Chapter 5

ANOMALY DETECTION IN AUTOMATION CONTROLLERS

Robert Mellish, Scott Graham, Stephen Dunlap and Patrick Sweeney

Abstract Cyber-physical systems incorporate powerful devices that are used to monitor and control physical processes. These devices along with collectable statistics can be leveraged as sensors for network-based and host-based anomaly detection. Host-based anomaly detection can be used in a defense-in-depth strategy to complement traditional network-based anomaly detection systems as well in systems for which network-based options are infeasible due to their operating environments.

This chapter discusses the development of an anomaly detection system for a SEL-3505 RTAC programmable logic controller using the recommended IEC 61131 programming tools. The required device statistics are harvested by creating a Modbus server on the test system and polling the server to retrieve data. The collected data is used to create a representative fingerprint for the associated task. When the measured behavior differs from the fingerprint, an anomaly is detected and an alarm is raised. This approach is flexible and easily implemented in existing installations. The performance of the anomaly detection system is evaluated against several network-based attacks across multiple firmware revisions and project types. Recommendations are made to improve anomaly detection performance.

Keywords: Anomaly detection, automation controllers

1. Introduction

The risks to industrial control systems have increased as their networks have become more interconnected with corporate networks and the open Internet. While advances have been made to secure industrial control networks by adapting traditional information technology network security solutions and developing specialized solutions such as

Published by Springer Nature Switzerland AG 2022
J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XV, IFIP AICT 636, pp. 97–118, 2022.
https://doi.org/10.1007/978-3-030-93511-5_5

process-aware detection methods, much more needs to be done to bolster the resilience and security of these vital assets.

This research explores the utilization of end devices such as programmable logic controllers as sensors for network intrusion detection in addition to their traditional process monitoring and control roles. Specifically, it examines the effects of network intrusions on the task execution time of a SEL-3505 real-time automation controller (RTAC) and the potential efficacy of an anomaly detection system (ADS) that engages the task execution time as its data feature.

The research has three contributions. The first is the analysis of the limitations of a real-time automation controller implementation that allow for network intrusions. The second is strong experimental evidence of the ability to detect network anomalies by tracking task times. The third is a data collection framework that evaluates discrimination strategies with multiple data features for the real-time automation controller.

2. SEL-3505 RTAC Device

The SEL-3505 RTAC used in this research is produced by Schweitzer Engineering Laboratories, a U.S. manufacturer of power protection and automation equipment. Schweitzer Engineering Laboratories does not provide information about the numbers of devices it has sold and installed. However, success stories posted on the company website [19] reveal that its devices protect power systems in the countries of Georgia and Grand Cayman, control microgrids in several U.S. universities, mitigate arc flashes at North American mining companies and manage power in refineries. A recent study of the worldwide protective relay market ranks Schweitzer Engineering Laboratories as the number one relay manufacturer in North America [18]. Although the exact numbers are elusive, it is clear that Schweitzer Engineering Laboratories devices are used throughout the global critical infrastructure.

The SEL-3505 RTAC is marketed as an electric substation controller. It combines physical input/output (I/O) with flexible IEC 61331 control logic and provides several serial ports and dual Ethernet ports. Schweitzer Engineering Laboratories provides communications libraries that enable SEL-3505 RTAC devices to interact with numerous other devices. A SEL-3505 RTAC can also be used as a data concentrator, communicating with multiple legacy devices over serial protocols and converting the data streams to Ethernet-based communications. These features render SEL-3505 RTAC devices popular for use in industrial control systems as well as attractive targets for attackers.

A SEL-3505 RTAC incorporates an embedded Linux host with applications that provide programmable logic controller functionality. A web server is provided for configuration and management. User accounts can be created and diagnostics executed from a password-protected web page. The diagnostics include checking the control logic status and performing factory resets if necessary. The web interface uses a PostgreSQL back-end that supports functions such as downloading new project files and making changes to the device firewall. Project files are created using proprietary Windows-based AcSELerator RTAC engineering software. The AcSELerator software, which is undocumented, creates a compiled binary that is executed by the widely-used CODESYS programmable logic controller framework [14]. The CODESYS runtime IEC 61131 logic engine enables the Linux host to act as a generic programmable logic controller. Understanding these interfaces is crucial to securing the devices and the networks in which they reside. Detecting anomalies in their operation can help recognize potential intrusions as they are taking place.

