3-2021

Anomaly Detection and Encrypted Programming Forensics for Automation Controllers

Robert W. Mellish

Anomaly Detection and Encrypted Programming Forensics for Automation Controllers

THESIS

Robert W. Mellish, First Lieutenant, USAF

AFIT-ENG-MS-20-M-060

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-20-M-060

ANOMALY DETECTION AND ENCRYPTED PROGRAMMING FORENSICS FOR AUTOMATION CONTROLLERS

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 Computer Engineering

Robert W. Mellish, B.S.
First Lieutenant, USAF

25 March 2021

AFIT-ENG-MS-20-M-060

ANOMALY DETECTION AND ENCRYPTED PROGRAMMING FORENSICS FOR AUTOMATION CONTROLLERS

THESIS

Robert W. Mellish, B.S.
First Lieutenant, USAF

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Stephen J. Dunlap, M.S.
Member

Lt Col Patrick J. Sweeney, Ph.D.
Member

AFIT-ENG-MS-20-M-060

Abstract

Securing the critical infrastructure of the United States is of utmost importance in ensuring the security of the nation. To secure this complex system a structured approach such as the NIST Cybersecurity framework is used, but systems are only as secure as the sum of their parts. Understanding the capabilities of the individual devices that make up the system, developing tools to help detect misoperations, and providing forensic evidence for incidence response are all essential to mitigating risk. This thesis examines the SEL-3505 Real Time Automation Controller to demonstrate the importance of existing security capabilities as well as creating new processes and tools to support the National Institute of Standards Cybersecurity Framework of Identify, Protect, Detect, Respond, and Recover.

The research examines the potential pitfalls of having small-form factor devices in poorly secured and geographically disparate locations. Additionally, the research builds a data-collection framework to provide a proof of concept anomaly detection system for detecting network intrusions by recognizing the change in task time distribution. This framework uses Python to collect data from a modbus server on the target device and perform statistical tests to distinguish between normal and anomalous behaviour. The high true positive rates and low false positive rates show the merit of such an anomaly detection system. Finally, the work presents a network forensic process for recreating control logic from encrypted programming traffic.

AFIT-ENG-MS-20-M-060

"In God we trust. All others must bring data." - W. Edwards Denning

Acknowledgements

I would first like to thank the tireless work of my research advisor Dr. Scott Graham for his mentorship and advice that extends far beyond this document but will help shape the rest of my academic and professional career.

I would also like to acknowledge Mr. Stephen Dunlap whose technical expertise was instrumental in shaping the understanding of the little box I have had sitting on my desk for the last 18 months.

Finally I would like to thank Lt Col Sweeney who is providing bookends to my AFIT experience serving as my first professor and final member of my committee.

Robert W. Mellish

Table of Contents

Page

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ADS	Anomaly Detection System
ARP	Address Resolution Protocol
AUC	Area Under the Curve
DCS	Distributed Control System
DoS	Denial of Service
FNR	False Negative Rate
FPR	False Positive Rate
HMI	Human Machine Interface
ICS	Industrial Control System
IDS	intrusion detection system
IP	Internet Protocol
IT	Information Technology
KS Test	Kolmogorov-Smirnov Test
MAC	Media Access Control
MITM	Man-in-the-Middle
NIST	National Institute of Standards and Technology
OT	Operational Technology
PLC	Programmable Logic Controller
ROC	Receiver Operating Characteristic
RTAC	Real Time Automation Controller
SCADA	Supervisory Control and Data Acquisition
SEL	Schweitzer Engineering Laboratories
TNR	True Negative Rate
TPR	True Positive Rate

ANOMALY DETECTION AND ENCRYPTED PROGRAMMING FORENSICS FOR AUTOMATION CONTROLLERS

I. Introduction

1.1 Background and Motivation

Underpinning modern society are various infrastructures with processes and systems that ensure smooth operation. If these "assets, systems, and networks, whether physical or virtual, are considered so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof" they are designated as Critical Infrastructure [1]. There are sixteen sectors designated by Presidential Directive 21 [2] including such vital services such as the energy and water sector. Many of these important sectors utilize Industrial Control System (ICS) which includes Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control System (DCS), or standalone Programmable Logic Controllers (PLCs) [3]. The resilience of these ICSs against a wide range of intrusions and adverse operating conditions is paramount to the security of the society that relies on them.

1.2 Problem Statement

Originally intended as standalone and isolated systems, many ICS devices were not designed with security in mind. Additionally while typical Information Technology (IT) networks have been the focus of cybersecurity professionals, Operational Technology (OT) networks languished, relying on security through obscurity rather than a robust set of security controls and processes. While IT networks benefit from commodity hardware and ubiquitous protocols ICS has long relied on specialized hardware and proprietary protocols that are long lived, ill documented, and difficult to upgrade. Now as ICS is incorporating Internet Protocol (IP) solutions to improve remote access and overall connection with corporate networks, the risk for adversarial access has increased substantially [3].

The security of the system relies on overarching processes as well as the integrity of individual devices. For this reason the U.S. has begun regulating from whom the largely private owners of critical infrastructure procure their devices [4]. This fear of a supply chain attack has recently been realized with the ongoing investigations into the recent Solarwinds exploit [5]. This is the latest in a line of high profile attacks against U.S. organizations such as the Russian targeting of the U.S energy sector in 2018 [6]. When a U.S. citizen reaches for the light switch, there is currently no doubt in their mind that the lamp will turn on, but the resilience of this service may be more precarious than perceived. Behind the power switch is a patchwork protection and control devices. This research will examine the SEL-3505 Real Time Automation Controller (RTAC), a flexible device representative of a family of power automation controllers used in critical infrastructure sectors.

1.3 Research Objectives

The research objectives of this work are outlined below:

- Identify the operation and attack surfaces of Automation Controllers within ICS.
- Explore available intrusion detection system (IDS) tools for Automation Controllers and gaps for future development.

- Develop a process for forensic artifact retrieval for Automation Controllers.
- Implement a data-collection framework for Anomaly Detection System (ADS) experimentation.
- Evaluate potential ADS algorithms to detect network intrusions using collected data.
- Assess the developed ADS, identify shortcomings and future improvements of the application.

The questions to be answered by this research in order to meet the aforementioned objectives are as follows:

- Are Industrial Automation Controllers secure?
- Can an ADS detect network intrusions using available device characteristics?
- Can application control logic be reconstructed by an attacker, without decrypting the encrypted programming traffic?

1.4 Hypothesis

The hypothesis of this research is that Industrial Automation Controllers are vulnerable to some forms of network intrusions, such as Denial of Service (DoS) attacks or ARP Spoofing. These intrusions can be detected by measuring the changes in device characteristics caused by the intrusions. Additionally, illicit access to the device using compromised credentials can be detected.

1.5 Approach

This research was conducted by first performing a penetration test on the target device using both manual and automated tools. With a strong grasp of the strengths and limitations of the current security posture, recommendations are formed. These findings informed the creation of the ADS. The ADS itself relies solely on the RTAC itself and a Windows workstation capable of running python connected over a local network. To perform some of the network intrusions, a third device is connected to the network so that its Media Access Control (MAC) address can be resolved and Address Resolution Protocol (ARP) spoofing conducted. Python is used for both the data collection and the data analysis with the utilization of various readily available packages.

1.6 Contributions

The contributions of this thesis to the field of ICS cybersecurity include the following:

- **Physical Vulnerability:** Demonstrated straightforward compromise of end devices through physical access.
- **Forensic Process:** Developed and demonstrated a network forensic process for reconstructing control logic from encrypted programming traffic.
- **Data Collection Tools:** Created a series of scripts to ease the setup of a data collection framework to test and evaluate an ADS against numerous network intrusion and project scenarios.
- **Qualitative Analysis:** Presented strong evidence that Task Time can be used to detect the additional burden on end devices caused by network intrusions.

1.7 Organization

This thesis is organized as follows:

Chapter II introduces the ICS security and ADS concepts and nomenclature. The NIST Cybersecurity Framework is discussed and its five functions are used to frame the following sections. It defines the terminology utilized throughout this thesis as it relates to an ADS. It provides an analysis of the device of interest for the research, a SEL-3505 RTAC seeking to perform the Identify function of the NIST Framework. The motivation for the included processes is discussed and related research is explored.

Chapter III focuses on the Protect function of the NIST framework. It demonstrates the vulnerabilities of having physical access to a device and explains how an attacker with physical access can do more than perform a DoS attack by bringing the device offline. It provides several recommendations and best practices to mitigate existing gaps in the ability to protect the RTAC

Chapter IV presents the creation of a forensic process for encrypted programming traffic using the RTAC as a case study. It describes three different potential implementations to protect programming traffic while still allowing forensic auditing. This process is part of the Respond function of the framework and is essential part of the analysis of incident response.

Chapters V and VI build on existing security capabilities to bring a robust Detect function to the RTAC. Chapter V begins by describing the developed ADS data collection framework and establishing how candidate algorithms will be evaluated.

Chapter VI presents an analysis of the collected ADS data. It compares the performance of several algorithms against the collected data set. It starts by using the entire data set and then tests the efficacy of modifying the data set before providing it to the algorithm. This includes exploring the effects of varying sample size as well as the elimination of outliers by only providing certain percentiles to the algorithms. Additionally, continuous evaluation vs discrete evaluation is compared.

Chapter VII concludes with a summary of the work presented and the contributions to the field. In addition, recommendations for those utilizing similar tools or frameworks are presented. Areas for future work are highlighted.

II. Background and Related Work

2.1 Overview

This chapter provides background information and knowledge about Industrial Control System (ICS) security including an overview of the National Institute of Standards and Technology (NIST) Cybersecurity framework. It then provides contextual information about the SEL-3505 as well as its manufacturer Schweitzer Engineering Laboratories. Finally, it provides related work subsections for both network forensic processes and Anomaly Detection Systems (ADSs).

2.2 NIST Cybersecurity Framework

Cybersecurity frameworks provide a defined process to manage and mitigate cybersecurity risk. The NIST framework was developed in accordance with the Cybersecurity Enhancement Act of 2014 to identify "a prioritized, flexible, repeatable, performance based, and cost-effective approach, including information security measures and controls that may be voluntarily adopted by owners and operators of critical infrastructure to help them identify, assess, and manage cyber risks" [7]. While targeted at critical infrastructure, due to its flexibility, it can be employed in any sector. The "Framework Core" consists of five cybersecurity functions, each of these is listed below with the provided NIST definition [7]:

- **Identify**: "Develop the organizational understanding to manage cybersecurity risk to systems, assets, data, and capabilities."
- **Protect**: "Develop and implement the appropriate safeguards to ensure delivery of critical infrastructure services."

- **Detect**: "Develop and implement the appropriate activities to identify the occurrence of a cybersecurity event."
- **Respond**: "Develop and implement the appropriate activities to take action regarding a detected cybersecurity event."
- **Recover**: "Develop and implement the appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity event."

Each of these core functions contains numerous categories and subcategories along with informative references to help stakeholders achieve the outcomes associated with each function.

Implementing the entirety is a process that can take a cross-functional team, thousands of man hours, and hundreds of pages of documentation. This research does not seek to apply the entirety of the process but ensure that an individual device has the necessary functionality to be successfully integrated into a complete system. The framework itself is a living document and its guidance constantly evolves. Moreover, the application of the framework is a continuous process to ensure the system of interest continues to be in compliance with the evolving threat landscape.

2.3 The Power System

The device of interest in this research is typically thought of in a power system perspective. This section answers the question 'What is a power system?'.

A power system is "a network of components designed to efficiently transmit and distribute the energy produced by generators to the locations where it is used" [8]. This network includes numerous components such as:

- Generators

- Transformers
- Power Lines
- Loads
- Protective Devices
- Control Devices
- Measurement Devices

There are two main failure modes of a power system, overloads and faults. Overloads occur when a component is supplying more electrical power than it is rated to safely handle. Most electrical components can temporarily handle an overload condition giving operators some time to correct the issue [8]. Fault conditions occur when power lines are shorted to ground or another line. As electrical current is equal to the difference in voltage divided by resistance and a short has very low resistance, faults must be cleared immediately to prevent damage caused by the large currents. Opposed to failures which are caused by external factors, misoperations are “The failure of a Composite Protection System to operate as intended for protection purposes.” The North American Electric Reliability Corporation gives 6 categories for misoperations [9]:

- Failure to Trip - During Fault
- Failure to Trip - Other Than Fault
- Slow Trip - During Fault
- Slow Trip - Other Than Fault
- Unnecessary Trip - During Fault

- Unnecessary Trip - Other Than Fault

The leading cause of these misoperations in the power system is due to incorrect logic on the protective relays [10]. However, misoperations could be caused by cyberattacks. DoS attacks may prevent devices from taking necessary actions or compromised devices may cause unnecessary trips. An example of a misoperation of a power system is the Aurora attack in which researchers were able to cause severe physical damage to an electric power generator [11].

Instead of looking at power systems as a whole, this research focuses on examining a specific device and ensuring that tools exist to conduct all functions. This process begins with the Identify function, understanding the devices in the network.

2.4 Schweitzer Engineering Laboratories' RTAC

Schweitzer Engineering Laboratories (SEL) is a United States based manufacturer of power protection and automation equipment. As an employee owned company, the number of devices they have sold and installed is not part of publicly available information. However, viewing the success stories that the company publishes online shows SEL devices protecting the power systems of the countries of Georgia and Grand Cayman, controlling Microgrids at American Universities, mitigating arc-flashes for North American Mining Companies, and managing the power for Oil Refineries [12]. Additionally, a Newton-Evans Research Company study of the worldwide Protective Relay Marketplace has put SEL as the number one ranked relay in the North American marketplace for the last decade [13]. Although an exact number is elusive, SEL devices are clearly prevalent throughout critical infrastructures, including those of the United States.

The device of interest for this research is the SEL-3505 RTAC. Marketed as a substation controller, the RTAC combines physical I/O with flexible IEC-61331 logic along with numerous serial ports and dual Ethernet ports. In addition to its physical capabilities SEL includes numerous communication libraries that allow the RTAC to 'speak' with countless devices. This allows the RTAC to be used as a data concentrator, communicating with multiple legacy devices over serial protocols and converting the data streams into Ethernet-based communication. This combination of features makes the RTAC both a powerful tool for a system operator and a likely potential target for network attackers as its compromise could cede control of all devices it communicates with.

The RTAC consists of a embedded Linux host with several applications running to act as a PLC. It runs a web server for configuration and management. From the password protected web page, user accounts can be created and diagnostics can be run. These diagnostics include checking logic status and performing factory resets if necessary. The web interface uses a PostgreSQL backend that contains numerous database functions. to download new control logic project files and make changes to the device firewall.

Project files are created using a Windows-based engineering software called AcSELerator RTAC. Control logic programming can be conducted in several of the IEC-61131-3 programming languages including structured text and function block diagrams. While undocumented, this software creates a compiled binary that is run by a widely used PLC framework, CODESYS [14]. The CODESYS runtime is the IEC-61331 logic engine and is what enables the Linux host to act as a generic PLC.

When viewed through the lens of the NIST Cybersecurity functions, the discovery of these three key interfaces perform part of the Identify function. Understanding the requirements of the web server, CODESYS runtime, and PostgreSQL database is crucial to the security of the device and the network in which it resides. The omission of the use of CODESYS in any RTAC documentation prevents product owners from fully understanding risks related to their devices.

The RTAC is designed with several robust security features. Among these features are: an application whitelisting solution that “provides protection against rootkits”, built-in Denial of Service (DoS) detection and system priority readjustment, and functionality that allows all Sequence of Events configured data tags to be available in a syslog client [15, 16]. While DoS detection is able to respond to brute force attacks on the network stack and whitelisting prevents nonauthorized processes from running, there is currently no detection strategy for the misuse or subtle exploitation of an authorized application such as an unauthorized connection to the CODESYS runtime or exploitation of the database. To fill the gap, this research proposes the addition of an ADS for anomaly detection and a forensic process for encrypted programming traffic. These processes could be employed across all generic PLCs providing end device anomaly detection and incident response in any ICS network. The creation of this ADS helps fulfill the Detect function of the framework and is explored in Chapter V.

