of 0.2 is chosen it allows the ScriptGenE script to create small enough clusters that are refined enough for microclustering to be effective.

3.5.5   ScriptGenEreplay.py Replay Options.

The ScriptGenEreplay.py replay options are:

```
./ScriptGenEreplay.py <p-tree> -I $IFACE -f -r always
```

where `-f` allows the server to run forever or until a keyboard interrupt shuts the server down. The "`-r` always" option enables the backtracking feature. A p-tree is loaded to the ScriptGenEreplay.py script and used for comparison with other p-trees. Backtracking is a novel algorithm within the ScriptGenEreplay.py script. Its purpose is to allow for more flexibility in the emulation of ICS devices. Backtracking is employed whenever an unknown transition must be handled by locating within p-tree the earliest edge that matches the current client request and return the corresponding next state (i.e., the server response) [War15]. Backtracking is necessary for the ScriptGenE framework to be utilized as an ADS.

3.6 Performance Metrics

Accuracy is used to measure the performance of the ADS. As shown in Figure 4, p-trees (labeled 'Baseline' and 'new') are compared using the ScriptGenEreplay.py script to determine if they are different from each other. The expected outcome is one of the following:

1. The compared p-trees are determined to be different, indicating an anomaly is detected,

2. The compared p-trees are determined to be similar, indicating no anomaly is detected.

The described outcomes are defined with the following metrics:

- true positive (TP) –anomaly exists and detected

- false positive (FP) – anomaly does not exist but an anomaly is reported

- true negative (TN) – anomaly does not exist and not detected

- false negative (FN) – anomaly exists but an anomaly is not reported

The true positive rate (TPR) and false positive rate (FPR) are common measures of accuracy for diagnostic systems and are used to determine the effectiveness of the ADS in classifying anomalies [Dun13, Swe88]. The TPR is defined as the number of correctly identified anomalies (TP) divided by the total number of anomalies (TP+FN).

$$TPR = \frac{TP}{(TP + FN)} \tag{1}$$

The FPR is the number of incorrectly identified anomalies (FP) divided by the total number of instances with no anomaly (FP+TN). The equation is as follows:

$$FPR = \frac{FP}{(FP + TN)} \tag{2}$$

The true negative rate (TNR) can be derived from the FPR as follows:

$$TNR = 1 - FPR \tag{3}$$

The false negative Rate (FNR) can be derived from the TPR as follows:

$$FNR = 1 - TPR \tag{4}$$

For this research, the ADS is considered effective (and ScriptGenE is considered viable as an ADS) if the TPR is greater than or equal to 0.9 and the FPR is less than 0.1.

Mutation Rates. Mutating regions are discovered by the Region Analysis algorithm in the ScriptGenE framework and expressed by the value, mutation rate (MR), in the following equation [War15]:

$$MR = \frac{\#\ unique\ bytes}{total\ bytes} \tag{5}$$

Mutation rates are calculated in this research to determine if there are differences between p-trees, namely the p-tree generated representing normal network behavior and the p-tree introduced to the ADS representing 'new' network traffic. The MR for each p-tree is calculated and provided as an output from the Region Analysis operation. The difference between the two MRs is taken to determine if the p-trees

are *different* or the *similar*. One of the following outcomes is expected:

- If the $MR_{Baseline}$ - MR_{new} is greater than 0.1 (10%), then p-trees are identified as *different* (+)

- If the $MR_{Baseline}$ - MR_{new} is less than or equal to 0.1 (10%), then p-trees are identified as *similar* (-)

A (-) is used to designate similar p-trees (the null hypothesis), and the (+) is used to designate differing p-trees (the alternate hypotheis) for this research.

3.7 Environment

The experimental set up, shown in Figure 6, includes three physical machines: one L61 PLC, one HP laptop with a Linux OS, and one HP laptop with a Windows OS; all were connected via Ethernet to a spanning port switch. The Windows 7 laptop hosts a Windows XP virtual machine (VM), which includes RSLogix 5000 suite. The VM simulates the EWS in this research. The Linux laptop is used to sniff and capture all network traffic between the PLC and EWS via the NetGear ProSafe GS108E spanning port switch. The Linux laptop contains the ScriptGenE framework, therefore the laptop simulates the ADS on the network. The Linux laptop is also used to run the experiments and collect data.
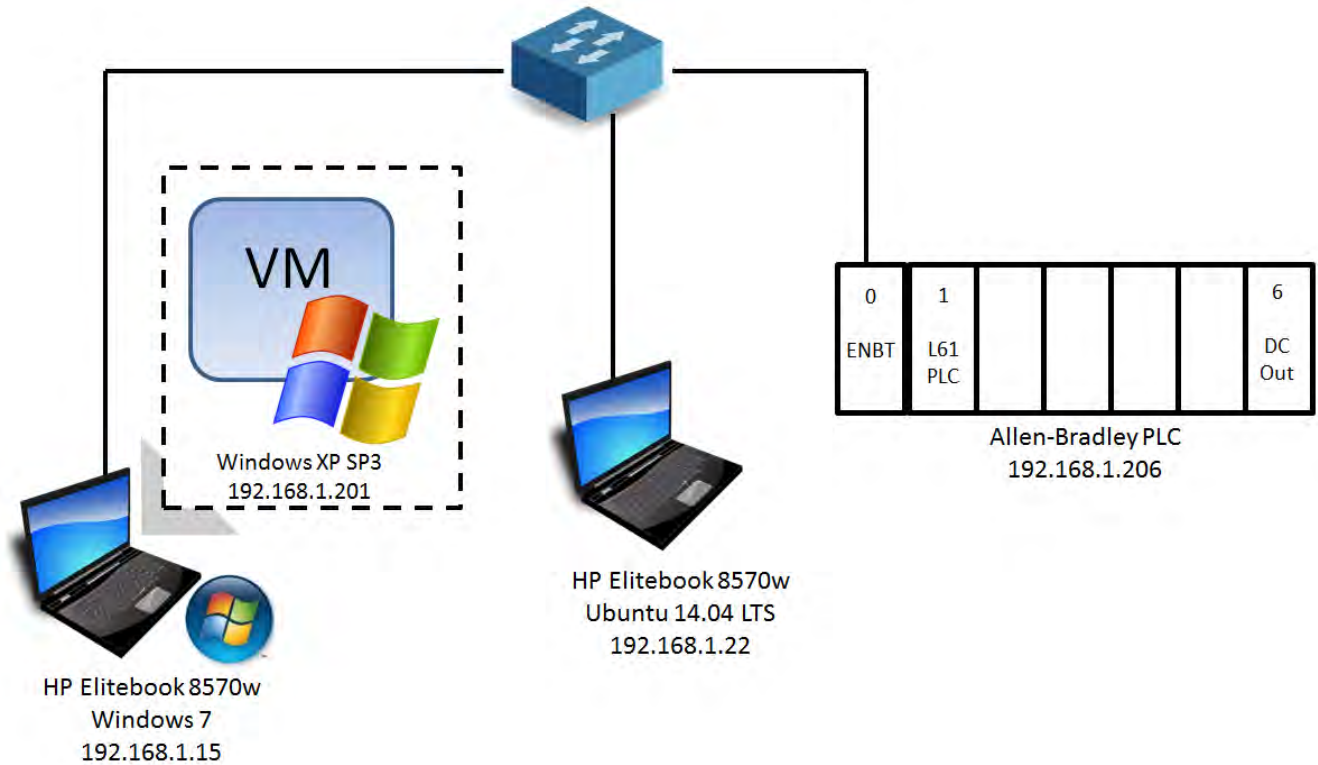
Figure 6. Environmental Setup