A SEL-3505 RTAC has several security features. These include application whitelisting that protects against rootkits, built-in denial-of-service detection, system priority readjustment and functionality that enables all configured sequence of events data tags to be available in a syslog client [8, 20]. Whitelisting prevents the execution of unauthorized processes and denial-of-service detection supports responses to brute force attacks on the network stack. However, the SEL-3505 RTAC does not provide detection functionality for identifying misuse or subtle exploitation of authorized applications.

To address this limitation, this research proposes the addition of an anomaly detection system to identify network intrusions. If the intrusion detection system is implemented in IEC 61131 function blocks, it could be generalized across other programmable logic controllers, providing end-device protection in any number of industrial control networks.

3. Anomaly Detection System

At an abstract level, a host-based anomaly detection system can be distilled into a workload, system outputs, system parameters and decision algorithm that work in concert with a data collector at the heart of the anomaly detection system. Figure 1 shows an anomaly detection system comprising these components.

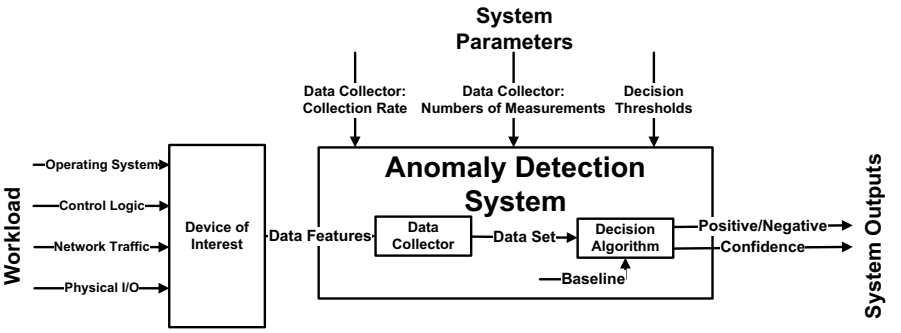


Figure 1. Anomaly detection system.

3.1 Workload

The workload encompasses actions that the hardware must complete on a continual basis. When completing these actions, the hardware produces measurable responses that can be leveraged in anomaly detection. Some actions are mandated by the manufacturer and are immutable and transparent to end users. Other actions are created by end users to fulfill their process requirements. The final category of workload includes the burden imposed on the hardware by the physical process it monitors and controls, and the network traffic it receives.

The workload is created by the following components:

- **Operating System:** While programmable logic controllers and other industrial devices may be portrayed as user logic running on bare metal, in practice there is firmware that handles the network stack and communications needed to program the devices. These firmware functions compete for resources and can slow down process control tasks even if they still meet the real-time requirements. Popular operating systems include OS-9, VxWorks and Linux. Modifications to an operating system, whether malicious or manufacturer mandated, can cause changes to the workload. Some workload changes have been shown to be detectible by current anomaly detection methods [4, 6].

- **Control Logic:** The control logic is implemented by user-created functions that a programmable logic controller continuously executes to monitor and control its assigned physical process. The control logic for a complex process often constitutes the bulk of the workload. The detection of malicious control logic modifications has been the subject of considerable research [4, 6].

- **Network Traffic:** The packets sent and received by a device impose a workload on the device. Some devices act as communications gateways that connect to numerous devices with a single upstream device. Others simply translate physical process information to a digital representation that is displayed on a human-machine interface. Engineering functions such as programming, performing diagnostics and retrieving system logs impose additional workload. The effects of the workload can be seen in the task times of devices that are under duress from network scans [13].

  Network intrusions are anomalous actions that are the focus of anomaly detection. An intrusion places additional burden on a device by creating network traffic, executing additional processes on the operating system and/or exfiltrating data. An anomaly detection system must discern between the normal workload and the burden imposed by intrusions through careful selection of features and analysis tools.

- **Physical Input/Output:** Programmable logic controllers are responsible for monitoring and controlling physical processes using specialized sensors and actuators. In some programmable logic controllers, input/output data processing has been shown to cause no increases in task times [4] because inputs vary in frequency. This may indicate that input/output is being handled by a scheduled task or by a secondary processor. A SEL-3505 RTAC has a field programmable gate array (FPGA) that conditions binary and analog input and output data, offloading the task from the processor. However, due to the limitations of monitoring field programmable gate arrays, the effects of physical input/output variations on task times are not explored in this research.

3.2 System Outputs

The outputs of an anomaly detection system are the algorithm decision and the confidence level associated with the decision. A common set of performance metrics is required to compare different anomaly detection systems. Historically, the following metrics have been used to compare the performance of anomaly detection systems [4, 11]:

- **True Positive Rate (TPR):** The true positive rate is the rate at which an anomaly detection system correctly identifies an anomaly.