Additionally, the SEL-3505 has both an accelerometer and light sensor, these features help detect intruders trying to gain physical access to the device. The dangers of this are explored in Chapter III and recommendations therein help fulfill the Protect function.

One category of the Respond function is Analysis. Within the category is the subcategory, forensics. While the RTAC has secured its programming traffic by encrypting it, no current tool exists to recreate project files from the encrypted programming traffic. If a device is rendered inoperable or is factory reset by an attacker or by the system’s out-of-memory monitoring [15] there is no way to recover the offending project file from the RTAC itself. Tools to reconstruct this logic are necessary for both incident response and control logic auditing to ensure that the project being sent across the network is the same as the project present on the Engineering Workstation. Chapter IV demonstrates a process developed for this purpose and to fulfill the Respond function.

The Recovery function of the NIST framework is readily fulfilled by existing features. The ability to factory reset the device from the web interface or from physical jumpers provides sufficient capability as long as the device owner retains appropriate archives of device settings such as IP address and the project file.

2.5 Related Work

This section presents work related to both the forensic process of Chapter IV and the ADS design of Chapter V.

2.5.1 Forensic Tools

Performing forensic analysis on SCADA systems poses significant hurdles [17]. Due to the proprietary nature of ICS protocols, the development of reconstruction tools is typically a manually intensive effort requiring intimate knowledge of either the programming protocol itself for each device or the ability to reverse engineer the binary format. Senthivel, Ahmed, and Roussev [18] reverse engineered the Programmable Controller Communication Commands (PCCC) protocol used to program Allen-Bradley's Micrologix 1400 to build a tool called Cutter, which is capable of extracting a low-level representation of the control logic as well as a human readable representation of some PLC configuration files from network traffic. Senthivel et al. [19] then extended Cutter to produce Laddis. Laddis was capable of decompiling the low-level control logic representation back into a high-level representation that is human readable and programmatically modifiable. This allows for a newly coined exploit, denial of engineering operations, in which engineering software is rendered unresponsive due to a malformed network packet. With the creation of this tool, the control logic that was sent over the network for one specific PLC could be reconstructed.

While tools that can be used for singular devices are useful, a general tool that can be applied to all devices has been the object of additional research. Keliris and Maniatakos [20] built an extendable automated reverse engineering framework called ICSREF. They then evaluated the framework against binaries compiled with CODESYS v2.3, a development environment used for various manufacturers of PLCs. Nochay (2019) provides a security analysis of the CODESYS runtime stating that the official list specifies 350 devices that use CODESYS, with more being undocumented. Instead of reverse engineering the compiled binaries, another approach to retrieve the high-level representation is to utilize built-in decompilers of the proprietary engineering software. Qasim, Lopez, and Ahmed [21] followed this strategy in the creation of Similo, a "Virtual PLC-framework", that built a knowledge database of message sent and received from engineering software in order use the engineering software to recreate the high-level representations from network traffic. All of these tools can reconstruct their intended control logic for the current programming paradigm in which the control logic is sent in plain text across the network. As ICS networks become more secure, primarily by encrypting the network traffic, this paradigm will shift and the reconstruction tools themselves or the process that archives network traffic will need to adapt to handle this change. Chapter IV provides several implementation alternatives and details the use of one of them to reconstruct the RTAC control logic.

2.5.2 Detection Systems for Industrial Controllers

One standard tool for detection in IT is an IDS. These systems monitor system events to try and detect misuse or malicious activity [22]. There are two standard placements of an IDS, a Network-Based IDS and a Host-Based IDS. This placement determines the type of data that the system can be used to detect intrusions [3]. Additionally an IDS can either be signature based or anomaly based. In a signature based system the IDS attempts to match patterns of data such as network traffic with a database of known attacks. In an anomaly based system it will seek to find differences between the present behaviour and known behaviour [23]. This style of system will be referred to as an ADS to differentiate it from a signature based system. An anomaly-based approach was chosen for this research for its ability to detect unknown attacks.

At an abstract level, a Host-Based Anomaly Detection System can be distilled into a Workload, System Outputs, Tuning Parameters, and a decision strategy working with a data collector at the heart of the ADS. Each of these individual facets are examined to build a framework upon which to examine the experimental system. Figure 1 shows an overview of the components of an ADS.

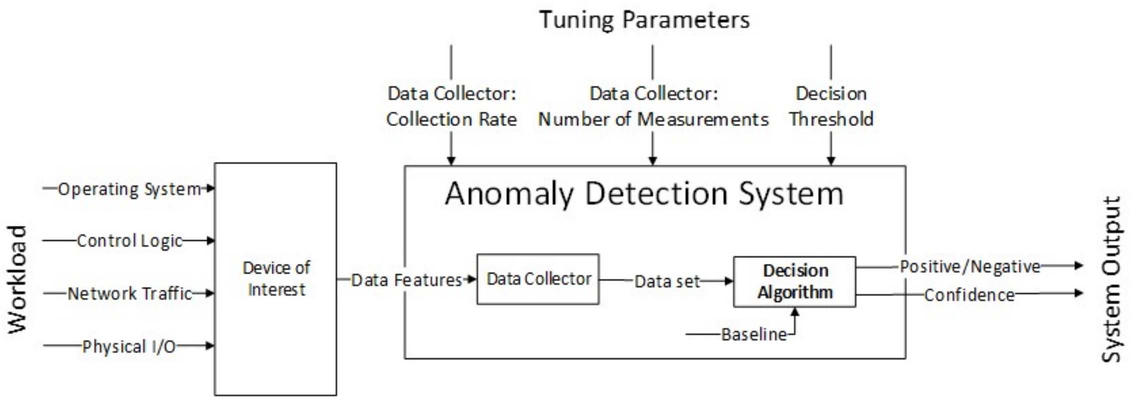


Figure 1. Notional Anomaly Detection System

2.5.2.1 Workload

The workload encompasses the actions the underlying hardware must complete on a continual basis. The hardware completing these actions causes measurable responses that can be used for anomaly detection. Some of these actions are mandated by the manufacturer and are both invisible to and unchangeable by the end user. Other actions are created by the end user to fulfill their process needs. The final category of workload includes the burden placed on the hardware by the physical process it is measuring and the network traffic it is receiving.

- Operating System Functions

  While programmable logic controllers and other industrial devices may be ideally portrayed as user logic running on bare metal, in practice there is an operating system that is handling the network stack and the communication necessary for programming the PLC. These tasks compete for resources and can cause the PLC tasks be executed more slowly even as the real time requirements are being met. Popular operating systems include OS-9, VxWorks, and Linux. Modifications to the OS, either malicious or manufacturer mandated, can cause changes to the workload. Some of these changes have been shown to be detectable by currently available ADS methods [24, 25].

- Control Logic

  The control logic is the user created functions that the PLC must continuously complete to successfully monitor and control its assigned process. The user provided control logic in complex processes can represent the bulk of the workload for the industrial device. The detection of malicious modification to this logic is the subject of numerous research endeavors [24, 25].

- Network Traffic

  The number of packets that are sent and received place a burden on the device. Some devices act as communication gateways connecting numerous devices with a single upstream device. Others simply translate the physical process information into a digital representation that can be displayed from the central HMI. Engineering functions such as programming, performing diagnostics, and retrieving system logs can also place an additional workload on the device's resources. The effect of this workload can be seen in the task time of devices under duress from network scans [26].

- Physical I/O

  PLC's and related devices are responsible for monitoring and controlling physical processes using specialized sensors and actuators. In some devices the processing of this data has been shown to cause no increase in task time [24] as inputs vary. This may be attributed to the handling of I/O possibly being offloaded to a secondary processor. In the RTAC there is an FPGA that may handle the data conditioning of the Binary and Analog I/O to prevent the CPU from needing to intervene. Due to the limitations of monitoring the FPGA, the effect of physical I/O variations will not be explored in this research.

- Network Intrusions

  Network intrusions represent one set of anomalous actions that an ADS is trying to detect. They place additional burden on the device by sending additional network traffic, running additional processes on the underlying operating system, or attempting to exfiltrate data. An ADS must discern between the normal workload and the burden of intrusions through careful selection of features and analysis tools.

2.5.2.2 System Outputs

The system outputs from an ADS are the algorithm decision and confidence level associated with the decision. To be able to compare the performance of two different anomaly detection systems a common set of performance statistics needs to be computed. Historically there have been four measures to indicate performance of an ADS [24, 23].

- True Positive Rate (TPR): The rate at which the ADS correctly identifies an anomaly.
- True Negative Rate (TNR): the rate at which the ADS correctly identifies a normally behaving system.
- False Positive Rate (FPR): the rate at which the ADS incorrectly detects a normally behaving system as an anomaly.
- False Negative Rate (FNR): the rate at which the ADS fails to identify a network intrusion.

The False Positive Rate is the type I error of the system. If this rate is too, high alarms become nuisances and will quickly be disregarded even when a true anomaly is occurring. The False Negative Rate represents the frequency at which anomalies are undetected. Keeping this rate low is the main objective of an ADS designer. While these have been the historical measures for a system there are additional factors to consider.

2.5.2.3 Detection Latency

Although not as frequently used as a measure of system performance the Detection Latency is another important aspect of the system. It measures the length of time required for the ADS to detect an anomaly. The infrequency of the inclusion of this statistic can be attributed to the need for a more mature ADS than those typically being explored in literature.

2.5.2.4 System Overhead

Industrial Devices are resource constrained and act in a hard real-time environment, therefore, a deployed ADS needs to be light-weight so as to not impact the controlled process. This overhead can be measured as an increase in task time or the increase in CPU burden percentage.

2.5.2.5 Tuning Parameters

Tuning parameters are properties of the ADS that can be changed in an attempt to increase overall system performance.

- Selection of System Features to Collect

  Depending on the user privileges of the ADS there are numerous available features to consider. These include statistics such as task time, CPU burden, the number of network packets sent and received, system RAM in use, and others. Different features may lend themselves better to detecting specific attacks and an ADS designer must carefully select, justify, and defend the chosen test statistic. Previous research has used task time [24, 25]. This research seeks to extend what intrusions can be discerned using the previously employed features.

- Data Collection Rate

  How often data is collected can directly affect the sensitivity of an ADS. If attacks are ephemeral, a polling rate that is too slow may miss the anomaly. If data polling becomes too frequent, the burden on the device becomes too great and the real-time requirements will not be met.

- Number of Data Points Collected

  The number of consecutive data points needed to make a decision in conjunction with the data collection rate determines the detection latency when the processing time of the decision algorithm is neglected.

2.5.2.6 Decision Strategy

The decision strategy is the heart of the ADS and is paramount to the overall success of the system. The decision strategy uses the collected data to make an assessment on the operation of the underlying device. Various methods have been employed in related research to detect anomalies. Depending on the data features collected different strategies have been taken. Vargas et al. monitored RAM usage using a Simple Moving Average [27]. Formby and Beyah used a Cumulative Sum algorithm on task time to detect logic changes [25]. Dunlap et al. also looked at task time but used the Permutation Test to discern changes [24]. Alves et al. used a embedded machine learning IDS in conjunction with OpenPLC to detect network anomalies by inspecting TCP headers [28]. In order to make a portable solution that could be readily implemented in IEC-61131 logic this research focused on statistical methods rather than machine learning.

III. Protect: The Dangers of Physical Access

This section briefly provides an example of the importance of physical controls preventing access to ICS devices as part of the Protect Function. Typical recommendations for ICS security restrict physical access to devices through the use of locks, card readers and guards [3]. These traditional methods work well when there is a large amount of centralization in both logical control and device placement. However, the SEL-3505 is not designed to be be deployed as a control center device. Its small form factor, shown in Figure 3 and 2, and surface mount rather than rack mount design allow it to be deployed in space-limited locations such as recloser enclosures, weather-proof boxes that contain hardware for clearing momentary faults on distribution lines that are placed throughout a power system. SEL markets this flexibility in location for the power and even irrigation sectors [29, 30].

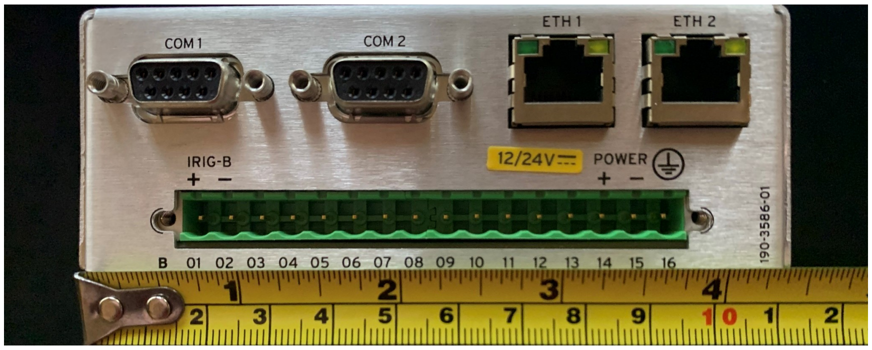


Figure 2. Back View of RTAC

This flexibility places these sensitive devices on power distribution poles throughout a power system or in small remote enclosures and not just within the confines of substations that have also been shown to be insecure [31]. This adds risks similar to those seen in wind farm installations [32]. While physical access allows an attacker to easily disrupt the functional control of devices downstream from the RTAC by destroying the device or disconnecting power, more insidious attacks can occur if the

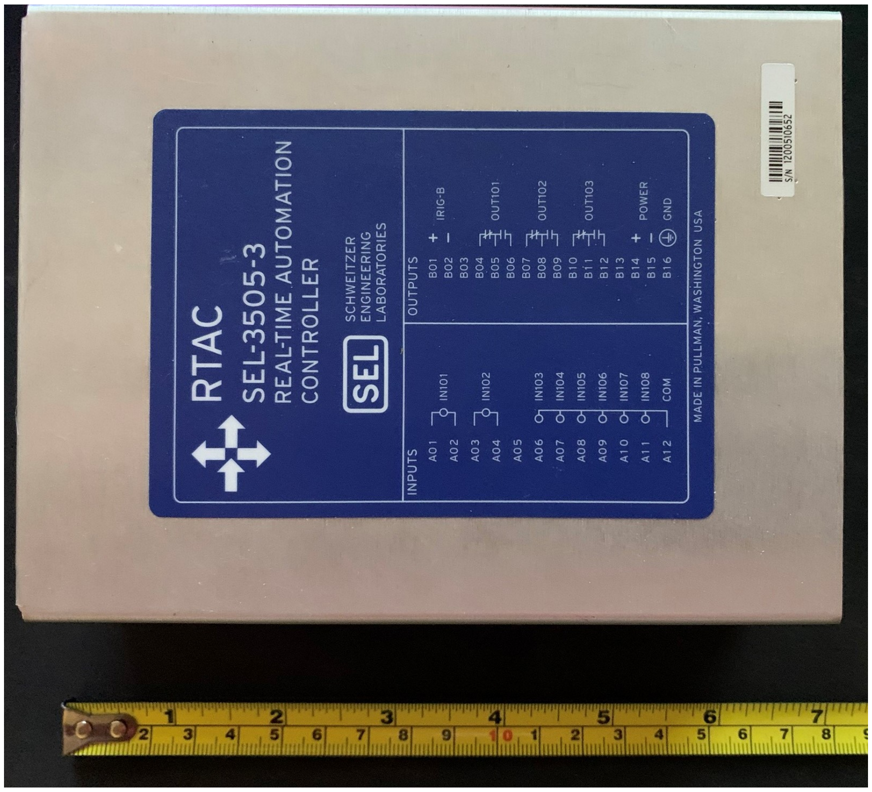

RTAC
SEL-3505-3
REAL-TIME AUTOMATION CONTROLLER
SEL
SCHWEITZER ENGINEERING LABORATORIES
INPUTS
OUTPUTS
MADE IN PULLMAN, WASHINGTON USA

Figure 3. Top View of RTAC

device is unknowingly compromised and remains on the network.