Machine Configurations. The specific laptop configurations are:

Laptop 1: Hewlett-Packard Elitebook 8570w

- Microsoft Windows 7 Service Pack 1

- 2.6GHz Intel Core i7-3720QM processor (4 cores)

- 16GB RAM

- VMware Workstation version 11.0.0 build-2305329

  Windows XP SP3, Version 2002

  2 processor cores, 3 GB RAM, 60 GB HD

Laptop 2: Hewlett-Packard Elitebook 8570w

- Ubuntu 14.04 LTS, kernel 3.13.0

- 2.6GHz Intel Core i7-3720QM processor (4 cores)

- 16GB RAM

- `tcpdump` version 4.5.1, libpcap version 1.5.3

3.8 Evaluation Technique

The evaluation technique measures the classification accuracy of the ScriptGenE framework as an ADS. The ScriptGenE framework is treated as a learning algorithm, and a 10-fold cross validation method is selected to ensure accuracy in the ADS's performance. This research uses TPR, FPR, TNR, and FNR as the means of evaluating the performance of the ADS. For this research, a TPR of 0.9 and a FPR of 0.1 is considered acceptable to claim that the ScriptGenE framework is a viable ADS. Because it was not originally designed as an ADS, meeting these performance metrics would allow for further investigation and design consideration as an ADS.

3.8.1 10-fold Cross Validation.

In k-fold cross validation, a data set is split into k mutually exclusive subsets. Cross validation takes advantage of using the entire data set for training and testing of a machine learning algorithm [Koh95]. Bias and accuracy can be issues if the folds do not contain the same percentage of attributes or features as the overall data set. To mitigate this, folds are randomized and stratified because stratification of the folds yields a less biased estimate of accuracy [Koh95].

Once the folds are generated, k rounds of learning are performed. k-1 folds are used for training (or learning) in each round. The kth fold, the remaining fold which

is unlabeled for that round, is used as the testing data set [Koh95, Wer14]. The true estimate of accuracy is determined by taking the final averaged accuracy from the k rounds [Koh95].

Ten-fold cross validation is chosen to measure the accuracy of the ScriptGenE as an ADS. [Koh95] determined that to achieve low variance and reasonably low biased estimations, 10 folds are required. Therefore, selecting $k = 10$ can provide an accurate estimate of the ability to classify anomalies in the ScriptGenE ADS.

3.8.2 Cross Validation Method Applied to ScriptGenE ADS.

In the 10-fold cross-validation technique, 10 rounds are performed in which each round learns on nine folds and tests on the remaining tenth fold. Therefore, every fold is eventually used for testing. Figure 7 depicts k rounds of the cross validation technique [Koh95].
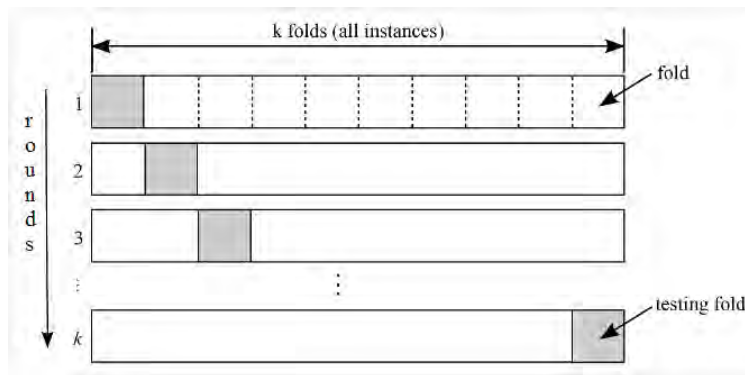
Figure 7. k-fold Cross Validation [Koh95]

An open-source tool, known as SplitCap, is used to split Pcap files into multiple files based on user datagram protocol (UDP) or TCP sessions or host-pair [Spl]. SplitCap is used on the packet captures that were generated for this study. Each Pcap file that is created by the SplitCap tool is considered a fold to either be used for training or testing on the ADS.

The 10-fold cross validation method is used for experimental validation mentioned

in Section 3.2. To characterize normal 'Baseline' behavior in the system, an analysis of each testing fold is performed after every round. The following procedure outlines the experimental validation for this research.

For each round,

1. One Pcap containing *Normal* network behavior is split into 10 folds by SplitCap.

2. Nine folds are used to train the ADS. 'Trained p-tree' is produced and its mutation rate is calculated.

3. The tenth fold is used to test the ADS. 'Test p-tree' is produced and its mutation rate is calculated.

4. 'Test p-tree' and 'Trained p-tree' are compared using ScriptGenEreplay.py script. Matches/no matches are reported in output log.

5. Calculate difference between 'Trained p-tree' MR and 'Test p-tree' MR.

The entire procedure is performed 10 times to ensure all folds are used for testing. The goal of the experimental validation is to ensure that a Pcap representative of attack-free, anomaly-free *Normal* network traffic is used as a 'Baseline' p-tree.

3.9 Experimental Design

This study uses a one-factor factorial design. The factor under study is the network traffic generated and captured between the EWS and PLC. There are three levels: *Normal*, *Malicious*, *Combined*, also mentioned in Section 3.4. Two experiments are used to determine the effectiveness of the ADS in addition to an experimental validation. The experiments are labeled 'Experimental Validation,' 'Experiment 1,' and 'Experiment 2.' Brief descriptions of each are provided in Section 3.2. The 10-fold cross validation method is used for the Experimental Validation as well as Experiment

2. Although 10-fold cross validation is used for both experiments, five repetitions of each experiment are run to ensure an accurate estimate of the ADS's performance. For Experiment 1, 10 repetitions are run (five repetitions with the *Normal* trace and five repetitions with the *Malicious* trace) to ensure the ADS is able to correctly distinguish between normal and anomalous network behavior. The rest of this section describes implementation details related to the experiments conducted.