- **True Negative Rate (TNR):** The true negative rate is the rate at which an anomaly detection system correctly identifies a normally-behaving system.

- **False Positive Rate (FPR):** The false positive rate is the rate at which an anomaly detection system incorrectly identifies a normally-behaving system as an anomaly.
- **False Negative Rate (FNR):** The false negative rate is the rate at which an anomaly detection system fails to identify an anomaly.

The false positive rate is the type I error of an anomaly detection system. If the rate is too high, alarms become a nuisance and may be disregarded when true anomalies occur. The false negative rate expresses the frequency at which anomalies go undetected. Keeping this rate low is the principal objective when designing an anomaly detection system.

In addition to the error rates, the following factors must be considered when designing an anomaly detection system:

- **Detection Latency:** Although it may not be used frequently, detection latency is an important metric for anomaly detection systems. The detection latency is the length of time required to detect an anomaly. Its infrequent use may be attributed to the need for a more mature anomaly detection system than those typically discussed in the literature.
- **System Overhead:** Industrial control devices have limited computational and storage resources, and must meet strict real-time operational constraints. Therefore, a deployed anomaly detection system must be as lightweight as possible to eliminate negative impacts on the controlled physical process. System overhead may be measured as an increase in task time or processor burden percentage.

3.3 Tuning Parameters

Tuning parameters are properties of an anomaly detection system that can be changed to increase its overall performance. The tuning parameters are:

- **System Features:** Depending on the privileges provided to an anomaly detection system, there are variety of features to consider. These include statistics such as task time, CPU burden, numbers of network packets sent and received, system RAM in use, and more. Certain features may lend themselves better to detecting specific attacks and an anomaly detection system designer must carefully identify, justify and defend the chosen features. Previous research efforts have used task time as a feature [4, 6]. This

research seeks to understand the intrusions that can be discerned using the previously-employed features.

- **Data Collection Rate:** The data collection rate can directly affect the sensitivity of an anomaly detection system. If attacks are ephemeral, a slow polling rate could miss anomalies. If data polling is too frequent, the burden on the device becomes significant and the real-time operational requirements will not be met.
- **Number of Data Points:** The number of consecutive data points needed to make a decision and the data collection rate directly correlate with detection latency. Depending on the time complexity of the decision algorithm, an increase in the number of data points could have an undesirable increase in detection latency.

3.4 Decision Algorithm

A decision algorithm is central to an anomaly detection system and is paramount to its overall success. A decision algorithm uses the collected data to make an assessment about the operation of the device of interest. Several methods have been employed to detect anomalies. Depending on the data features collected, different decision algorithms have been adopted. Vargas et al. [23] have monitored RAM usage using simple moving averages. Formby and Beyah [6] have used a cumulative sum algorithm on task time to detect modifications to control logic. Dunlap et al. [4] have also considered task time, but they selected a permutation test to discern changes. Alves et al. [1] have leveraged an embedded machine learning intrusion detection system in conjunction with OpenPLC to detect network anomalies by inspecting TCP headers. To facilitate a portable solution that could be implemented using IEC 61131 control logic, this research engages statistical methods instead of machine learning approaches.

4. Experimental Design

This section describes the experimental design, including the experimental factors, data collection, discriminator selection and system evaluation.

4.1 Experimental Factors

The proposed anomaly detection system employs functions native to the SEL-3505 RTAC as well as supplementary functions written in Python that could be integrated by the manufacturer into a future SEL-3505 RTAC firmware release or they could be programmed in the con-

Table 1. Experimental factors.

Firmware Revision	Project File Type	Network Intrusion
R145	Low task time	Baseline
R146	High task time	ARP spoofing
R147		PostgreSQL queries
		CODESYS connection

trol logic engine by a control engineer. In this research, an experimental treatment comprises a firmware revision, control logic project file type and network intrusion.

Table 1 shows the experimental factors. Selecting one factor from each column yielded 24 ($= 3 \times 2 \times 4$) combinations. These variations were intended to capture the effectiveness of an anomaly detection system across various workloads and intrusions. Ten trials were conducted for each experimental treatment, leading to 240 total trials. The order in which trials were conducted was determined by creating a list of all the trials and randomizing their order.