3.1 Disabling the Password

Located near the front of the RTAC lies a jumper block with five sets of pins that can be connected by shunts or jumpers. Of the 5 pairs, two have documented uses [15]. Pair A can be used used to factory reset the RTAC by first powering off the device, jumping the pair, and powering on the device. Pair C is used similarly but instead of resetting the device, it simply disables the need for password authentication. It also activates a default admin account so that even if all usernames have been forgotten the device can be recovered. These functions act as conveniences for control engineers and removes the need for network backdoors. However, the presence of these physical bypasses for authentication in remote devices poses some risk.

While the manufacturer recommends that the device be powered down before connecting the password jumper pins, most likely to prevent electrical shorts while performing the action, the password pin connection is tested continuously rather than merely at startup. This can be verified by powering on the device with the top of its case removed, accessing the web interface and monitoring the Sequence of Events as the password jumper is added and removed. The password disabled state is logged as a device level alarm by default. This gives a skilled and determined attacker the ability disable the password and gain administrative access to the device. This in-place intrusion avoids the difficulty of bringing the device offline, removing various wiring and connections, removing the top cover, and then rewiring the device once access had been gained and the top cover has been replaced. The ability to bypass authentication of the device inplace greatly reduces to the time to exploit and may decrease the ability for the system operator to send a response team. This places both the data on the device and the process it controls at risk.

3.2 In-place Intrusion

To verify this possible attack vector, the intrusion tool shown in Figure 4 was created using a bent paperclip.

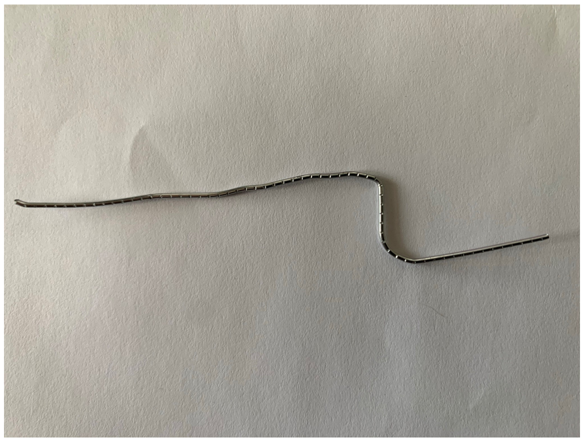


Figure 4. Intrusion Tool

It was then used to connect the password jumper with the top cover removed to ensure that a proper connection could be made with such a rudimentary tool. Figure 5 demonstrates the placement of the simple hook stretching over the front of the device and contacting the correct pins. The password disabled alarm was observed each time the paperclip was placed across the pins.

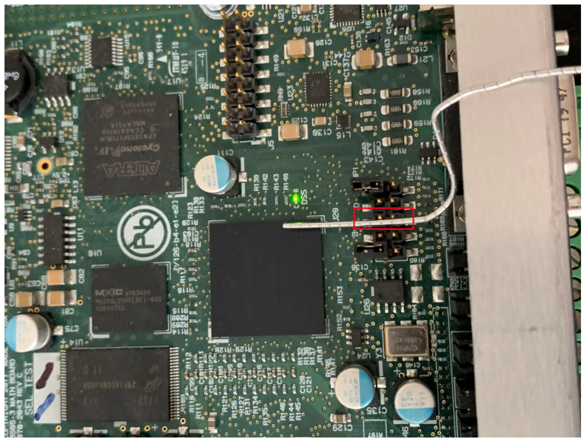


Figure 5. Password Jumper Connected Using Paper Clip

As the paperclip was shown to be a viable candidate the top cover was replaced and the RTAC was powered on. A flat-head screwdriver was used to pry the cover away from the body of the RTAC and the tool was inserted and then rotated to properly align it with the jumper pins. This action is depicted in Figure 6. These steps allowed for the bypass of authentication without any power cycling or removal of any wiring and the web interface was able to be accessed. This methodology might also be applicable to other devices in the RTAC family such as the SEL-3530 and SEL-3555 or a device that appears to share the same hardware platform, the SEL-3622 Security Gateway.

This method is not without fear of detection. A mistaken connection between the grounded case and other conductive material on the circuit board can cause an electrical short and cause the RTAC to restart for the safety of its hardware. This reboot incurs a three and a half minute wait while the device powers up before the

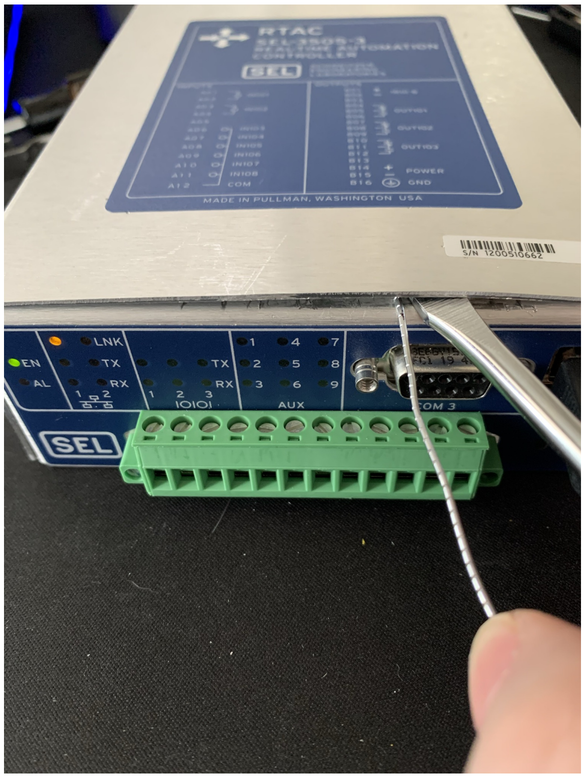


Figure 6. Connecting Password Jumper Without Opening RTAC

intrusion can be attempted again.

3.3 Abusing Administrator Access

Once the password has been disabled, the attacker has full admin privileges to the device. This can be used to add/remove/modify device accounts. This can restrict system owner access without performing a factory reset or password deactivation of their own. Additionally, the control logic program could be retrieved, allowing upstream data to be modified according to the attacker's desire, to create control system misoperations. Similarly, downstream commands could be interrupted, or maliciously generated, to produce selective disruptions of service.

If the attacker was then able to elevate their PostgreSQL user to a superuser on the database, specific database functions necessary to the correct operation of the

RTAC could be modified or completely removed. If this superuser status was used to run system level commands pivoting throughout the network using the RTAC may be possible. These actions should be limited by the RTAC's whitelisting solution but some system level commands such as pinging hosts to try and map out the rest of the network are allowed. The database superuser could also retrieve the list of all users present on the device and the hash of their passwords, providing an additional reconnaissance option to the attacker.

3.4 Recommendations

Physical protections are not always able to perfectly protect all devices, especially those that are distributed in close proximity to the processes they control. To help bolster the Protect function of the NIST Framework for the RTAC the following recommendations are made:

- Deactivate the front USB port unless remote maintenance is being conducted.
- Deactivate any unused Ethernet ports
- Place a non-conductive cover over the jumper pins preventing a connection being made without its removal.
- Ensure that all device level alarms are transmitted to a central monitoring system including:
  - Password Disabled Jumper Status
  - Ambient Light Detection Levels
  - Accelerometer Values
- Designate a physical input to act as a cabinet door contact sensor

- Have a response plan in place to investigate any alarms in person to prevent device compromise.

These recommendations help create overlapping security controls that cover both Protect and Detect functions. Having a response plan that considers the compromise of end devices such as the RTAC is valuable to the security of the overall system. These plans are the heart of the Respond and Recover functions of the NIST Framework. Another aspect of a strong response plan is having capable forensic tools to help analyze anomalies, misoperations, and network intrusions and the subject of Chapter IV.

IV. Respond: RTAC Forensic Process

This chapter focuses on the creation of a new tool for the Respond function of the NIST framework to replace presently available forensic processes that rely on insecure and unprotected communication protocols to create forensic artifacts.

4.1 Securing Programming Traffic

When programming a PLC, the control logic is created in one of the five high-level IEC-61131 programming languages. Once the control engineer has finished creating the control logic, the proprietary engineering software acts to transform it into the correct format for the target device. This end product is specific to the target device but can include formats such as an executable binary format that controls the specified PLC or a series of commands to modify the program stored in PLC memory. This product is then sent to the PLC over an Ethernet network or a local interface such as USB or serial located on the PLC. If this programming data is sent over the network, the packets containing the binary can be used as a valuable forensic artifact [33]. The binaries can be reverse engineered to determine the high-level source code or the programming commands can be decompiled with a tool such as Similo [21].

While useful as an artifact for Forensic investigators, a passive network intruder could also use the network traffic as a reconnaissance tool, inspecting the programming traffic to gain insight into the physical process the PLC controls and obtaining further information on the PLCs communication configuration. If instead the actor wanted to change the PLC control logic, they could intercept the packets as they traversed the network and modify their contents before they reach the end device. This interception and modification would be an example of a Man-in-the-Middle (MITM) attack as shown in Figure 7. It would also require the ability to reverse engineer the commands to understand the original intended logic. Another version of a control injection attack, Stuxnet attacked this programming chain on the Engineering Workstation, modifying the programming logic during the compilation process before it was sent over the network [34]. The ability to audit the programming logic and compare it to what was created by the controls engineer is where the forensic artifact derives its value.

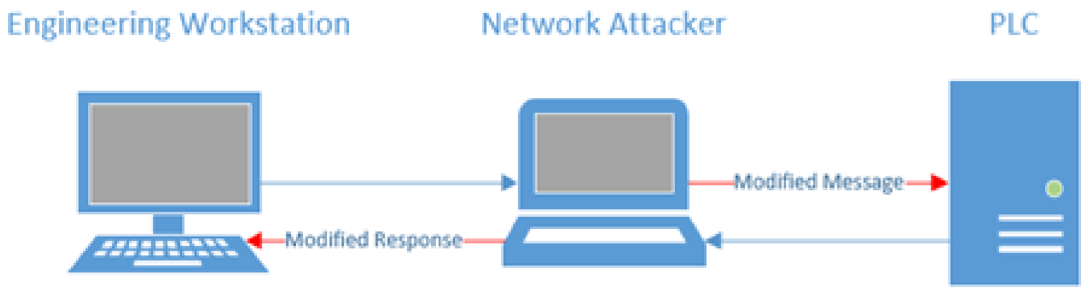


Figure 7. Man-in-the-Middle Attack Modifying Programming Traffic

Network based MITM attacks are enabled by the use of protocols that send their payloads in plaintext, which has been widely documented as a security vulnerability in ICS protocols [35]. Even with this demonstrated need for payload protection, the timing constraints in the process control, coupled with legacy hardware, make applying encryption a non-trivial task [35]. In contrast, programming the devices has no timing considerations, so encryption can and has been implemented on some newer devices. Digital signing is a cryptographic method in which a private key is used to "sign" the contents of a message. The signature is created by encrypting a hash code of the message using the private key. Encrypting the hash instead of the entire message is due to the relatively slow performance of asymmetric encryption. Ideally this method achieves two security services. The integrity of the message is ensured as the signature is based on its contents, and the identity of the message sender is verified [36]. Digital signing is one alternative that allows for reconstruction tools to continue to function because the basic message is still sent in plain text while the attached signature is generated from a hash of the message. Obviously, a malicious network agent could also use these plain text messages to build knowledge about the programming logic, making Digital Signing only a slightly more secure method. Additionally, naïve implementations may simply distribute the private key with the engineering software allowing an advanced threat to easily impersonate valid programming traffic. Figure 8 shows the digital signing process for an individual message sent between the Engineering Workstation and the PLC.

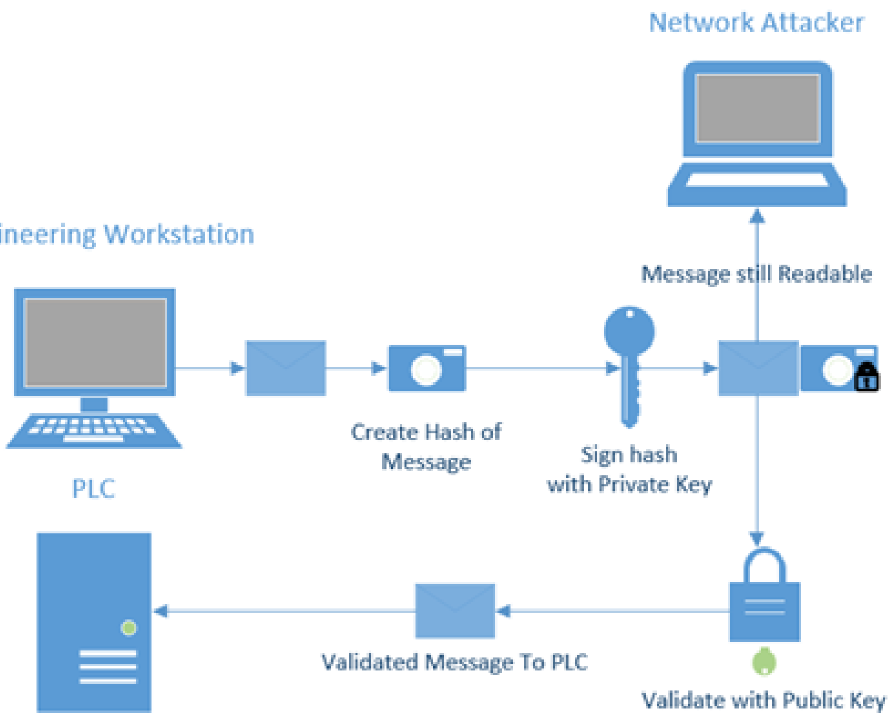


Figure 8. Digital Signatures Allows for the Validation of Network Messages

Asymmetric Cryptography uses a pair of keys, public and private, to encrypt and decrypt messages that can be sent across a network. Created to solve the key distribution problems present in symmetric cryptography, asymmetric cryptography ensures the integrity of the message and does not allow network intruders to inspect the contents of encrypted packets [36]. Figure 9 highlights the unavailability of encrypted messages to an intruder on a sensitive network. Despite the encryption of the network traffic, if correctly implemented, there are several ways to allow reconstruction tools to have access to the data required to perform forensic analysis on the network traffic.