3.9.1   Generating experimental network traces.

An experimental script is developed for this research to automate network traffic between the EWS and PLC. The experimental script was originally developed by [War15]. The script is modified to generate network traffic between the EWS and PLC while performing two tasks, but does not include taking measurements as was done in the original experimental script. An overview of the experimental script follows. Ten reference traces are created by generating traffic between the PLC and the EWS. The network traffic is captured by using `tcpdump`. The `tcpdump` command used is as follows:

```
tcpdump -i $IFACE -s 65535 -w <file>
```

where `-i` designates which network interface to listen on, `-s` specifies the snapshot length to be the full 65535 bytes and `-w` instructs the tool to write the raw packets to a file. As shown in Figure 6, the Ubuntu laptop is positioned between the PLC and EWS (represented by the Windows VM laptop). The switch is configured via a management console to mirror traffic between the PLC and EWS onto a third port where the Ubuntu machine is connected. This configuration allows the Ubuntu machine to use `tcpdump` to capture packets between the PLC and EWS via the Ethernet (`eth0`) interface.

In order to simulate *Normal* behavior between the PLC and EWS, two tasks are

performed:

- Establish communication between the PLC and EWS using RSLinx

- Uploading a simple ladder logic program from the EWS to the PLC

These tasks are executed using GUI automation software known as SikuliX, which is also used in [War15]. SikuliX allows anything that can be seen on the screen of a computer to be emulated and automated, including keyboard and mouse events [Hoc15]. This research adapts the SikuliX scripts from [War15] to automate the tasks mentioned above. An example of how the RSLinx task is developed in the SikuliX IDE is shown in Figure 8. The figure shows the commands used to open the RSLinx window to begin establishing communication with the PLC from the EWS.

Figure 8. RSLinx SikuliX script snippet

These tasks are chosen because they are representative of common tasks between a PLC and EWS that normally need to be accomplished when a new PLC comes online in an ICS network [Roc00]. Based on pilot studies performed previously, it takes 5 seconds to successfully upload the ladder logic program to the PLC. After

the ladder logic program is uploaded to the PLC, the experimental script is then instructed to run for five seconds to collect network traffic of the PLC performing the lighting demo ladder logic task. This capture ensures that not only is the TCP communication represented in the network trace, but the activity between the PLC and EWS is recorded to provide an accurate model of behavior between the PLC and EWS. After each reference trace is collected, the experimental script resets the PLC and EWS to begin another capture. From the 10 reference traces, five are randomly chosen to create one experimental Pcap file. As indicated in previous research, five traces are recommended to overcome sampling issues and achieve byte variability to allow the backtracking algorithm to handle more transitions during replay [War15].

3.9.2 Splitting the Pcap.

SplitCap is used to split the experimental capture into 10 subsets, or folds. Split-Cap divides each trace based on TCP sessions, therefore each fold is expected to contain at least one TCP connection [Spl]. Each fold is stored as a separate Pcap file. Recall that in 10-fold cross validation, nine folds are used for training and the tenth fold is used for testing. When performing the Experimental Validation, one experimental Pcap is split into 10 folds with the SplitCap tool. Therefore, for each round, nine Pcap files are merged together by using the 'merge' function in Wireshark.

The command to split the Pcap is:

```
SplitCap -r <pcap_input_file> -o <output_directory>}
```

The intent behind splitting the trace Pcap into 10 folds is to take advantage of utilizing the entire trace in training and evaluating the ScriptGenE framework. Since there are 10 folds, 10 rounds of testing are executed to ensure each fold is utilized (refer to Section 3.8). In each round, nine folds are used for training, meaning ScriptGenE.py creates on p-tree that is designated as the network 'Baseline' for that

44

round. The 10th fold is used for testing, and the resultant p-tree is compared against the 'Baseline.'

3.9.3   P-tree comparison and anomaly detection.

Once the 'Baseline' p-tree is generated from ScriptGenE.py, it is loaded to the ScriptGenEreplay.py script and established as a server. The 'test' p-tree is loaded to a client.py script and specifies the IP address and port number of the server to connect to the server..

The build options for ScriptGenEreplay.py in which the 'baseline' p-tree is used for comparison is:

```
./ScriptGenEreplay.py <tree_file> <port> -i \$IFNAME -f -r always}
```

where `-i` is the interface, `-f` is to run the server `forever`, and `-r always` is to enable backtracking.

The build options for the client.py script in which the 'test' p-tree is loaded:

```
./client.py <tree_file> <server_ip> <server_port>}
```

Once both p-trees are loaded, the client selects an edge from the 'test' p-tree and sends it to the server to locate a corresponding node in the 'Baseline' p-tree. One of the novel features of the ScriptGenE framework is the ability the replay script has to backtrack through a p-tree. Backtracking on the 'Baseline' p-tree is permitted via the `-r` build option. If a requesting client edge is not found immediately in the 'Baseline' p-tree, the backtracking algorithm searches for the nearest matching edge. If found, the corresponding server message node is sent to the client and another edge is selected from the 'test' p-tree. If the node does not exist, i.e., there is no match in the 'Baseline' p-tree, it is assumed that an anomaly exists. In the event anomalous activity is identified, an alert is generated via the ScriptGenEreplay.py output log, which is to be viewed on the EWS for a human operator for action. Figure 9 shows

an example of the alert output that is generated if an anomaly is detected.

```
-----------------------------------------------------------
Getting server msg...
==> Sending smsg 11 (62 bytes)... done
getting data from client...

****WARNING****Unable to find edge for msg 12
****WARNING****No data sent
****WARNING****Continue or Close Connection

-----------------------------------------------------------
```

Figure 9. Alert Output Example

3.10   Design Summary

The goal of this research is to determine the feasibility of the ScriptGenE framework as an ADS in an ICS environment. ScriptGenE is treated as an ADS implemented between a simulated EWS and real PLC. For the ADS to be deemed viable, a TPR of 0.9 and FPR of 0.1 are selected as acceptable thresholds and can be tailored to meet specific and individual system requirements at a later time if these thresholds are first met in this proof-of-concept study. This chapter describes the experimental methodology and equipment used to test the accuracy of the ScriptGenE framework as an ADS. To evaluate the testing method, 10-fold cross-validation is selected to ensure an accurate estimate of the performance of the ADS.

IV. Results and Analysis

This chapter provides details of the implementation and results of the ScriptGenE framework ADS. Three experiments are evaluated in this chapter and analysis of each of the data sets are provided as well as an assessment of the TPR/FPR in the form of a receiving operating characteristic (ROC) curve of each experiment.

4.1 Experimental Validation

4.1.1 Hypothesis.

This experiment focuses on establishing a baseline model of normal network behavior between a PLC and EWS using 10-fold cross validation. Since the experimental Pcap under question is free of attacks, it is expected that no anomalies are reported in the output log from each of the 10 testing folds. Additionally, it is also expected that the overall FPR $= 0$ and the TNR is $= 1$. Recall that TNR is a performance metric defined as an anomaly does not exist and is not detected by the IDS.