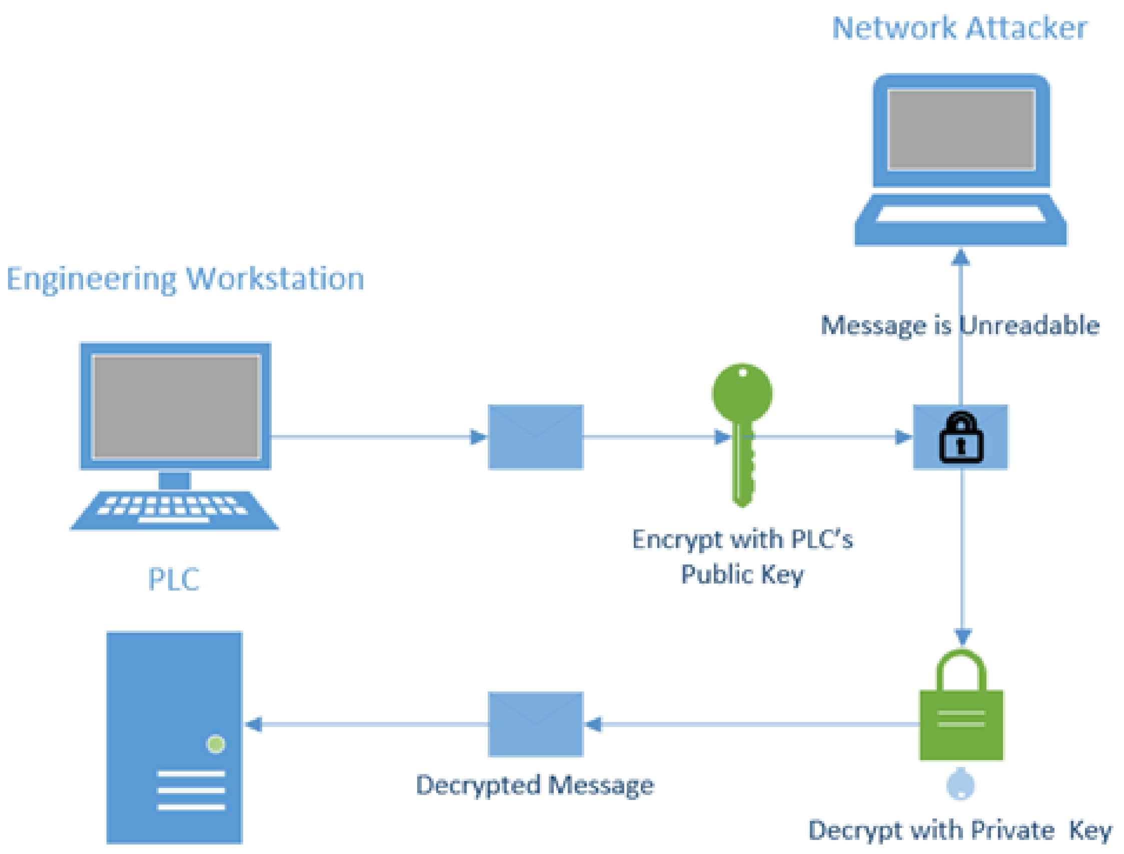


Figure 9. Asymmetric Encryption

One of the proposed approaches to solve the problem of decrypting the network traffic is the use of a secure central database of private keys and the collection and storage of network traffic between the engineering workstation and the end devices. In this approach, when the network is commissioned and end devices receive their initial configuration, the private key of the key pair for each end device is retained. Then, when incident response is conducted, or for a routine audit, the private key vault is accessed to decrypt the stored network traffic. A notional diagram of this can be seen in Figure 10. When a forensic investigator wants to inspect saved network traffic, they retrieve the encrypted traffic and the private keys of the devices associated with that session.

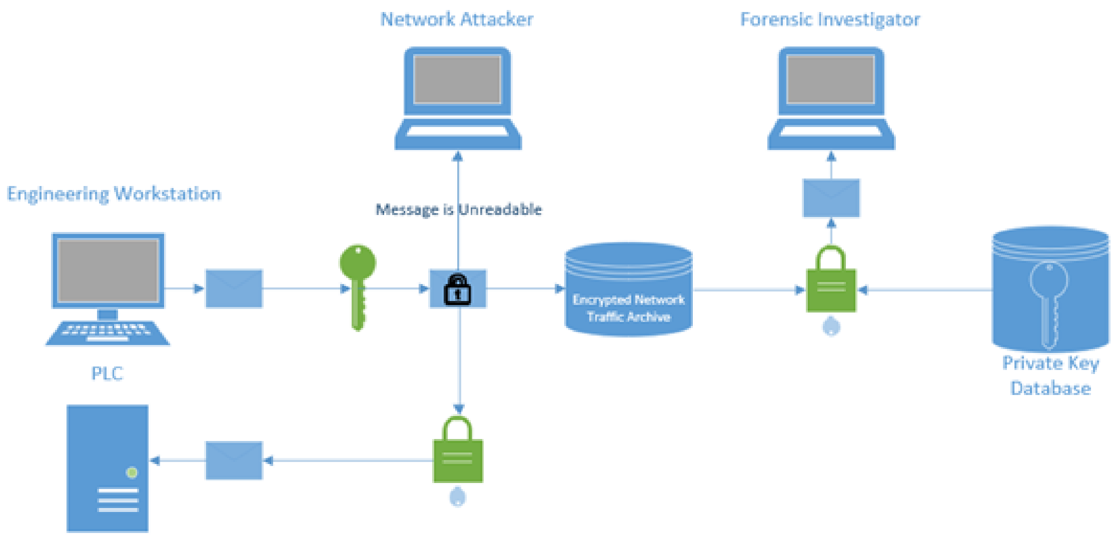


Figure 10. Private Key Database Implementation

In this approach, the security of the database storing the private keys is of utmost importance in this process. If the device that stored the keys was connected to the network and subsequently compromised, an attacker would gain access to the network traffic that the encryption was intended to protect. Additionally, an attacker with the private keys would be able to masquerade as one of the end devices of their choosing, with potentially serious consequences. This implementation is only successful for encryption techniques in which the shared secret can be determined from external observation of the session handshake. Protocols that use methods such as the Diffie-Hellman key exchange in which the key exchange is conducted over an assumed insecure channel will be unencryptable by the auditing processes, despite access to the private keys. These session-based protocols need a more involved decryption process.

Another approach, perhaps more difficult to implement than the central private key depository, would be a database containing each of the ephemeral session keys. This implementation would require that at the conclusion of a secured session between two devices, one of the devices would set up a secure connection with the central repository in order to deposit the shared secret of the now completed sessions. These session keys would be correlated with the network traffic based on the IP address, TCP ports, and network time to allow for network forensics. Sending the shared secret after the completion of the session would ensure the integrity and confidentiality of the network traffic while it is active. Figure 11 shows the storage of all traffic in an Encrypted Network Traffic Archive and the storage of session keys in a database. When an investigator wants to see the contents of traffic they retrieve both the traffic from the archive and the associated session key from the database. The security of this session key repository would still be of great importance to prevent any network traffic captured by malicious actors from being decrypted.

Another possible implementation would be to funnel network traffic of interest that is encrypted through a proxy server that would act as a historian, as shown in Figure 12. The proxy server would maintain two distinct encrypted network sessions, one with Device A and one with Device B. The proxy server would decrypt the traffic from device A, record the plain text network traffic, re-encrypt the network traffic, and send it to device B. This would in effect be a sanctioned man-in-the-middle attack. The traffic sent through the proxy server should be selected carefully as to avoid adding delay to time sensitive messages and to not increase the amount of data being archived to an undue amount. Any compromise to the proxy server could modify the traffic while it is in plain text. The proxy server should also maintain a record of its shared secrets for its encrypted sessions so that the upstream and downstream encrypted traffic could be compared to ensure that no changes have been made to

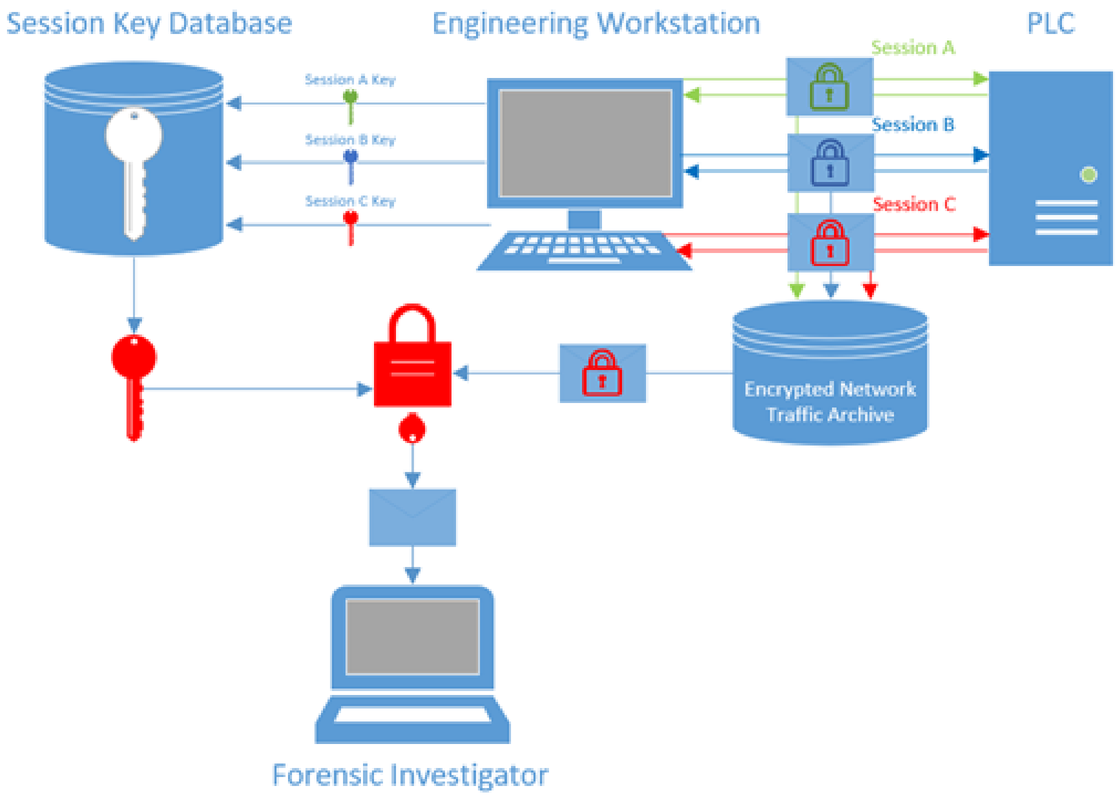


Figure 11. Session Key Database for Forensic Investigation

the payloads.

4.2 Network Forensic Process

The ability to recreate logic settings from network traffic is an important artifact for incidence response and enables the auditing of control settings that are actually sent to devices and not only what is present on the engineering workstation. When encrypted traffic is used to transport settings one of the previously explored approaches needs to be implemented.

For the RTAC, the current capabilities built into the device allow for the preshared private key paradigm to be employed. Figure 13 shows the necessary steps to perform forensics on the RTAC programming process. From the web interface, an externally

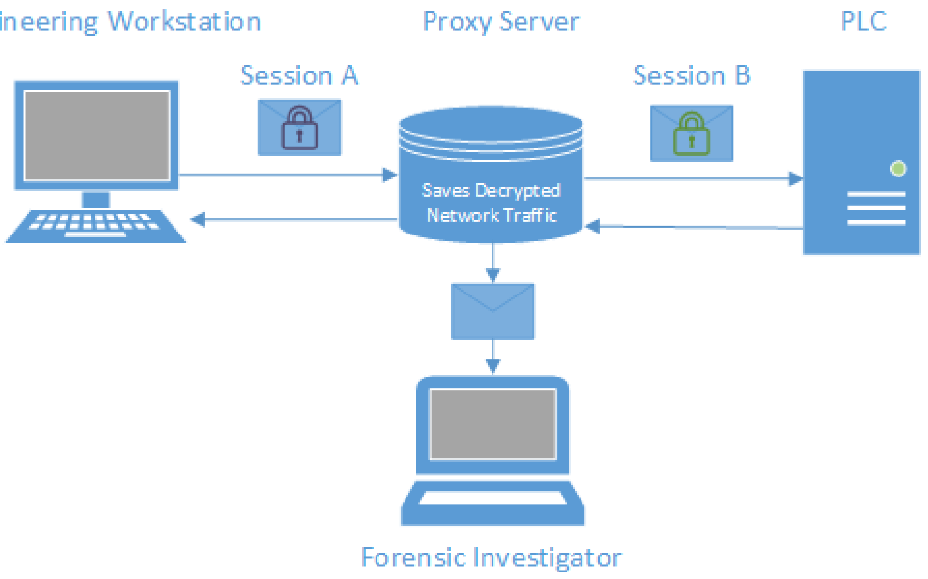


Figure 12. Proxy Server for Saving High Interest Traffic

generated private and public key pair can be uploaded to replace the self-signed certificate. To perform this demonstration a new X.509 Certificate was generated and uploaded to the RTAC. The private key was retained and added to Wireshark's RSA keys list for Transport Layer Security. Then Wireshark was used to capture the network traffic on the Engineering Workstations ethernet interface as the engineering software uploaded a new project file to the RTAC. At this point it can be shown that Wireshark is able to decrypt the programming network traffic sent to port 5432. Figure 14 shows the encrypted TLS segment data and Figure 15 shows the Decrypted TLS Data with the PGSQL commands clearly in plain text.

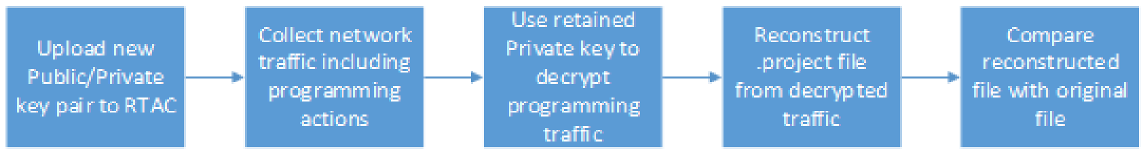
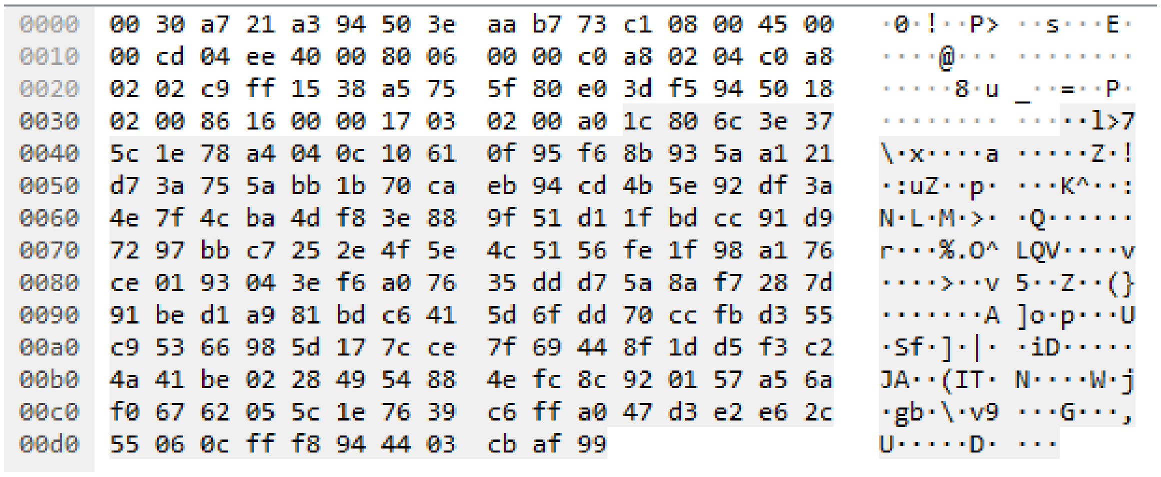
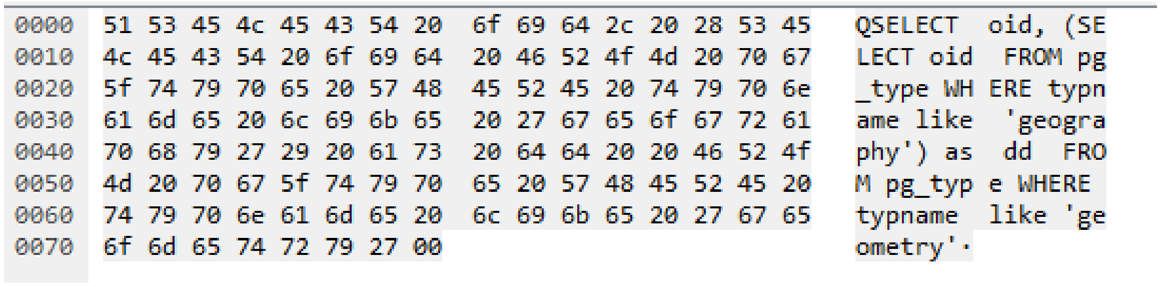

Upload new Public/Private key pair to RTAC
Collect network traffic including programming actions
Use retained Private key to decrypt programming traffic
Reconstruct .project file from decrypted traffic
Compare reconstructed file with original file

Figure 13. Network Forensic Process for SEL RTAC



```
0000  00 30 a7 21 a3 94 50 3e  aa b7 73 c1 08 00 45 00   ·0·!··P> ··s···E·
0010  00 cd 04 ee 40 00 80 06  00 00 c0 a8 02 04 c0 a8   ····@··· ········
0020  02 02 c9 ff 15 38 a5 75  5f 80 e0 3d f5 94 50 18   ·····8·u _··=··P·
0030  02 00 86 16 00 00 17 03  02 00 a0 1c 80 6c 3e 37   ········ ·····l>7
0040  5c 1e 78 a4 04 0c 10 61  0f 95 f6 8b 93 5a a1 21   \·x····a ·····Z·!
0050  d7 3a 75 5a bb 1b 70 ca  eb 94 cd 4b 5e 92 df 3a   ·:uZ··p· ···K^··:
0060  4e 7f 4c ba 4d f8 3e 88  9f 51 d1 1f bd cc 91 d9   N·L·M·>· ·Q······
0070  72 97 bb c7 25 2e 4f 5e  4c 51 56 fe 1f 98 a1 76   r···%.O^ LQV····v
0080  ce 01 93 04 3e f6 a0 76  35 dd d7 5a 8a f7 28 7d   ····>··v 5··Z··(}
0090  91 be d1 a9 81 bd c6 41  5d 6f dd 70 cc fb d3 55   ·······A ]o·p···U
00a0  c9 53 66 98 5d 17 7c ce  7f 69 44 8f 1d d5 f3 c2   ·Sf·]·|· ·iD·····
00b0  4a 41 be 02 28 49 54 88  4e fc 8c 92 01 57 a5 6a   JA··(IT· N····W·j
00c0  f0 67 62 05 5c 1e 76 39  c6 ff a0 47 d3 e2 e6 2c   ·gb·\·v9 ···G···,
00d0  55 06 0c ff f8 94 44 03  cb af 99                  U·····D· ···
```

Figure 14. Encrypted TLS Data

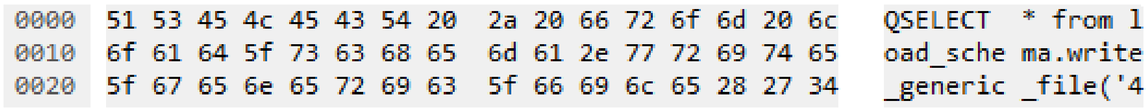


```
0000  51 53 45 4c 45 43 54 20  6f 69 64 2c 20 28 53 45   QSELECT  oid, (SE
0010  4c 45 43 54 20 6f 69 64  20 46 52 4f 4d 20 70 67   LECT oid  FROM pg
0020  5f 74 79 70 65 20 57 48  45 52 45 20 74 79 70 6e   _type WH ERE typn
0030  61 6d 65 20 6c 69 6b 65  20 27 67 65 6f 67 72 61   ame like  'geogra
0040  70 68 79 27 29 20 61 73  20 64 64 20 20 46 52 4f   phy') as  dd  FRO
0050  4d 20 70 67 5f 74 79 70  65 20 57 48 45 52 45 20   M pg_typ e WHERE 
0060  74 79 70 6e 61 6d 65 20  6c 69 6b 65 20 27 67 65   typname  like 'ge
0070  6f 6d 65 74 72 79 27 00                            ometry'·
```

Figure 15. Decrypted TLS Data

With the network traffic decrypted, a reconstruction tool could then recreate the programming logic that was sent to the device. An automated tool does not yet exist for the RTAC so a manual reconstruction must be conducted. When the RTAC is programmed, three files are sent: a compressed project archive containing the CODESYS `.project` file, a `.APP` file containing the compiled executable code, and a .CRC file containing a checksum for the .APP file. These files are sent by calling a PGSQL function, load_schema.write_generic_file, and then passing the byte contents of the file. Figure 16 shows the beginning of a TLS payload in which a file is written.

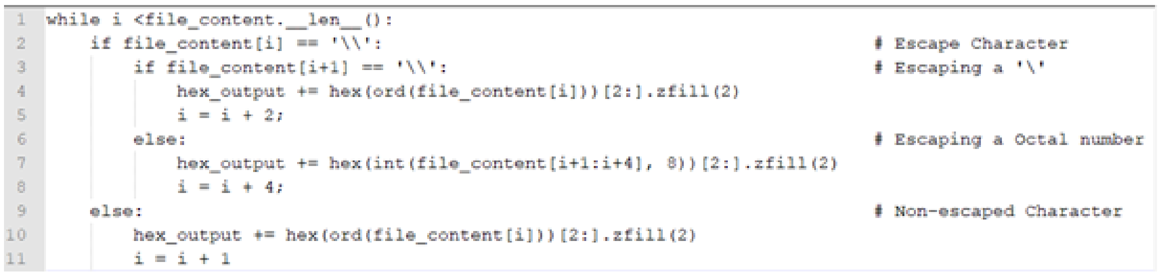
The TLS segment data must then be decoded; it contains a mix of octal encoded numerals, escaped special characters and some plaintext ascii characters. This was



```
0000  51 53 45 4c 45 43 54 20  2a 20 66 72 6f 6d 20 6c   QSELECT  * from l
0010  6f 61 64 5f 73 63 68 65  6d 61 2e 77 72 69 74 65   oad_sche ma.write
0020  5f 67 65 6e 65 72 69 63  5f 66 69 6c 65 28 27 34   _generic _file('4
```

Figure 16. Create File PGSQL Function Call

achieved using the python code in Figure 17, which created the hex-encoded file contents.



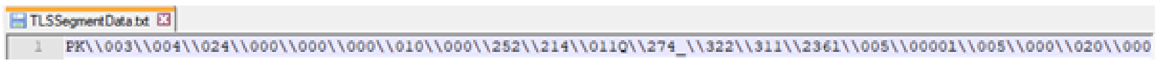
```
while i <file_content.__len__():
    if file_content[i] == '\\':                                      # Escape Character
        if file_content[i+1] == '\\':                                # Escaping a '\'
            hex_output += hex(ord(file_content[i]))[2:].zfill(2)
            i = i + 2;
        else:                                                        # Escaping a Octal number
            hex_output += hex(int(file_content[i+1:i+4], 8))[2:].zfill(2)
            i = i + 4;
    else:                                                            # Non-escaped Character
        hex_output += hex(ord(file_content[i]))[2:].zfill(2)
        i = i + 1
```

Figure 17. Python Decoding Script

Once the file has been reconstructed, the next step is to either perform a differential analysis between what was sent over the network and what was expected to be sent, or to save it as a baseline file to detect any later changes. RTAC AcSELerator does not provide the functionality to compare CODESYS `.project` files natively. However, when the engineering software prepares to send the RTAC programming logic it places the CODESYS `.project` file in an accessible temporary directory. This file can be retrieved and opened in the CODESYS development environment to use the provided compare function against the reconstructed file to find any differences in either logic or configuration. CODESYS can also be used to create the .APP file from the retrieved `.project` file to find any differences to the compiled binaries that were sent across the network.

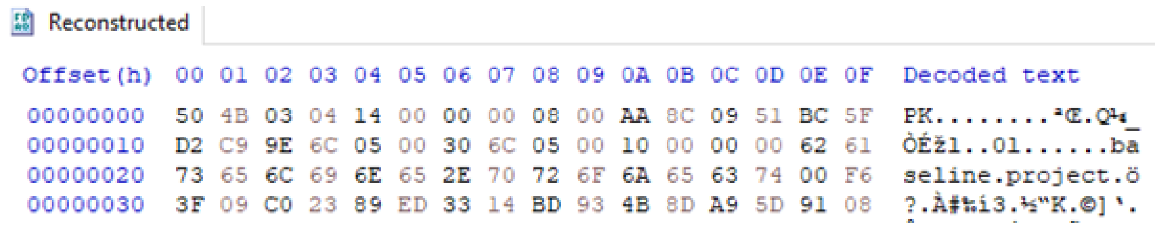
4.2.1 Forensic Process Results

This process was employed successfully against the programming of an SEL-RTAC. Figure 11 shows the first several bytes of the encoded `.project` file that was extracted from the network traffic. The entirety of the encoded file was decoded with the python script to retrieve the sent `.project`. The header for this file is shown in Figure 12. Note the first two bytes, "PK", indicating that it is a zip file and the plaintext "baseline.project" referring to the name of the application being run on the RTAC. This reconstructed `.project` file was then successfully opened in CODESYS to ensure that there was no corruption and compared against the original `.project` file with no differences found.



```
PK\\003\\004\\024\\000\\000\\000\\010\\000\\252\\214\\011Q\\274_\\322\\311\\2361\\005\\00001\\005\\000\\020\\000
```