4.1.2 Analysis.

Ten rounds of the cross validation method are performed to complete one experiment. Five total experiments are performed for the Experimental Validation to ensure an accurate estimate of the ADS's performance as well as establish a baseline model of normal network behavior between a PLC and EWS. Each round consists of nine training folds and one test fold. The training folds produce one p-tree (referred hereafter to as the 'Trained p-tree') and the test fold produces one p-tree (referred to as the 'Test p-tree'). Recall from Section 3.6 that mutation rates are calculated from both p-trees and the difference is taken to determine if the p-trees are different or similar.

As expected, no alert was produced in the output logs from any of the rounds and experiments. Mutation rates reported in the Region Analysis outputs also confirms that the p-trees produced in each round are similar, i.e., the difference between the mutation rates of the 'Trained p-tree' and 'Test p-tree' is less than 0.1. Since no anomalies are reported and confirmed by similar p-trees in each round, the FPR = 0 and the TNR = 1. The Experimental Validation provides the 'Baseline' p-tree to be used in Experiment 1 for comparison and represents normal network behavior between a PLC and EWS. Figure 10 shows the resultant 'Baseline' p-tree and a magnified snippet of the ROOT node. Labels within the symbolic nodes simply represent the corresponding server message from the Pcap file that is used to build the p-tree.
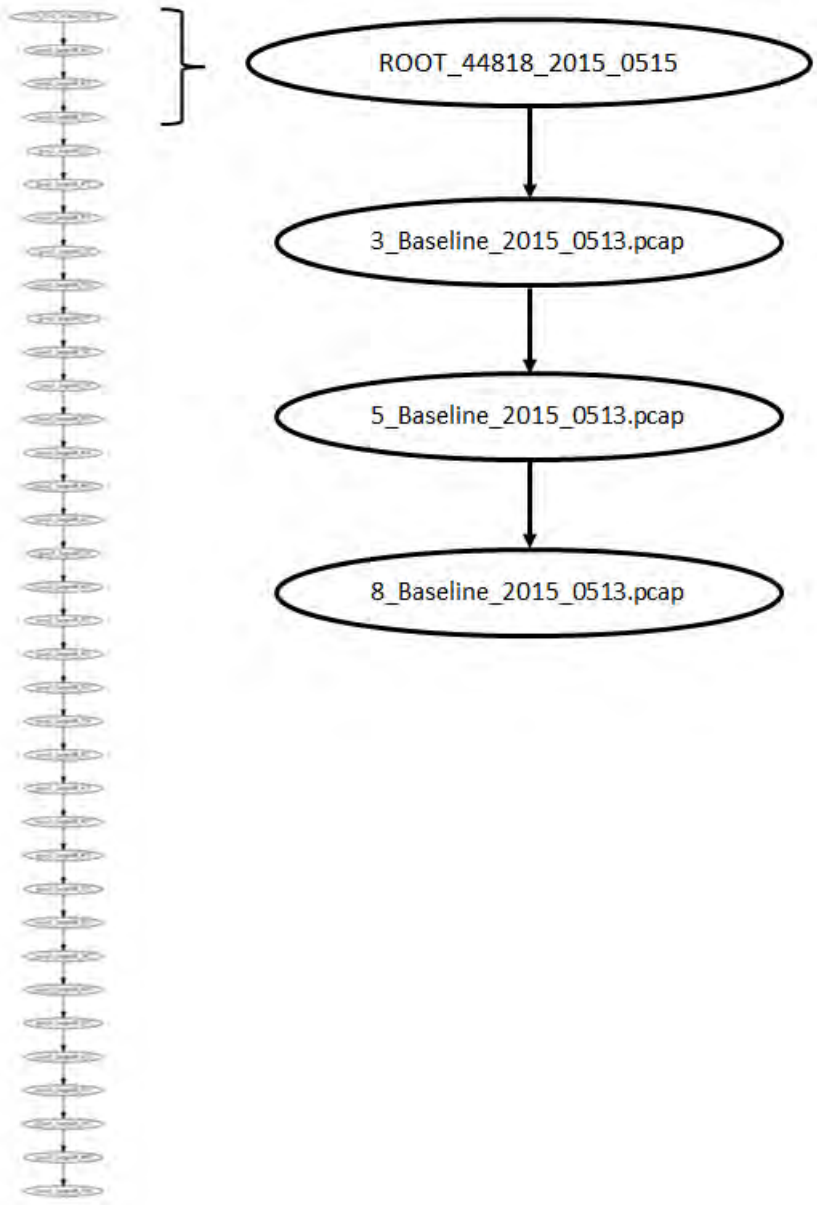
Figure 10. Final Baseline p-tree snippet

4.2 Experiment 1: Attack-free Network Traffic

4.2.1 Hypothesis.

This experiment uses attack-free network traffic generated between a PLC and EWS (*Normal*), the *Malicious* network trace, and the 'Baseline' p-tree created from the Experimental Validation. The *Normal* network trace is provided as input to the ADS and the resultant p-tree is used to compare against the 'Baseline' p-tree. Next, the *Malicious* network trace is provided as input to the ADS and the resultant p-tree is compared against the 'Baseline' p-tree in the same manner. This procedure is repeated 5 times each (total of 10 experiment runs) to ensure an accurate measure of the performance of the ADS.

A common issue with ADSs, in comparison to signature-based IDSs, is the reporting of higher false positive rates. Anomalies are logged via the ScriptGenEreplay.py replay script, should any be reported. This logging mechanism simulates information that would be useful to an operator or engineer for further investigation.

It is hypothesized that since no anomalous behavior is expected in these traces, no false positives should be reported with the *Normal* network trace. However, the *Malicious* network trace should produce alerts in the output log and the difference between the mutation rates of the 'Baseline' and *Malicious* p-trees should be greater than 0.1. An FPR of 0 is expected with the *Normal* test run, and a TPR of 1 is expected with the *Malicious* test. To be clear, it must be noted that a FPR of 0 is sought after for this experiment to determine if the ScriptGenEreplay.py script would recognize similar p-trees as different. This result would indicate that the ScriptGenE framework is not a viable solution as an ADS.

4.2.2 Analysis.

A ROC curve is a function of the TPR and FPR. An effective ADS deployed in an ICS environment cannot afford tolerate even one FP, especially in critical infrastructure, because it can be costly for an operator or engineer to stop operations to inspect the system looking for malicious activity that does not exist, but is reported. In a ROC curve, the horizontal axis denotes the FPR and the vertical denotes the TPR of a classifier under investigation. The area under the curve (AUC) measures the effectiveness of the ADS; an area close to 1, indicates that the ADS is effective in classifying between normal and anomalous behavior [Swe88]. A ROC curve closer to the reference straight line in the graph indicates the ADS is ineffective at classifying between normal and anomalous network behavior.