Figure 18. Encoded .project File



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  50 4B 03 04 14 00 00 00 08 00 AA 8C 09 51 BC 5F  PK........ªŒ.Q¼_
00000010  D2 C9 9E 6C 05 00 30 6C 05 00 10 00 00 00 62 61  ÒÉžl..0l......ba
00000020  73 65 6C 69 6E 65 2E 70 72 6F 6A 65 63 74 00 F6  seline.project.ö
00000030  3F 09 C0 23 89 ED 33 14 BD 93 4B 8D A9 5D 91 08  ?.À#‰í3.½"K.©]'.
```

Figure 19. Decoded .project File

This process is able to reconstruct any arbitrary project sent over the network from the network traffic. This provides a valuable service to determine what logic was present on the RTAC at a specific time. The RTAC does not maintain a history of previous control logic, and there are no available reconstruction tools that were designed to reconstruct encrypted control logic. This process could also supplement the lack of subversion tools present on the engineering software. Subversion is an application that is used for maintaining historical versions of coding projects and allows for changes to be documented and easily reverted. This process allows for the data mining of stored network traffic to recover the project as sent even if the version present on the engineering workstation has been updated.

4.3 Summary and Future Implementations

This chapter detailed the move from plaintext protocols to more secure encrypted versions and the need to maintain the current capabilities to create forensic artifacts for incident response. The described implementation of this process relies on the use of TLS 1.1 to encrypt the PGSQL commands. If SEL moves to a more recent implementation of TLS, it will employ session-based keys which cannot be calculated from the TLS handshake. This will make network traffic unencryptable unless SEL makes the ephemeral session keys that are typically only in RAM available for network forensics. To make the traffic available for reconstruction, one of the more involved decryption implementations would need to be pursued, either the proxy server or the session key database. The proxy server implementation would mimic the Qasim, Lopez and Ahmed's Similo Virtual PLC-framework as it would act as an intermediary to build the knowledge database but with the addition of encryption handling [21]. Forensic tools are used in response to detected anomalies and misoperations of ICS. Detecting these anomalous activities for the RTAC is the subject area of Chapter V.

V. Detect: Anomaly Detection System Design

Chapter II provided the general components of an ADS, this Chapter details the specific implementation of an ADS and the related framework to collect the needed data and test its performance. This system is created to supplement the existing RTAC security features helping detect network intrusions without the addition of any supplementary devices to existing installations. The goal of the system is to detect intrusions before an attacker can cause misoperations or create persistent footholds.

5.1 Experimental Design

The data collection framework uses a combination of functions native to the RTAC and supplementary functions implemented in python that could be integrated by the manufacturer into the device's firmware for future releases or implemented in the logic engine by an operator. An experimental treatment or set of controlled factors for each trial consisted of a firmware revision, a control logic file, and the use of a specific network intrusion. Table 1 shows the possible factors. Selecting one factor from each column for each treatment gives 24 different combinations. These variations sought to capture the effectiveness of an ADS across varying workloads and various intrusions. 10 trials were conducted for each experimental treatment leading to 240 total trials. The order in which each trial was conducted was selected by creating a list of all trials and randomizing their order. This randomization was to reduce the effect of uncontrollable environmental factors such as operating temperature or power supply voltage. The devices were on a closed network so no outside network traffic was present.

Table 1. Experimental Factors

Firmware
R145
R146
R147

Project Type
Low Task Time
High Task Time

Network Intrusion
Baseline
ARP Spoofing
PostgreSQL Queries
CODESYS Connection

5.1.1 Factors

5.1.1.1 Firmware Revisions

Previous research had shown that monitoring task times in PLCs can identify modifications to firmware [24, 25]. To verify that this research can be generalized to the RTAC, three firmware revisions released by SEL are used: R145, R146, and R147.

5.1.1.2 RTAC Project Files

Two RTAC Project Files were used for the experiment. Project 1 was a limited functionality project with an average task time of approximately 1300 microseconds. It contained only the necessary logic for the RTAC operation with addition of a single Modbus server for data collection. While the deployment of such a project is unlikely to be done in an operational environment, as even a data concentrator would have additional network connections and data mapping logic, this project provides the lowest possible task time while still providing the necessary data for the ADS.
Project 2 is a much more complex project, designed for an average task time of approximately 9000 microseconds. It achieves this long task time by performing numerous complex mathematical operations that use both pseudo-random inputs provided by the RTAC's *SELRand* library and time varying inputs such as the number of bytes sent and received by the ethernet ports. These two projects represent both ends of the spectrum of possible project task times. Future work beyond the scope

of this thesis will examine the use of branching projects whose complexity can vary based on the current state of the process it is monitoring.

5.1.1.3 Network Intrusions

In order to assess the viability of using task time as a data feature for intrusion detection, a variety of feasible network intrusions or attacks needed to be developed. Hence, implementation details of the RTAC were carefully explored in order to understand features that malicious attackers could exploit. Successful exploits would result in arguably detectable footholds; detecting the footholds and the mechanisms that create these footholds is the purpose of the ADS. The following intrusion mechanisms were used in the experiment for their demonstrated viability as an attack vector or as the manifestation of a created foothold.

- Baseline

  The baseline treatment only uses the network traffic necessary to harvest the ADS data. This provides an experimental backdrop from which to detect the network intrusions. Repeated baseline trials are tested against each other to determine the FPR for each algorithm.

- ARP Spoofing

  ARP spoofing is a standard mechanism for the performance of MITM attacks and extensively used in research to demonstrate the vulnerability of ICS devices and networks [37, 38, 32, 39]. The exploit takes advantage of the lack of authentication in the Address Resolution Protocol (ARP) to send unsolicited resolution responses containing false information about the topology of the network. If undetected, this fake link layer information causes network traffic to be misdirected to a network attacker.

  Encrypted traffic severely limits an attacker's ability to collect information or inject packets. Widely used industry protocols, however, still rely on legacy implementations and are largely sent in plain text. This lack of security allows for a range of modification attacks against widely used protocols such as Modbus and DNP3 [37, 39]. The lack of encryption within the configuration protocols of various devices also pose security risks to ICS networks in regards to ARP spoofing [40, 19].

Improper configuration of the RTAC can also pose additional risks to ARP spoofing. The RTAC generates a self-signed certificate upon the installation of a new firmware revision. This certificate is used for encrypting the HTTPS connection for initial configuration and for encrypting PostgreSQL traffic. If this certificate is not replaced, an adversary could perform a MITM attack and impersonate the RTAC using a previously compromised certificate. This would allow an attacker to harvest credentials from the supposed secure connection and inject arbitrary modifications into the system configuration or programming traffic.

- PostgreSQL Queries

  The RTAC uses a PostgreSQL database to manage the control logic on the device and act as an interface with the underlying operating system. Engineering functions are programmed as SQL queries that are callable from the web interface. These functions include changing the Internet Protocol address, testing network connectivity by pinging other devices, and reading system diagnostic information. As an open source database, PostgreSQL vulnerabilities are typically discovered and patched relatively quickly as compared to PLC manufacturer patching responses. However, there is still a burden placed on the manufacturer to incorporate the most up-to-date database version in the device firmware and for end users to patch their own devices.

If the PostgreSQL database is exploited or database credentials are compromised an attacker is free to perform DoS attacks, as the RTAC can be restarted by a database query. A system restart is an easily detectable anomaly but subtler attacks are also possible. The PostgreSQL queries employed for foothold emulation during this experiment were reads of the control logic project from the RTAC. The exfiltration of this control logic represents a major reconnaissance tool for an attacker as it provides in-depth knowledge of both the process the RTAC is controlling and the various devices it is communicating with.

- CODESYS Connection

  The *codesys* runtime application executing on the RTAC is responsible for the industry protocol communication and the control logic operation. It has a well known network port, 1217, that allows for various engineering functions. These functions include uploading/downloading control logic programs, monitoring real-time logic values, forcing logic values for debugging purposes, and stopping/starting the control logic execution. If enabled, the connection also allows access to the device's file system. Vulnerabilities related to the use of the *codesys* runtime as a PLC framework have been previously documented [14] and exploited for the decompilation of control logic [20]. While *codesys* has recently implemented authentication features to create a connection with the runtime, the RTAC has not activated this feature and does not require credentials on the port itself. Instead, TCP connections to the port are blocked by default and the port can be opened and closed by accessing the PostgreSQL database.

  The file system access is a significant concern on the RTAC due to its use of TLS 1.1 for PostgreSQL traffic and use as a web server for configuration. The private server key that is used for encryption can be exfiltrated from the RTAC using a *codesys* connection and then used to decrypt any communication and extract information such as username and password. Additionally, attackers could insert or modify files on the device that may enable further exploitation. To emulate this network intrusion, a passive *codesys* connection was active during the data collection. This passive connection would be the minimum workload generated by a *codesys* connection with no files being read from the RTAC and no process values being changed from the engineering workstation.

5.1.2 Data Set Collection

For each trial, 30,000 task time samples were collected. This data was collected by creating a modbus server on the RTAC and polling the server using a modbus client implemented in python using the *pymodbus* module. After receiving a response from the server, the client waited 1 ms before polling again. Task time was used as the data feature due to its ability to detect logic and firmware changes, and sensitivity to changes in network traffic as seen in previous research [25, 24, 26]. Each data set was saved as a `.csv` file for future analysis using the selected decision methods. Figure 20 shows a notional data flow for the data collection process as well as the experimental factors being tested. The steps for each trial are as follows:

1. Install the selected firmware version for the current trial.

2. Upload the selected RTAC project file to the device. This step ensures that the correct firmware has been installed as each project file is firmware specific.

3. The data collection script is started, waiting 1 minute for the device to reach steady state.

4. The associated network intrusion for the trial begins.

5. 30,000 data points are collected.

6. The data is saved in a .csv

The data as collected is then used as inputs for various tests to discriminate between baseline trials and data sets containing network intrusions.

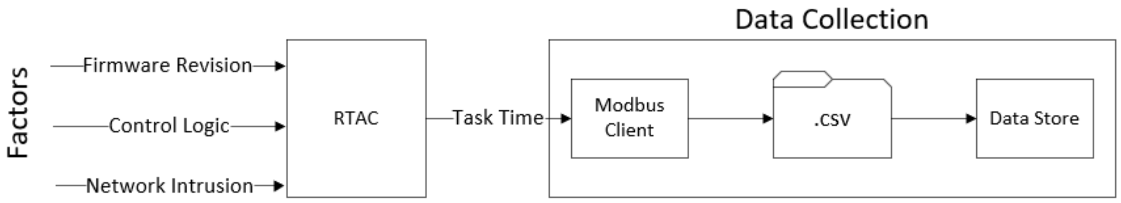


Figure 20. Experimental Data Collection System

5.1.3 Selecting a Discriminator

Pilot studies confirmed previous research indicating that the task time was not normally distributed [24, 26]. Figure 21 shows a scatter plot of the sampled task times from a pilot study with no network intrusions. There are three visually distinct clusters in task time and no apparent correlation between when the sample is taken and the associated task time. These three clusters can be seen in greater detail when the data is viewed as the kernel density estimation plot that is shown in Figure 22.

Because the task time population is non-normal, three non-parametric statistical tests were chosen from literature. The efficacy of algorithms to detect network intrusions from task time is then analyzed.

- Permutation Test

  The permutation test is a re-sampling test that investigates the probability that two sample populations are from the same distribution. This test can be done without making assumptions about the distribution of the samples [41]. This test has been used previously to detect logic and firmware changes [24]. To perform this test the *mlxtend* python module was used [42].

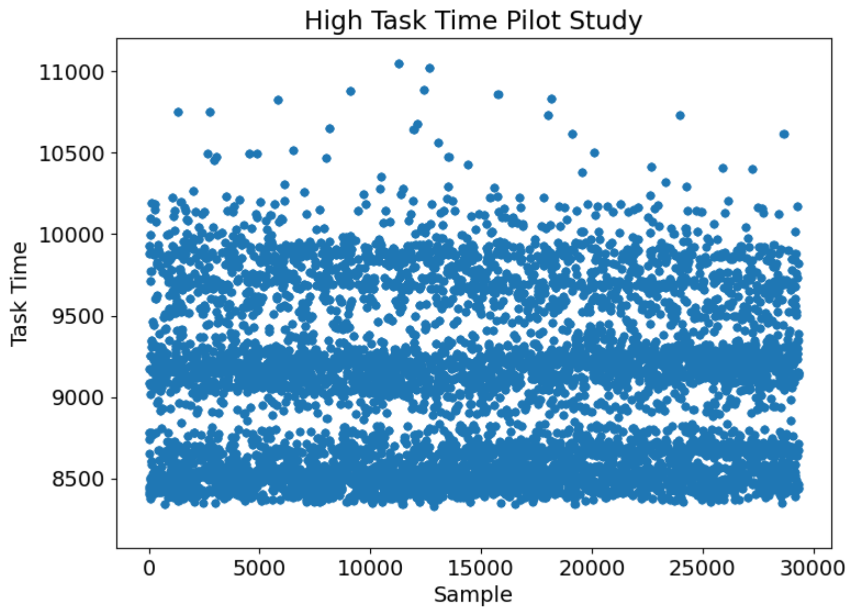


Figure 21. Scatter Plot of Task Time Distribution

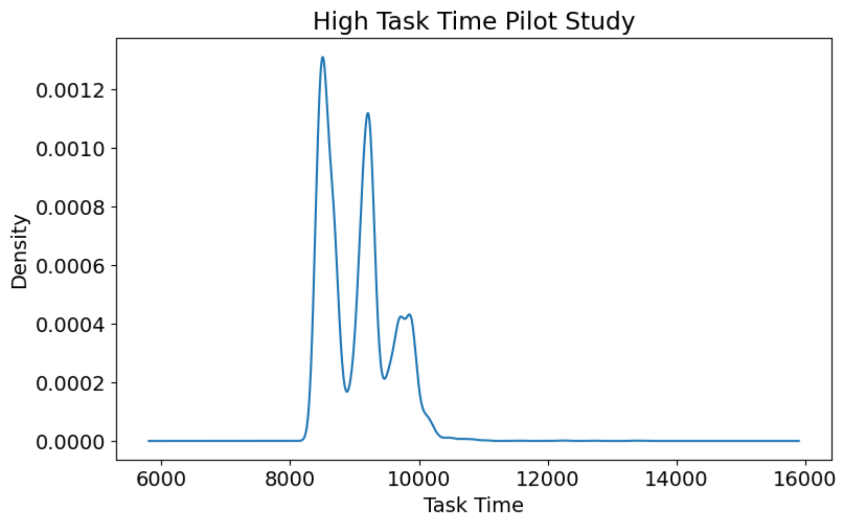


Figure 22. Kernel Density Estimation of Task Time

- Mann-Whitney U Test

  The Mann-Whitney U test is a non-parametric test that explores the null hypothesis that one population is stochastically larger than the other [43, 44].

While typically used in behavioral sciences, the Mann-Whitney has been employed to test for statistical differences in sampled and forecasted network traffic [45, 46]. This test was conducted on the task time data in python using the *scipy.stats.mannwhitneyu* function [47].

- Kolmogorov-Smirnov Test

  The Kolmogorov-Smirnov Test (KS Test) is a non-parametric goodness-of-fit test that probes the hypothesis that two independent samples are drawn from the same distribution. This test has the advantage of considering the entirety of a distribution function rather than just the difference in a test statistic. For this test to be valid, the task time distribution is assumed to be continuous. The KS Test statistic is the maximum absolute difference between the two distribution functions [43] and was used by Formby and Beyah to detect logic changes [25]. For this work, the test was conducted in python using the *scipy.stats.kstest* function [47].

Figure 23 shows the Kernel Density estimation for the combined data sets of all trials for the high task time project with firmware rev 145. Visually, clear differences can be seen in the number, height and location of the distribution peaks. Each of these tests is an algorithmic way to try and detect these differences and other unperceived differences.

5.1.4 System Evaluator

To test each discriminator, the data sets are separated by both firmware and project type. This reduces the buoying effect of testing a low task time data set against a high task time data set inflating the true positive rate. Then each intrusion-free data set is tested against all other data sets in its firmware/project subset. Figure 24 shows the steps of the analysis process.

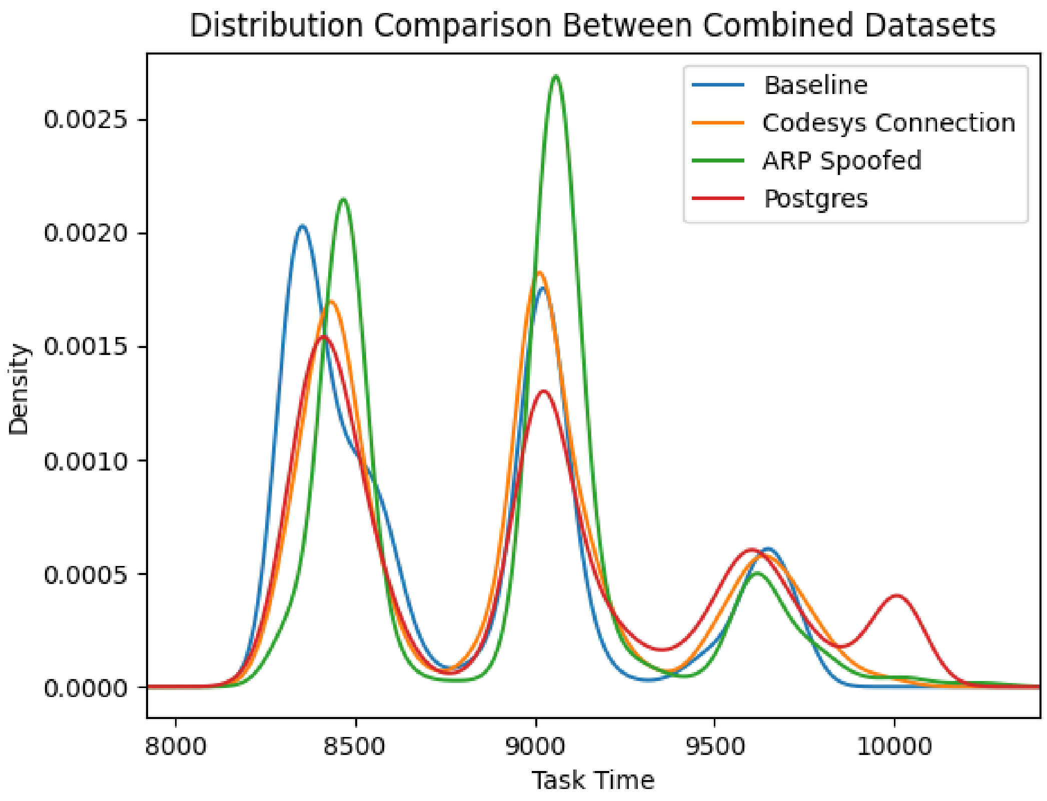


Figure 23. Distribution Comparison of Combined Data Sets

The results of these tests are used to calculate the true positive and false positive rate for each decision algorithm. This process can be repeated, varying the decision threshold to generate a Receiver Operating Characteristic (ROC) curve for each algorithm. This curve helps explore the tradeoff between True and False positive rates [24]. Additionally, the Area Under the Curve (AUC) can help compare the performance of multiple classifiers [48].

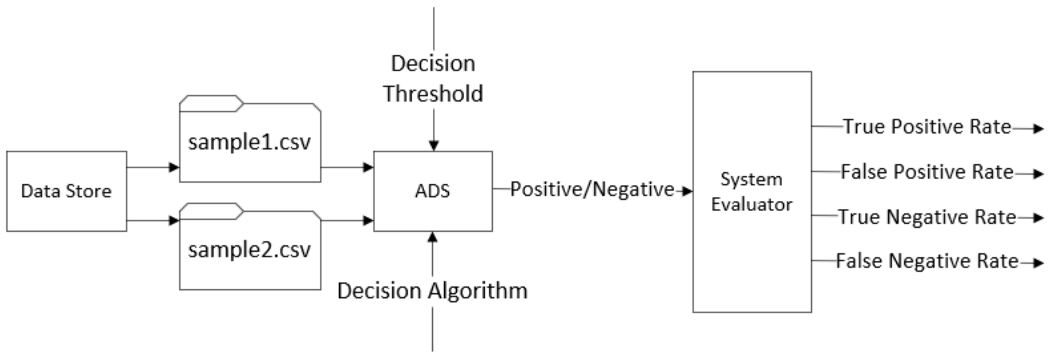


Figure 24. Experimental Analysis System

VI. Analysis and Results

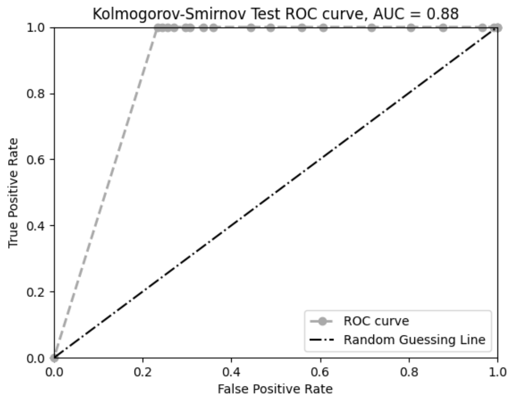
6.1 Overview

This chapter presents the observations, results, and analysis from the experimental activities described in Chapters V. It focuses on the ability of the ADS to perform the Detect Function of the NIST Framework.

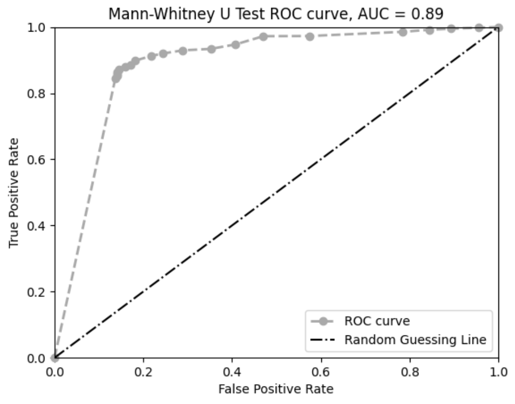
6.1.1 Anomaly Detection Rates

The body of this section focuses on the compiled results from the 240 experimental trials. First, ROC Curves were created for each of the decision algorithms, these curves can be found in Figure 25. The area under each curve was calculated and is displayed on the figures. The Mann-Whitney U test had the highest AUC with a value of 0.89 with the KS test being slightly lower at 0.88, and the Permutation test had a value of 0.80. An ideal classifier would have an AUC of 1.0. Using these curves a decision threshold for each algorithm was selected to showcase their representative performance. Each test having an area greater than that of the random guessing line's .5, lends credibility to the the claim that task time is a good data feature to try and detect the network intrusions examined by this research.

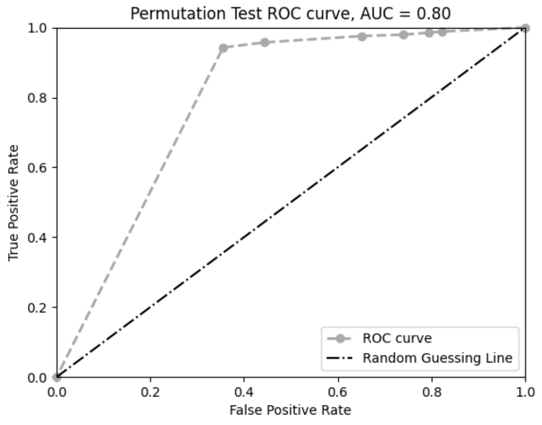
Table 2 has the results of the permutation test, Table 4 has the results for the KS test, and Table 3 has the results of the Mann-Whitney U Test. Values shown in bold do not meet the desired performance threshold of less than 0.1 for the FPR or greater than 0.9 for the combined TPR. The desired performance thresholds are based on previous research in anomaly detection systems [24]. Each firmware-project pair is shown to better understand if an algorithm did poorly overall or if specific test cases were more difficult to detect. The Permutation test failed to meet any of the FPR thresholds but performed well detecting the network intrusions with two



(a)



(b)



(c)

Figure 25. Algorithm ROC Curves

firmware/project pairs coming close but not meeting the 0.9 cutoff. The Mann-Whitney U test did well on FPR overall but failed to have sufficient TPR in half of the test cases. The KS test had a TPR of 1.0 across all test cases but struggled with FPR, failing to meet the desired threshold in 4 of the 6 firmware/project pairs. The strength of the KS test is an extremely promising result.

One Experimental treatment that may be a potential outlier in the data is Firmware revision 146 with High Project Task time. The performance of all tests have markedly different FPRs than the rest of the body of data. This could be due to some unknown behaviour in the firmware version or in the automated conversion process for an RTAC

Table 2. Permutation Test Results

Permutation Test: Decision Threshold 0.0001						
Firmware	Project Task Time	FPR	TPR			
			Combined	CODESYS	ARP Spoof	PostgreSQL
145	Low	**0.46**	0.94	0.96	0.86	0.99
146	Low	**0.22**	0.97	1	1	0.92
147	Low	**0.40**	**0.873333**	0.81	0.91	0.9
145	High	**0.20**	1.00	1.00	1.00	1.00
146	High	**0.69**	**0.90**	0.92	0.92	0.85
147	High	**0.17**	0.98	0.98	1.00	0.95

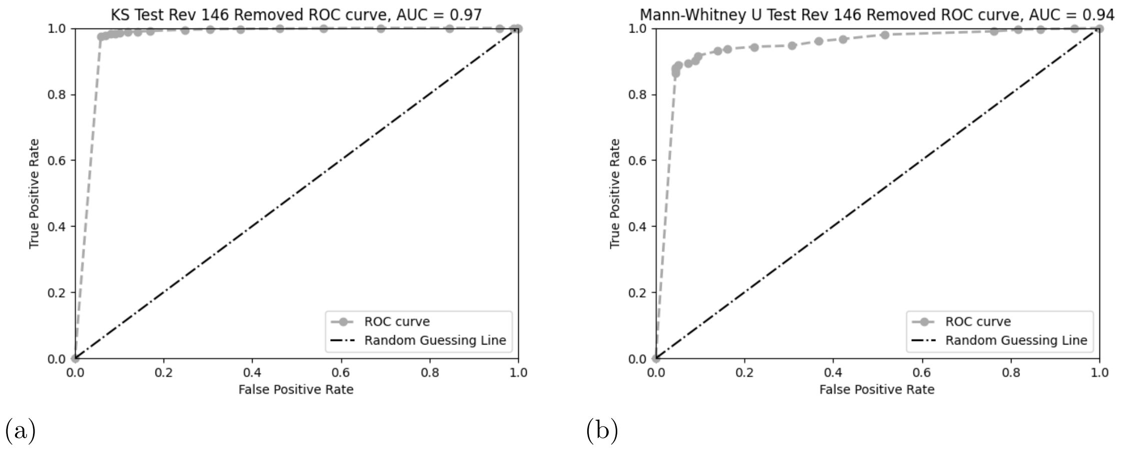
Table 3. Mann-Whitney U Test Results

Mann-Whitney U Test: Decision Threshold $1 * 10^{-14}$						
Firmware	Project Task Time	FPR	TPR			
			Combined	CODESYS	ARP Spoof	PostgreSQL
145	Low	0.089	0.96	0.99	0.97	0.91
146	Low	0	0.96	1.00	0.90	0.99
147	Low	0.080	**0.81**	0.81	0.81	0.81
145	High	0	0.97	1.00	0.92	1.00
146	High	**0.67**	**0.70**	0.70	0.73	0.67
147	High	0	**0.78**	0.74	0.75	0.86

Table 4. KS Test Results

Kolmogorov-Smirnov Test: Decision Threshold $1 * 10^{-17}$						
Firmware	Project Task Time	FPR	TPR			
			Combined	CODESYS	ARP Spoof	PostgreSQL
145	Low	**0.11**	1.00	1.00	1.00	1.00
146	Low	0.00	1.00	1.00	1.00	1.00
147	Low	**0.16**	1.00	1.00	1.00	1.00
145	High	**0.22**	1.00	1.00	1.00	1.00
146	High	**0.91**	1.00	1.00	1.00	1.00
147	High	0.00	1.00	1.00	1.00	1.00

AcSELerator project to be made compatible with a new firmware version. Figure 26 shows the ROC curves for each algorithm with this outlier's data removed, boosting the AUC of the KS test to 0.97 and the AUC of the Mann-Whitney U test to 0.94. The Permutation test showed a lesser improvement to 0.83. All further analysis completed in this chapter still includes the data from this potential outlier.



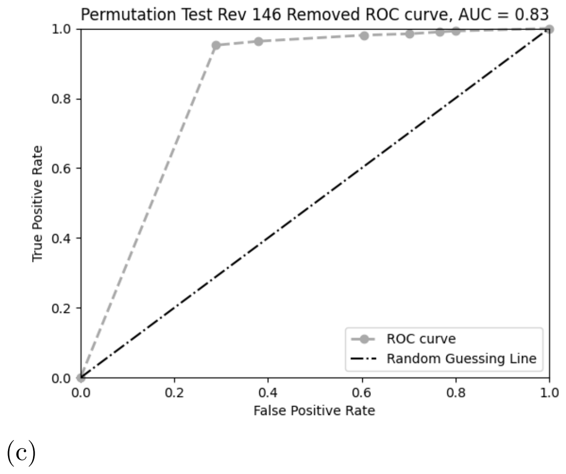


Figure 26. Algorithm ROC Curves Without Rev 146 Data

6.1.2 Testing Percentiles

While this analysis focused on testing the entire data set, previous research has performed some treatments on the data before testing to improve the performance of the system under test [24]. These data treatments seek to remove data points that are either spurious or outliers by only testing regions of the data distribution. To conduct this analysis each data set was sorted and then a range of percentiles was used for the analysis each statistical test. The results of these tests can be seen Table 5. As showing the TPR and FPR for each experimental treatment would be come cumbersome for the number of percentile combinations used the results have been reduced to just the overall AUC for each test combining all firmware versions and project files results. The highest value for each test is shown in bold.

The best results stem from the tests in which the entire data set was used. This high performance without removing alleged outliers supports the supposition that the statistically detectable differences between the baseline and network intrusion data sets lie in the extremes at either end of the distribution. Further bolstering this hypothesis, there is a marked difference between the results of tests looking at the first 10% of the data set and the tests only looking at the data between the 2nd to 10th percentile. Even as the upper bound increases there is actually a slight depression in performance and then it buoys back up as the top percentiles are incorporated. This analysis supports that the data set should not be treated before being fed into the ADS decision algorithm. Modifying the percentile also changes the number of data points being tested.The effect of varying sample size also needs to be analyzed.

6.1.3 Effect of Sample Size

A major contributor to detection latency is the amount of time it takes to collect the representative sample. If the sample size can be reduced without having an adverse effect on detection rates the ADS will be more responsive and be able to better mitigate the effects of the intrusion. A device with an average task cycle time of 10 ms and sampled every task cycle would take 5 minutes to collect 30000 samples. If that task cycle time was instead 100 ms it would be 50 minutes for the same data set size. Table 6 shows the results of varying sample size by only testing the first X samples in each data set. Performance for each statistical test increases as sample size grows until an inflection point is reached. For the Mann-Whitney U Test and the Kolmogorov-Smirnov test peak performance is reached with 8000 samples. The

Table 5. Effect of Varying Percentile Range on Area Under Curve

Percentile Tested		AUC		
Lower Bound	Upper Bound	Mann-Whitney	KS Test	Permutation
0	10	0.86	0.84	0.73
0	20	0.84	0.85	0.67
0	25	0.83	0.84	0.67
0	50	0.83	0.83	0.70
0	75	0.85	0.86	0.70
0	95	0.86	0.87	0.71
0	100	**0.89**	**0.88**	**0.80**
50	60	0.63	0.68	0.56
50	75	0.73	0.77	0.59
50	95	0.79	0.79	0.67
50	98	0.81	0.81	0.68
2	10	0.77	0.79	0.62
2	20	0.79	0.82	0.65
2	25	0.79	0.82	0.65
2	50	0.82	0.82	0.65
2	75	0.83	0.86	0.68
2	95	0.86	0.87	0.75
2	98	0.87	0.87	0.76

permutation test's best AUC is reached with 16000 samples.

The decrease in performance as sample size gets too large may be attributable to the test detecting minute differences in baseline data sets and causing the FPR to rise while the TPR remains high. An example of this negative effect is seen in the Kolmogorov-Smirnov Test (KS Test) with both the 8000 and 16000 data sample having a 1.0 TPR but the 8000 sample test has a FPR of 0.044 while the 16000 sample test has an FPR of 0.11.

6.1.4 Performance of Continual Monitoring

In order to further reduce detection latency the discrete data sets could be abandoned for the use of a sliding window of data. This change in collection strategy necessitates a change in decision strategy. Without the ability to amortize the cost

Table 6. Effect of Sample Size on Area Under Curve

	AUC		
Sample Size	Mann-Whitney	KS Test	Permutation
100	0.654	0.736	0.602
200	0.697	0.773	0.648
500	0.714	0.848	0.631
1000	0.770	0.904	0.659
2000	0.813	0.923	0.697
4000	0.874	0.937	0.758
8000	**0.919**	**0.953**	0.780
16000	0.912	0.924	**0.829**
28000	0.881	0.883	0.795

of each statistical test as data is collected the decision engine would quickly fall behind. To perform such an evaluation scheme a more light weight algorithm is needed. Example of these continuous monitoring schemes includes the cumulative sum or CUSUM [49]. The CUSUM algorithm was used by Formby et al. for logic change detection to great success being able to detect small logic changes within minutes with zero false positives [25]. When the CUSUM is implemented it is assumed that the distribution is normally distributed [50], this assumption does not hold true for the collected RTAC data but the performance of the algorithm will still be explored. The algorithm as described by Formby et al is shown in Algorithm 1.

This algorithm omits the allowance or slack value that is traditionally part of the CUSUM algorithm [50]. This value typically is chosen to be halfway between the target mean and the "out-of-control" value. Its omission causes the accumulated error to slew much more rapidly as any difference from the mean causes a change but allows for very minute, consistent changes to be detected. The ten baseline trials for each firmware were used as inputs to the algorithm to generate firmware/project pair unique thresholds to detect the network intrusions. Because these thresholds were empirically derived from the baseline cases they allow for the quickest possible detection times without any false positives, possibly at the expense of the TPR.

Algorithm 1 General Anomaly Detection Algorithm