After 10 experimental runs (five with *Normal* and five with *Malicious* network traces), the TPR is 1 and the FPR is 0. The *Malicious* network traffic caused alerts to be generated in the output logs, as expected. Figure 11 shows the corresponding ROC curve for this experiment. Note that the AUC is 1. For this experiment, the ADS perfectly discriminates between normal and anomalous network traffic as expected. Closer observations of the Region Analysis outputs from ScriptGenE.py show that each of the *Normal* p-trees did not differ from the 'Baseline' p-tree, meaning mutating regions are similar. This is to be expected because the *normal* and the 'Baseline' p-trees were generated with attack-free network traffic. The macroclustering and Region Analysis algorithms align and analyze bytes that are semantically the same for both the 'Baseline' p-tree and the *Normal* p-tree [LMD05, War15]. However, the *Malicious* experiment runs did produce the opposite result from the *Normal* experiments, as expected. Anomalous behavior is detected by the ADS in each run because the *Malicious* p-trees do not match the 'Baseline' p-tree. As a result, alerts are generated in the output logs for the operator to further analyze.
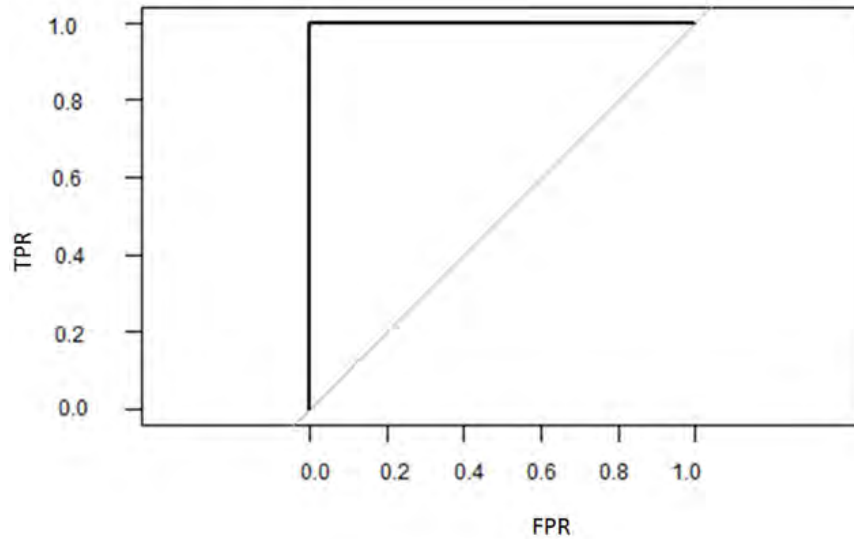
Figure 11. ROC curve for Experiment 1

4.3 Experiment 2: Simulated **SYN** Flood Denial of Service Attack

4.3.1 Hypothesis.

This experiment focuses on the scenario in which a malicious actor remotely trains the ADS to accept malicious behavior with the intent of launching an attack that would go undetected because the ADS is trained to see the behavior as normal. However, the goal of this experiment is to determine if the ADS can still detect anomalies even if anomalies exist in the network traffic used for training. The 10-fold cross validation method is used for this experiment. A Pcap containing *Combined* network traffic is used to train and test the ADS and ten rounds are completed.

For this experiment, the attack in question is the SYN flood attack. This attack is chosen because ScriptGenE builds p-trees from valid TCP connections parsed from Pcaps. Valid TCP connections are defined in [War15] as observing a 3-way handshake or observing a server sending data to a client, which is defined as a 'missed handshake session.' ScriptGenE initially handles 'missed handshake sessions' by building

a dummy client message with the data set to 'MISSING.' However, during the main consolidation in the ScriptGenE.py script, the consolidated 'missed handshake edges' are deleted. To execute this experiment, a minor modification to the script had to be made so branches consolidated under the 'MISSING' label would not be deleted as they represent incomplete 3-way TCP handshakes. Pilot testing of this modification revealed no negative effect on ScriptGenE.py functionality.

The SYN flood attack is crafted using an open-source Linux-based tool known as hping2 [San09]. Hping2 is a free packet generator and analyzer for the TCP/IP protocol. The command to execute the SYN flood with hping2 is:

```
./hping2 -i u1 -S -p 44818 -c 400
```

where -i is the interval to wait 1 microsecond between each packet, -S indicates using the SYN flag, -p is the destination port 44818 for EtherNet/IP, and -c is the packet count, which is set to 400 packets. Pilot studies of this attack on the PLC showed that sending over 450 packets overwhelmed the PLC in the experimental environment, rendering it ineffective in experimentation. The intent of this experiment is to produce anomalous, malicious network events that simulate an attack, but could potentially be identified by the ADS.

It is hypothesized that since anomalous behavior is present in the *Combine* trace, true positives should be reported and the 'Trained p-trees' are different from the 'Test p-trees.' The goal of this experiment is to use the ScriptGenE framework to identify anomalous behavior even when it is learned and to achieve a FPR of 0.1 and TPR of 0.9.

4.3.2 Analysis.

Unlike the first experiment, the ROC curve is not a perfect curve with an AUC of 1. Using the R package, *pROC*, the AUC for the ROC curve in Figure 12 is 0.9002.

TPR is 0.911, and the FPR is 0.054. As expected with the *Combined* network trace, alerts are generated in the output logs. Comparison of the mutation rates for each round in the experiment indicate that eight of the 10 rounds identified differing p-trees and 2 of the 10 identified similar p-trees. Table 1 captures the results for the mutation rates of the p-trees for each round. The column labeled 'diffMR' shows the difference in the mutation rates between the 'Trained p-tree' and the 'Test p-tree' for every round. Recall from Section 3.6, mutation rates that differ by more than 0.1 are considered different reported as (+). Mutation rates that differ by less than or equal to 0.1 are reported similar and receive a (-) as a result.

Table 1. Difference in Mutation Rates for 'Trained' and 'Test' p-trees for 10 rounds

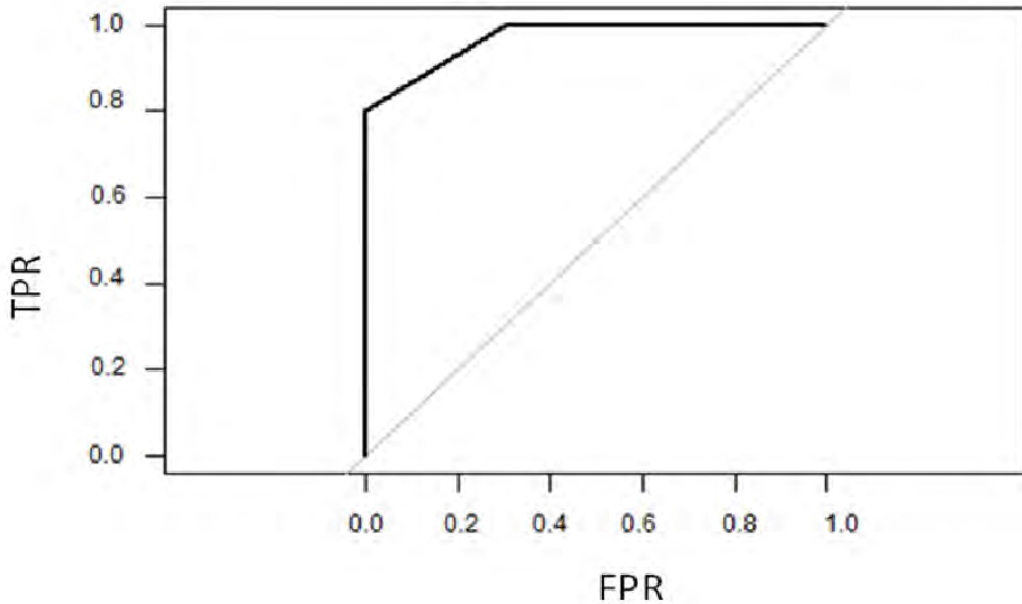| Round	diffMR	+/-
1	0.322	+
2	0.334	+
3	0.351	+
4	0.289	+
5	0.363	+
6	0.104	-
7	0.351	+
8	0.344	+
9	0.332	+
10	0.100	-

Figure 12. ROC curve for Experiment 2

The goal of this experiment is to accurately characterize anomalous network behavior from normal network behavior given that the ADS is training on *Combined* network traffic. The lower TPR and higher FPR (in comparison to Experiment 1) could be due to the incorporation of extra SYN packets to the 'normal' network traffic. During the training phase, the ADS could be trained into believing those extraneous SYN packets are 'normal.' A major drawback of ADSs, is that an attacker who is patient, can incorporate malicious packets into an ICS network where an ADS can be tricked into accepting those packets as normal network behavior. Over time, the attacker could increase his immunity to the ADS and eventually obtain full access into the system [YUH06, JY13].

4.4 Summary

This chapter provides details about the implementation of the ADS. The ADS in question is the ScriptGenE framework. The Experimental Validation shows that the 'Baseline' network traffic capture contains no anomalies and is therefore the representation of normal network behavior between the EWS and PLC. The resultant 'Baseline' p-tree is used for Experiment 1. The overall TNR for Experiment 1 is 1 and the FPR is 0. The AUC for Experiment 1 is 1 and shows that the ADS can correctly discriminate between normal and anomalous network behavior given the 'Baseline' model. For Experiment 2, the TPR is 0.9011 and the FPR is 0.054. The ROC curves and these findings demonstrate the ability of the ScriptGenE framework to classify anomalous network behavior from normal network behavior.

V. Conclusions

This chapter summarizes the results of the research. Section 5.1 discusses the conclusions based on the results in Chapter 4. Section 5.2 identifies contributions of this research, and Section 5.3 discusses recommendations for future work.

5.1 Research Conclusions

This research presents a study on the feasibility of implementing an automatically configurable PLC emulator as an anomaly detection system. The ScriptGenE framework was originally developed to emulate PLCs in honeypots in ICS environments. The novel clustering, distance and search algorithms offered unique features that made the ScriptGenE framework a viable option as an ADS. The backtracking algorithm inherent in the ScriptGenE framework is also noted as a useful tool in pattern matching.

The ScriptGenE framework is implemented as an ADS between an EWS and PLC. In this position, the ADS monitors generated network traffic and develops a model of normal behavior and stores it as a p-tree. Incoming network traffic, which may contain malicious traffic, undergoes the same process and a p-tree is built to be compared against the 'Baseline' p-tree. In theory, if the new tree is able to find matches in the 'Baseline' tree, then the network traffic is permitted to pass through the system and on to the real network. The ability for the ADS to block and pass legitimate network traffic after it has been verified; this is not implemented for this research. If the ADS detects anomalies, i.e., no matches to the 'Baseline' p-tree, then an alert is logged for the operator to investigate. Like any IDS, the ScriptGenE framework implemented as an ADS still requires human operation to inspect and validate output logs when alerts are generated for anomalies.

The ADS presented in this research is able to detect anomalous network behavior and maintain a low FPR when trained with attack-free network traffic.

For the Experimental Validation, a 'Baseline' p-tree is validated as the representation of normal network behavior and is used in Experiment 1. The 10-fold cross validation method is used for the Experimental Validation as well as Experiment 2. In Experiment 1, the ADS achieves a FPR of 0 and TNR of 1. Analyzing the ROC curve, the AUC is 1, which shows that the ADS is a perfect classifier when no attack data is available to learn or train on.

However, for Experiment 2, the ADS achieves a TPR of 0.9011, FPR of 0.054 and AUC is 0.9002. Introduction of malicious, anomalous packets decreased the performance of the ADS due to the system building p-trees with anomalies in the form of extra SYN packets incorporated into the folds. Indeed, this showcases a common disadvantage of ADSs, however, it should not be unexpected that the ADS would view anomalous data as normal if it trained on traffic with anomalies combined with normal traffic. However, it seems counterintuitive that any anomaly would be reported at all if the ADS is trained to accept the *Combined* network traffic as "normal." The results of the experiment show that the ADS *is* able to identify anomalies given that it trained with *Combined* network traffic. Further inspection of this result reveals that the "missed handshake sessions" is caused by the extra SYN packets. In other words, as ScriptGenE.py parses through a Pcap and builds a p-tree, the extra SYN packets are represented in the p-tree as an edge with a weight based on the number of times the SYN messages are merged together. The node attached to the client edge is labeled as 'MISSING' indicating that data from the server is missing and is considered a dummy node, but part of the p-tree. As the ADS trains on the *Combined* network traffic, it generates a 'Trained p-tree' which contains the 'MISSING' node and client edge. It is observed the reason anomalies are reported in this experiment

is due to the varying weights of the client edges with 'MISSING' nodes. For instance, if the 'Trained p-tree' contained a 'MISSING' node with a client edge weight of five (five SYN packets), and the 'Test p-tree' contained a 'MISSING' node with an edge weight of 10 (representing 10 SYN packets), the ADS reported this discrepancy as an alert in the output log.

Differences in mutation rates between the 'Trained p-tree' and 'Test p-tree' show that the p-trees differ due to the uneven folds and distribution of SYN packets created by the SplitCap tool. Therefore, based on this observation, Experiment 2 cannot be used to definitively conclude the ADS can detect anomalies when *Combined* network traffic is introduced.

5.2   Significance of Research

This research seeks to determine the feasibility of using a PLC emulator as an anomaly detection system. The ScriptGenE framework contains features that make it attractive for use as an ADS. The framework's learning and replay abilities demonstrate that it could be implemented in an ICS environment without any prior knowledge of the system. Much research has been accomplished in applying machine learning to anomaly detection. Given the unique challenges presented by ICS and SCADA systems, the possibility for an intrusion detection system to be implemented into a system and function quickly is a research and operational goal by many security professionals.

5.3   Future Work

This section provides three recommendations to further this research: enhance algorithms, protocol expansion and honeypot incorporation.

5.3.1 Enhancing current algorithms.

The current implementation of ScriptGenE does not handle inter-protocol dependencies. Inter-protocol dependencies involve handling conversations that span across different TCP sessions. Currently, the algorithm returns an error message that states it cannot handle inter-protocol dependencies when it detects conversations across several TCP connections [LDM06]. The addition of this feature would allow for more realistic network traffic to be captured and used for protocol trees and emulating normal network behavior.