```
procedure CUSUM                                        ▷ Cumulative Sum Algorithm [25]
    $S_n \leftarrow 0$
    $S_p \leftarrow 0$
    $m \leftarrow$ population mean
    $T_n \leftarrow$ negative change threshold
    $T_p \leftarrow$ positive change threshold
    while $True$ do                                    ▷ Collect Samples Forever
        $x \leftarrow$ new task time sample
        $S_n \leftarrow min(0, S_n + x - m)$
        $S_p \leftarrow max(0, S_p + x - m)$
        if $S_n \leq T_n$ then
            Negative Change Alarm
        end if
        if $S_p \geq T_p$ then
            Positive Change Alarm
        end if
    end while
end procedure
```

Table 7 shows the results of the CUSUM algorithm using the derived thresholds. Each firmware/project pair performed extremely well detecting the network intrusions. The TPR for the high task time project on rev 146 was the lowest failing to detect intrusions in 6 of the 30 data sets. The number of samples to the detection were quite high requiring on average over 14000 samples to reach the detection threshold. Revision 145 with the high task time performed the best, detecting each network intrusion and requiring only an average of 3732 samples to reach the detection threshold. This strong TPR coupled with its low detection time and lightweight computational requirements make the CUSUM method an obvious choice for future research.

6.1.5 Improving Detection Rates and Future Work

With both a compiled data set and a data collection framework in place further discrimination strategies may be employed. The use of multi-dimensionality will add

Table 7. Cumulative Sum Performance

	Cumulative Sum Algorithm Results		
Firmware	Project Task Time	Average Number of Samples To Detection	TPR
145	Low	10577	0.97
146	Low	4035	1
147	Low	8069	0.9
145	High	3732	1
146	High	14230	0.8
147	High	7345	0.97

computational complexity and should be weighed against the added performance. Additional continuous monitoring schemes can also be tested such as the Simple Moving Average supported by Vargas et al. [27] or more advanced methods such as the Exponentially Weighted Moving Average [50].

As candidate algorithms are vetted they should be implemented in the CODESYS environment for deployment to the RTAC. By moving the detection from a central system to the RTAC itself the burden of network traffic will be removed and the potential for malicious tampering while the data is in transit will be reduced. This will also allow for integration with preexisting capabilities such as syslog to allow for a single point of security auditing for the RTAC. Additionally, with multiple detection tests being implemented in discrete function blocks, voting schemes can be used and tuned. The CUSUM algorithm may be the most promising as its implementation could be rapidly placed on the RTAC has been previously used to detect logic and firmware changes by Formby et al. and has been shown to rapidly detect the network intrusions under test in this research.

6.2 Summary

This chapter discussed the experimental results of the created data collection framework and selected detection tests. It provides strong evidence that task time can be used to detect network intrusions in addition to the logic and firmware changes demonstrated by previous research. Both statistical and continuous monitoring strategies were explored along with the effect of data treatments and varying sample sizes. Future work will seek to build on these promising results placing the detection algorithms on the device itself to remove the network burden related to data collection.

VII. Conclusion

7.1 Overview

This chapter summarizes the work performed for this research including a demonstration of physical access risks, the creation of a forensic process for encrypted programming traffic, and the design of a data-collection framework for the development and evaluation of an ADS. It reiterates contributions of the work and summarizes the observations and analysis of the tested scenarios. Recommendations and future work are also discussed.

7.2 Summary

This research focused on the cybersecurity of the SEL-3505 RTAC using the five functions of the NIST Cybersecurity framework as a lens to inform necessary security controllers. Its specific contributions to the fields are as follows:

- **Physical Vulnerability:** Demonstration of the ability to compromise end devices when physical access is given.
- **Forensic Process:** A network forensic process for reconstructing control logic from encrypted programming traffic was developed and demonstrated.
- **Data Collection Tools:** A series of scripts to ease the setup of a data collection framework to test and evaluate an ADS against numerous network intrusion and project scenarios.
- **Qualitative Analysis:** Strong evidence that Task Time can be used to detect the additional burden on end devices caused by network intrusions.

Chapter III highlighted the Protect function and detailed the shortcomings of physical backdoors in remote devices. Several recommendations were made to help mitigate the risks of inadequate physical protections and ensure a compromised device does not remain on the network to create misoperations.

Chapter IV discussed the creation of a forensic process for encrypted programming traffic and additional implementations to maintain present analytical capabilities as ICS communications move away from plaintext and use more secure methods.

Chapters V and VI described the creation and evaluation of an ADS for an SEL-3505 RTAC using task time as the data feature. It explored the ability to detect network intrusions using this data by employing three separate statistical tests. While used previously to detect logic and firmware changes, the Permutation test as presented was found to be unable to discriminate between normal variation in device behavior and the burden created by network intrusions having an unacceptably high FPR overall. The Mann-Whitney U Test and Kolmogorov-Smirnov show potential, failing to meet the desired performance thresholds on only a few experimental treatments. The KS test was able to detect the network intrusions in all trials and when the performance associated with the Rev 146 High Task Time trials was removed, the KS test had only 44 false positives out of 450 baseline to baseline comparisons, a strong showing for a proof of concept test. A continuous monitoring scheme based on the CUSUM was evaluated using the same collected data sets. With comparable detection rates, lower computational requirements, and lower detection latency the CUSUM is an extremely viable candidate to be employed in numerous system scenarios.

The RTAC has several security features, but is not currently designed to detect the feasible network intrusions that were tested in this research. ARP spoofing in the absence of a network-based IDS will go undetected, and leaves the potential for damaging effects. While the ADS approach is not the only alternative, the use of task time as a data feature for an ADS shows significant promise to be able to detect network intrusions. This system can be implemented on existing installations without the addition of any new devices by being deployed in PLC function blocks. In systems were a network based IDS already exists the host based IDS can be use as part of a defense-in-depth strategy providing overlapping security controls.

7.3 Future Work

Further research should be conducted to help find vulnerabilities and mitigations for the RTAC and additional SEL protective equipment. Finding should be used to examine devices that share the same hardware platform such as the SEL-3622 Security Gateway. The encapsulation of the intrusion detection algorithms and subsequent deployment on the RTAC allows for further research against different classes of intrusions such as processes that are able to bypass the white listing solution or the unauthorized access of modbus servers.

7.4 Conclusion

Individual devices should not be below the examination of security professionals. Dispelling the obscurity of devices that underpin the Critical Infrastructure of our nation should be championed. Critical Infrastructure will continue to be a target for our adversaries and we must rise to the occasion to ensure the prosperity of our country.

Appendix A. RTAC Firmware Update Batch Script

```
@ECHO OFF

Echo Start: %date% %time%
REM Start the RTAC AcSELerator Process
acrtaccmd start


ECHO logging into database: %date% %time%
acrtaccmd login DatabaseUser -p Password

REM Unlocking the project ensures that it is able to send the
    project
acrtaccmd unlock TaskTimer_Complex_R146

ECHO Upgrading Firmware: %date% %time%
acrtaccmd upgradefirmware -p Password 192.168.2.2 RTACuser C:\
    XXXX\XXXXX\XXXXX\XXXX\SEL-3505-3-R146-V0-Z000002-D20200224.
    upg

ECHO Connecting: %date% %time%
acrtaccmd connect 192.168.2.2 RTACuser -p RTACPassword -n
    TaskTimer_Complex_R146 -s

REM Cleanup the processes and disconnect the codesys
    connection
acrtaccmd disconnect
acrtaccmd close
acrtaccmd stop

ECHO Starting Data Collection: %date% %time%
REM Launch the data collection script passing in the
    neccessary naming arguement
python DataCollection.py %1
ECHO Finished: %date% %time%
```

Appendix B. Data Collection Python Script

```python
from pymodbus.client.sync import ModbusTcpClient as MC
import pandas as pd
import time
from datetime import datetime
import argparse


#Parse out the sent arguements
parser = argparse.ArgumentParser()
parser.add_argument('revision', type=str, nargs =1, metavar='c')
args=parser.parse_args()

#Create a data frame to store the collected data
df = pd.DataFrame(columns=['received', 'transmitted', 'ports_active', 'connections', 'bytes_received', 'bytes_sent', 'average', 'cpu burden', 'cpu burden average', 'system time'])

#Wait 60 seconds for device to reach steady state
time.sleep(60)

for i in range(0,30000):
    try :
        #Connect to Modbus Server
        client = MC('192.168.2.2')
        #Read Input Registers From Server
        result = client.read_input_registers(0,24)
        #Close Connection
        client.close()

        #Bit shift modbus registers to be proper magnitude
        received = (result.getRegister(0) << 16) + result.getRegister(1)
        transmitted = (result.getRegister(2) << 16) + result.getRegister(3)
        ports_active = (result.getRegister(4) << 16) + result.getRegister(5)
        connections = (result.getRegister(6) << 16) + result.getRegister(7)
        bytes_received = (result.getRegister(8) << 16) + result.getRegister(9)
        bytes_sent = (result.getRegister(10) << 16) + result.getRegister(11)
        last_tasktime = (result.getRegister(12) << 16) + result.getRegister(13)
        average_tasktime = (result.getRegister(14) << 16) + result.getRegister(15)
        memory_in_use = (result.getRegister(16) << 16) + result.getRegister(17)
        flash_used = (result.getRegister(18) << 16) + result.getRegister(19)
        cpuburden = (result.getRegister(20) << 16) + result.getRegister(21)
        cpuaverage = (result.getRegister(22) << 16) + result.getRegister(23)

        #Add Data to Data Frame
        d = {'received' : received, 'transmitted' : transmitted, 'ports_active' : ports_active, \
             'connections' : connections, 'bytes_received' : bytes_received, 'bytes_sent' : bytes_sent, \
             'last tasktime' : last_tasktime, 'average' : average_tasktime, 'memory in use' : memory_in_use,\
             'flash in use' : flash_used,  'cpu burden' : cpuburden, 'cpu burden average' : cpuaverage, \
             'system time' : datetime.now() }

        df = df.append(d, ignore_index=True)

    except:
    #Error handling for network transmission errors
        d ={'received' : -1, 'transmitted' : -1, 'ports_active' : -1, \
             'connections' : -1, 'bytes_received' : -1, 'bytes_sent' : -1, \
             'last tasktime' : -1, 'average' : -1, 'memory in use' : -1,\
             'flash in use' : -1,  'cpu burden' : -1, 'cpu burden average' : -1, \
             'system time' : datetime.now() }
        print("No response")

    #Wait 1 ms before requesting again
    time.sleep(.001)


#Create .csv with all of collected data using passed in trial argument as file name
file_name = args.revision[0] +".csv"
print(file_name)
df.to_csv(file_name)
```

Bibliography

1. "Critical Infrastructure Sectors," 2020. [Online]. Available: https://www.cisa.gov/critical-infrastructure-sectors [Accessed: 2020-06-12]

2. B. Obama, "Presidential policy directive – critical infrastructure security and resilience," February 2013.

3. K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to Industrial Control Systems (ICS) Security," *Special Publication (NIST SP)*, no. 800-82 Rev, 2015. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-82r2

4. D. Trump, "Executive order on securing the information and communications technology and services supply chain," May 2019.

5. FireEye, "Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor," December 2019. [Online]. Available: https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html

6. Cybersecurity & Infrastructure Security Agency, "Russian government cyber activity targeting energy and other critical infrastructure sectors." [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/TA18-074A

7. M. Barrett, "Framework for improving critical infrastructure cybersecurity," in *Proceedings of the Annual ISA Analysis Division Symposium*, vol. 535, 2018, pp. 9–25. [Online]. Available: https://doi.org/10.6028/NIST.CSWP.04162018

8. S. J. Chapman, *Electric Machinery and Power System Fundamentals*, international edition ed. McGraw-Hill, 2002.

9. "Glossary of terms used in nerc reliability standards." [Online]. Available: https://www.energy.gov/sites/prod/files/2017/09/f36/NERC%20Glossary.pdf

10. J. J. Bian, A. D. Slone, and P. J. Tatro, "Protection system misoperation analysis," in *2014 IEEE PES General Meeting — Conference Exposition*, 2014, pp. 1–5.

11. M. Zeller, "Myth or reality — does the aurora vulnerability pose a risk to my generator?" in *2011 64th Annual Conference for Protective Relay Engineers*, 2011, pp. 130–136.

12. Schweitzer Engineering Laboratories, "Customer Highlights — Schweitzer Engineering Laboratories," 2020. [Online]. Available: https://selinc.com/solutions/success-stories/ [Accessed: 2020-12-07]

13. ——, "SEL ranked top protective relay manufacturer in industry survey — Schweitzer Engineering Laboratories," 2019. [Online]. Available: https://selinc.com/company/news/126347/ [Accessed: 2020-12-07]

14. A. Nochvay, "Security research : CODESYS Runtime , a PLC control framework," Kaspersky, Tech. Rep., 2019.

15. Schweitzer Engineering Laboratories, *SEL-3505 Real-Time Automation Controller Instruction Manual*, Schweitzer Engineering Laboratories, 2020.

16. D. Kite, "Leveraging Security-Using the SEL RTAC's Built-In Security Features," SEL, Tech. Rep., 2016.

17. I. Ahmed, S. Obermeier, M. Naedele, and G. G. Richard, "SCADA systems: Challenges for forensic investigators," *Computer*, vol. 45, no. 12, pp. 44–51, 2012.

18. S. Senthivel, I. Ahmed, and V. Roussev, "SCADA network forensics of the PCCC protocol," in *DFRWS 2017 USA - Proceedings of the 17th Annual DFRWS USA*. Digital Forensic Research Workshop, 2017, pp. S57–S65.

19. S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, and V. Roussev, "Denial of engineering operations attacks in industrial control systems," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 319–329. [Online]. Available: https://doi.org/10.1145/3176258.3176319

20. A. Keliris and M. Maniatakos, "ICSREF: A framework for automated reverse engineering of industrial control systems binaries," in *26th Annual Network and Distributed System Security Symposium*. The Internet Society, 2019.

21. S. A. Qasim, J. Lopez, and I. Ahmed, "Automated reconstruction of control logic for programmable logic controller forensics," in *Information Security*, Z. Lin, C. Papamanthou, and M. Polychronakis, Eds. Cham: Springer International Publishing, 2019, pp. 402–422.

22. R. A. Bridges, T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, and Q. Chen, "A survey of intrusion detection systems leveraging host data," *ACM Computing Surveys*, vol. 52, no. 6, 2019. [Online]. Available: https://doi.org/10.1145/3344382

23. R. Mitchell, I.-R. Chen, and V. Tech, "A Survey of Intrusion Detection Techniques for Cyber-Physical Systems," *ACM Comput. Surv*, vol. 46, 2014. [Online]. Available: http://dx.doi.org/10.1145/2542049

24. S. Dunlap, J. Butts, J. Lopez, M. Rice, and B. Mullins, "Using timing-based side channels for anomaly detection in industrial control systems," *International Journal of Critical Infrastructure Protection*, vol. 15, pp. 12–26, dec 2016.

25. D. Formby and R. Beyah, "Temporal execution behavior for host anomaly detection in programmable logic controllers," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1455–1469, 2020.

26. M. Niedermaier, J. O. Malchow, F. Fischer, D. Marzin, D. Merli, V. Roth, and A. von Bodisco, "You snooze, you lose: Measuring PLC cycle times under attacks," in *12th USENIX Workshop on Offensive Technologies, WOOT 2018, co-located with USENIX Security 2018*, 2018.

27. C. Vargas Martinez and B. Vogel-Heuser, "A host intrusion detection system architecture for embedded industrial devices," *Journal of the Franklin Institute*, vol. 358, no. 1, pp. 210 – 236, 2021.

28. T. Alves, R. Das, and T. Morris, "Embedding Encryption and Machine Learning Intrusion Prevention Systems on Programmable Logic Controllers," *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 99–102, sep 2018.

29. Schweitzer Engineering Laboratories, *SEL-3505/SEL-3505-3 Real-Time Automation Controller*, Schweitzer Engineering Laboratories, 2020.

30. ——, *Modern Water Delivery Solutions*, Schweitzer Engineering Laboratories, 2020.

31. C. Snyder and P. Szoldra, "Watch hackers break into the us power grid," May 2016. [Online]. Available: https://www.youtube.com/watch?v=pL9q2lOZ1Fw&index=4&list=PLh9vABlZOSlONaWbv88Tun0YsaduoaM1H

32. J. Staggs, D. Ferlemann, and S. Shenoi, "Wind farm security: attack surface, targets, scenarios and mitigation," *International Journal of Critical Infrastructure Protection*, vol. 17, pp. 3–14, jun 2017.

33. M. Cook, I. Stavrou, S. Dimmock, and C. Johnson, "Introducing a forensics data type taxonomy of acquirable artefacts from programmable logic controllers," in *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2020, pp. 1–8.

34. N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet Dossier," Symantec, Tech. Rep. February, 2011. [Online]. Available: https://www.symantec.com/content/en/us/enterprise/media/security{_}response/whitepapers/w32{_}stuxnet{_}dossier.pdf

35. I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev, "Programmable logic controller forensics," *IEEE Security Privacy*, vol. 15, no. 6, pp. 18–24, 2017.

36. C. Paar and J. Pelzl, *Understanding Cryptography.* Springer, 2010.

37. S. East, J. Butts, M. Papa, and S. Shenoi, "A taxonomy of attacks on the DNP3 protocol," in *IFIP Advances in Information and Communication Technology*, vol. 311, 2009, pp. 67–81.

38. D. Pliatsios, P. Sarigiannidis, T. Lagkas, and A. G. Sarigiannidis, "A Survey on SCADA Systems: Secure Protocols, Incidents, Threats and Tactics," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 3, pp. 1942–1976, apr 2020.

39. C. Parian, T. Guldimann, and S. Bhatia, "Fooling the Master: Exploiting Weaknesses in the Modbus Protocol," *Procedia Computer Science*, vol. 171, pp. 2453–2458, jan 2020.

40. L. Martín-Liras, M. A. Prada, J. J. Fuertes, A. Morán, S. Alonso, and M. Domínguez, "Comparative analysis of the security of configuration protocols for industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 19, pp. 4–15, 2017.

41. R. R. Wilcox, *Applying Contemporary Statistical Techniques.* Burlington: Academic Press, 2003, pp. 237 – 284.

42. S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018.

43. Y. Dodge, *The Concise Encyclopedia of Statistics.* New York, NY: Springer, 2008.

44. H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 03 1947. [Online]. Available: https://doi.org/10.1214/aoms/1177730491

45. N. Nachar, "The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution," *Tutorials in Quantitative Methods for Psychology*, vol. 4, no. 1, pp. 13–20, 2008.

46. J. Bernacki and G. Kołaczek, “Anomaly Detection in Network Traffic Using Selected Methods of Time Series Analysis,” *International Journal of Computer Network and Information Security*, vol. 7, no. 9, pp. 10–18, 2015.

47. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, ..., and SciPy 1.0 Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.

48. Dandan Zhu and Yan Cui, “Understanding random guessing line in roc curve,” in *2017 2nd International Conference on Image, Vision and Computing (ICIVC)*, 2017, pp. 1156–1159.

49. E. S. Page, “Continuous inspection schemes,” *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954. [Online]. Available: http://www.jstor.org/stable/2333009

50. D. Montgomery, *Introduction to Statistical Quality Control*, 6th ed. John Wiley & Sons, 2009.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 25–03–2021	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From — To)* Jun 2019 — Mar 2021

4. TITLE AND SUBTITLE

Anomaly Detection and Encrypted Programming Forensics for Automation Controllers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Mellish, Robert W., 1st Lt, USAF

5d. PROJECT NUMBER

21G171

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-21-M-060

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

Oak Ridge National Labs
1 Bethel Valley Rd,
Oak Ridge, TN 37830
Attn: Dr. Mason Rice
COMM: 865-576-2452
Email: ricemj@ornl.gov

10. SPONSOR/MONITOR'S ACRONYM(S)

ORNL

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

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

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Securing the critical infrastructure of the United States is of utmost importance in ensuring the security of the nation. To secure this complex system a structured approach such as the NIST Cybersecurity framework is used, but systems are only as secure as the sum of their parts. Understanding the capabilities of the individual devices, developing tools to help detect misoperations, and providing forensic evidence for incidence response are all essential to mitigating risk. This thesis examines the SEL-3505 RTAC to demonstrate the importance of existing security capabilities as well as creating new processes and tools to support the NIST Framework. The research examines the potential pitfalls of having small-form factor devices in poorly secured and geographically disparate locations. Additionally, the research builds a data-collection framework to provide a proof of concept anomaly detection system for detecting network intrusions by recognizing the change in task time distribution. Statistical tests distinguish between normal and anomalous behaviour. The high true positive rates and low false positive rates show the merit of such an anomaly detection system. Finally, the work presents a network forensic process for recreating control logic from encrypted programming traffic.

15. SUBJECT TERMS

ICS, Anomaly Detection, RTAC, Physical Security, Cybersecurity

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Scott Graham, AFIT/ENG
U	U	U	UU	86	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-6565 x4581; scott.graham@afit.edu