5.3.2 Testing.

Testing of the ADS was conducted in a closed world with network traffic that was generated between two devices in a controlled environment. The protocol under study is EtherNet/IP, although it is recommended that more industrial protocols be studied to test the limitations of the algorithm as an ADS. Performing the experiments in a closed environment is not realistic when the ScriptGenE begins to improve in capability. Future work to improve the ADS's performance should include using more malicious and anomalous traffic samples.

5.3.3 Hybrid network intrusion.

A more complete solution should be examined. That is, an examination of the ADS alongside a honeypot would be more interesting and offer a more complete solution to the intrusion detection problem in ICS. Referring to Section 2.2.4, ScriptGenE has the capability of fulfilling both the honeypot and the ADS. Implementation of a version of a shadow honeypot with the ScriptGenE framework offers a unique solution that could greatly improve ICS security.

5.4 Summary

Clustering algorithms and machine learning have proven useful in an ADS. Other methods, such as the novel ScriptGenE, can provide a low-cost solution for anomaly detection in an ICS environment. This research investigates the feasibility of the ScriptGenE framework, originally developed as a PLC emulator, as an ADS. Sophisticated clustering algorithms and protocol tree generation make the ScriptGenE framework an attractive solution to many security vulnerabilities in the ICS community. This research shows that the ScriptGenE framework is a viable option as an ADS as it correctly classifies anomalous network traffic from normal network traffic when trained on attack-free network traffic.

Bibliography

ASA⁺05. K.G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A.D. Keromytis. Detecting targeted attacks using shadow honeypots. In *14th USENIX Security Symposium*, pages 1–16, Baltimore, MD, August 2005. Retrieved 3 March, 2015 from *https://www.usenix.org/legacy/publications/library/proceedings/sec05/tech/full_papers/anagnostakis/anagnostakis_html/index.html*.

BTR13. C. Bronk and E. Tikk-Ringas. Hack or attack? Working Paper, February 2013. Retrieved 15 May, 2015 from *http://bakerinstitute.org/media/files/Research/dd3345ce/ITP-pub-WorkingPaper-ShamoonCyberConflict-020113.pdf*.

Cen. U.S. and world population clock, 2015. World Wide Web Page. Retrieved 5 Jun, 15 from *http://census.gov/popclock/*.

CKBR06. G. Carl, G. Kesidis, R. Brooks, and S. Rai. Denial-of-service attack-detection techniques. *IEEE Internet Computing*, 10:82–89, 2006.

CMA03. P. Chan, M. Mahoney, and M. Arshad. A machine learning approach to anomaly detection. Technical Report CS-2003-06, Florida Institute of Technology, 2003. Retrieved 15 February, 2015 from *https://repository.lib.fit.edu/handle/11141/114*.

Dem13. Defending the nation at network speed: A discussion on cybersecurity with general martin e. dempsey, u.s. army. Transcript, June 2013. Retrieved 6 September, 2014 *http://www.brookings.edu/events/2013/06/27-defense-cybersecurity-dempsey*.

DHS14. ICS-CERT year in review, 2014. Technical report, 2014. Retrieved 8 February, 2015 *https://ics-cert.us-cert.gov/sites/default/files/documents/YearinReviewFY2014Final.pdf*.

Dom12. P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55:78–87, October 2012. Retrieved 11 October, 2014 from *https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf*.

DOS06. S. DAntonio, F. Oliviero, and R. Setola. Intrusion detection in support of critical infrastructure protection. *Critical Information Infrastructures Security*, 4347:222–234, 2006. Retrieved 14 October, 2014 from *http://www.academia.edu/8018768/Intrusion_Detection_for_Critical_Infrastructure_Protection*.

Dun13. S. Dunlap. Timing-based side channel analysis for anomaly detection in industrial control system environment. Master's thesis, Air Force Institute of Technology, 2013.

FMC11.   N. Falliere, L. Murchu, and E. Chien. W32.stuxnet dossier. White Paper, 2011. Retrieved 5 August, 2014 from *https://www.symantec. com/content/en/us/enterprise/media/security_response/whitepapers/ w32_stuxnet_dossier.pdf*.

GUZ11.   I. Garitano, R. Uribeetxeberria, and U. Zurutuza. A review of SCADA anomaly detection systems. In *Soft Computing Models in Industrial and Environmental Applications, 6th International Conference SOCO 2011*, pages 1–10, Salamanca, Spain, April 2011. Springer-Verlag Berlin Heidelberg.

Hgv06.   Anomaly-based intrusion detection. Patent, February 2006. Retrieved 12 February, 2015 at *http://www.google.com/patents/US20060034305*.

Hoc15.   Raimund Hocke. SikuliX powered by RaiMan, 2015. Retrieved 9 May, 2015 from *http://www.sikulix.com*.

JY13.   J. Jiang and L. Yasakethu. Anomaly detection via one class SVM for protection of SCADA systems. In *2013 International Conference on Cyber-enabled distributed computing and knowledge discovery*, pages 82–88, Beijing, China, October 2013.

Koh95.   R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, pages 1137–1143, Quebec, Canada, August 1995. Retrieved 5 June, 2015 from *http://web.cs.iastate.edu/~jtian/cs573/ Papers/Kohavi-IJCAI-95.pdf*.

LDM06.   C. Leita, M. Dacier, and F. Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots. In *Recent Advances in Intrusion Detection, Proceedings of the 9th International Conference on RAID 06*, pages 185–206, Hamburg, Germany, September 2006. Springer Berlin Heidelberg. Retrieved 15 April, 2015 from *http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid= 42A7EEFBDBED56EB3803CC89A7FE3F8F?doi=10.1.1.453.8186& rep=rep1&type=pdf*.

LMD05.   C. Leita, K. Mermoud, and M. Dacier. ScriptGen: an automated script generation tool for honeyd. In *Computer Security Applications Conference, 21st Annual*, pages 202–214, Tucson, Arizona, December 2005. IEEE. Retrieved 15 April, 2015 from *http://citeseerx.ist.psu.edu/viewdoc/ summary?doi=10.1.1.330.3033*.

Men15.   J. Menn. Destructive hacking attempts target critical infrastructure in Americas: survey. World Wide Web Page, April 2015. Retrieved 3 May, 2015 from

http://www.reuters.com/article/2015/04/08/us-cybersecurity-americas-idUSKBN0MY06Z20150408.

MJ14.   L. Maglaras and J. Jiang. Intrusion detection in SCADA systems using machine learning techniques. In *Science and Information Conference*, pages 626–631, London, United Kingdom, August 2014. IEEE.

PS10.   V. Paxson and R. Sommer. Outside the closed world: on using machine learning for network intrusion detection. In *Security and Privacy, 2010 IEEE Symposium*, pages 305–316, Oakland, California, May 2010. IEEE.

RB15.   B. Radvanovsky and J. Brodsky. Project SHINE: What we discovered and why you should care. In *10th SANS ICS Security Summit*, pages 1–23, Orlando, FL, February 2015. SANS Institute. Retrieved 20 June, 2015 from *https://files.sans.org/summit/ics2015/PDFs/Project_SHINE_What_We_Discovered_and_Why_You_Should_Care_Bob_Radvanovsky_Infracritical.pdf*.

RGH07.   P. Ralston, J. Graham, and J. Hieb. Cyber security risk assessment for SCADA and DCS networks. *ISA Transactions*, 46(4):583–594, 2007.

Roc00.   Rockwell Automation. *Rockwell Automation RSLinx Setup Guide*, November 2000. Retrieved 10 April, 2015 from *http://controlwiki.com/images/1/1a/Rslinx$_s$etup.pdf*.

Roc14.   Rockwell Automation. *Rockwell Automation RSLinx Setup Guide*, July 2014. Retrieved 10 April, 2015 from *http://literature.rockwellautomation.com/idc/groups/literature/documents/gr/linx-gr001_en − e.pdf*.

San09.   S. Sanfilippo. Hping wiki. World Wide Web Page, September 2009. Retrieved 18 June, 2015 from *http://wiki.hping.org*.

Sch14.   D. Schwen. Hackers could take control of your car. World Wide Web Page, July 2014. Retrieved 8 May, 2015 from *http://www.wired.com/2014/07/car-hacker/*.

SFS11.   Guide to Industrial Control Systems Security: Supervisory Control and Data Acquisition systems, Distributed Control Systems (DCS), and other control system configurations, such as Programmable Logic Controllers. Technical Report SP 800-82, 2011. Retrieved 13 April, 2015 from *http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf*.

Sha06.   W. Shaw. *Cybersecurity for SCADA systems*. PennWell Corp, Tulsa, Oklahoma, 2006.

Spl.   Splitcap. World Wide Web Page. Retrieved 10 June, 2015 *http://www.netresec.com/?page=SplitCap*.

SS12.   N. Sharma and G. Singh. Intrusion detection system using shadow honeypots. *International Journal of Emerging Technology and Advanced Engineering*, 2:498–500, August 2012. Retrieved 3 March, 2015 from *http://www.ijetae.com/files/Volume2Issue8/IJETAE_0812_84.pdf*.

Swe88.   J. Swets. Measuring the accuracy of diagnostic systems. *American Association for the Advancement of Science*, 240:1285–1293, June 1988. Retrieved 11 June, 2015 from *http://www.sciencemag.org/content/240/4857/1285.short*.

VCD⁺07. A. Valdes, S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, and K. Skinner. Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium*, pages 1–12, Miami, Florida, January 2007. Retrieved 22 October, 2015 from *http://www.csl.sri.com/users/cheung/SCADA-IDS-S4-2007.pdf*.

War15.   P. Warner. Automatic configuration of programmable logic controller emulators. Master's thesis, Air Force Institute of Technology, 2015.

Wer14.   J. Werling. Behavioral profiling of SCADA network traffic using machine learning algorithms. Master's thesis, Air Force Institute of Technology, 2014. Retrieved 5 October, 2014 from *http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA610906*.

YJ13.   S. Yasakethu and J. Jiang. Intrusion detection via machine learning for SCADA system protection. In *Proceedings of the 1st International Symposium for ICS and SCADA Cyber Security Research*, pages 101–105, Leicester, Englad, September 2013. Retrieved 18 October, 2014 from *http://ewic.bcs.org/upload/pdf/ewic_icscsr13_paper12.pdf*.

YUH06.   D. Yang, A. Usynin, and J. Hines. Anomaly-based intrusion detection for SCADA systems. In *5th International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human Machine Interface Technologies*, pages 12–16. American Nuclear Society, 2006. Retrieved 15 February, 2015 from *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.102.1649&rep=rep1&type=pdf*.

Zan04.   S. Zanero. Detecting 0-day attacks with learning intrusion detection system. In *Black Hat 2004*, pages 1–31, Amsterdam, Netherlands, May 2004. Retrieved 1 June, 2015 from *https://www.blackhat.com/presentations/bh-europe-04/bh-eu-04-zanero-up.pdf*.

ZS10.   B. Zhu and S. Sastry. SCADA-specific intrusion detection/prevention systems: a survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Systems: A Survey and Taxonomy*, pages 1–16, Stockholm, Sweden, April 2010. Retrieved 6 May, 2015 from *http://www.cse.psu.edu/~sem284/cse598e-f11/papers/zhu.pdf*.

ZZLJ08.   R. Zhang, S. Zhang, Y. Lan, and J. Jiang. Network anomaly detection using one class support vector machine. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, pages 1–5, Hong Kong, China, March 2008. Retrieved 15 May, 2015 from *citeseerx.ist.psu. edu/viewdoc/summary?doi=10.1.1.148.4756*.

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
17–09–2015	Master's Thesis	May 2013 — Sep 2015

4. TITLE AND SUBTITLE

A Feasibility Study on the Application of the ScriptGenE Framework as an Anomaly Detection System in Industrial Control Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Corvin, Charito, M., Capt, USAF

5d. PROJECT NUMBER

15G264

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-15-S-010

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of Homeland Security ICS-CERT POC: Neil Hershfield, DHS ICS-CERT Technical Lead ATTN: NPPD/CS&C/NCSD/US-CERT Mailstop: 0635, 245 Murray Lane, SW, Bldg 410, Washington, DC 20528 Email: ics-cert@dhs.gov phone: 1-877-776-7585

10. SPONSOR/MONITOR'S ACRONYM(S)

DHS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

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

14. ABSTRACT

Recent events such as Stuxnet and the Shamoon Aramco have brought to light how vulnerable industrial control systems (ICSs) are to cyber attacks. Modern society relies heavily on critical infrastructure, including the electric power grid, water treatment facilities, and nuclear energy plants. Malicious attempts to disrupt, destroy and disable such systems can have devastating effects on a populations way of life, possibly leading to loss of life. The need to implement security controls in the ICS environment is more vital than ever. ICSs were not originally designed with network security in mind. Today, intrusion detection systems are employed to detect attacks that penetrate the ICS network. This research proposes the use of a novel algorithm known as the ScriptGenE framework as an anomaly-based intrusion detection system. The anomaly detection system (ADS) is implemented between an engineering workstation and programmable logic controller to monitor traffic and alert the operator to anomalous behavior. The ADS achieves true positive rates of 0.9011 and 1.00 with false positive rates of 0 and 0.054. This research demonstrates the viability of using the ScriptGenE framework as an anomaly detection system in a simulated ICS environment.

15. SUBJECT TERMS

ICS, SCADA, anomaly detection

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Barry E. Mullins (ENG)
U	U	U	U	81	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x7979 Barry.Mullins@afit.edu