Firmware Revisions. Previous research has shown that monitoring task times in programmable logic controllers can identify firmware modifications [4, 6]. To verify that this research could be generalized to the SEL-3505 RTAC, three firmware revisions released by Schweitzer Engineering Laboratories were considered, R145, R146 and R147.

SEL-3505 RTAC Project File Types. Two project files were considered, one a low task time project and the other a high task time project. Project 1 was a limited-functionality project with an average task time of approximately 1,300 μs. It contained only the control logic needed for SEL-3505 RTAC operation along with a single Modbus server for data collection. Such a project would be unlikely to be deployed in an operational environment, because even a data concentrator would have additional network connections and data mapping logic. Nevertheless, the project realized the lowest possible task time while still providing the data needed for anomaly detection.

Project 2 was much more complex with an average task time of approximately 9,000 μs. The long task time was realized by performing numerous complex mathematical operations that used pseudorandom inputs provided by SEL-3505 RTAC's SELRand library and time-varying inputs such as the numbers of bytes sent and received by the Ethernet ports.

The two projects represent the extremes in task times. Future research will examine the use of branching projects whose complexity varies based on the current states of the processes being monitored.

Network Intrusions. In order to assess the viability of using task time as a data feature for intrusion detection, a number of feasible network intrusions or attacks had to be devised against the SEL-3505 RTAC. Therefore, the implementation details of the SEL-3505 RTAC were carefully explored to understand features that attackers could exploit. Successful exploits result in potentially-detectible footholds; identifying the exploits and mechanisms that enable them is the purpose of an anomaly detection system.

The following intrusion mechanisms were employed in the experiments for their demonstrated viability as attack vectors or as manifestations of established footholds:

- **Baseline:** The baseline treatment used the network traffic necessary to harvest anomaly detection system data. This provided an experimental backdrop for detecting network intrusions. Repeated baseline trials were evaluated against each other to determine the false positive rate for each algorithm.

- **ARP Spoofing:** ARP spoofing is a standard mechanism for performing man-in-the-middle attacks that are often used to demonstrate the vulnerabilities of industrial control devices and networks [5, 15, 16, 22]. The exploit leverages the lack of authentication in the address resolution protocol (ARP) to send unsolicited resolution responses containing false information about the network topology. The false link layer information causes network traffic to be misdirected to a network attacker.

  Although the use of encrypted traffic would severely limit an attacker's ability to collect information or inject packets, common industrial control protocols still rely on legacy implementations and their communications are largely in plaintext. This lack of security enables a variety of modification attacks against common protocols such as Modbus [15] and DNP3 [5]. The lack of encryption in the configuration protocols of devices also poses security risks to industrial control networks with regard to ARP spoofing [10, 21].

  Improper configuration of a SEL-3505 RTAC can pose additional risks. The device generates a self-signed certificate upon installing a new firmware revision. The certificate is used to encrypt the HTTPS connection for initial configuration as well as PostgreSQL

traffic. If the certificate is not replaced, an adversary could perform a man-in-the-middle attack and impersonate the SEL-3505 RTAC using a previously-compromised certificate from another SEL-3505 RTAC. This would enable an attacker to harvest credentials from the supposed secure connection and inject arbitrary modifications into the system configuration or programming traffic.

- **PostgreSQL Queries:** A SEL-3505 RTAC uses a PostgreSQL database to manage the control logic on the device and to interface with the operating system. Engineering functions are programmed as SQL queries that are callable from the web interface. These functions include changing the IP 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 compared with programmable logic controller patching by manufacturers. However, a burden is placed on manufacturers to incorporate the most up-to-date database versions in their device firmware and for end users to patch their own devices.

  If the PostgreSQL database is exploited or database credentials are compromised, an attacker can launch denial-of-service attacks because a SEL-3505 RTAC can be restarted by a database query. A system restart is an easily-detectible anomaly but subtler attacks are also possible. The PostgreSQL queries employed for foothold emulation during the experiments included reads of the control logic project from the SEL-3505 RTAC. The exfiltration of the control logic is a major reconnaissance activity by an attacker as it provides in-depth knowledge of the physical process controlled by the SEL-3505 RTAC as well as the devices that communicate with the SEL-3505 RTAC.

- **CODESYS Connection:** The CODESYS runtime application executing on a SEL-3505 RTAC is responsible for industrial control protocol communications and control logic operation. It uses the standard port 1217 for engineering functions. The functions include uploading and downloading control logic programs, monitoring real-time control logic values, forcing control logic values for debugging purposes, and stopping and starting control logic execution. If enabled, the connection also allows access to the SEL-3505 RTAC filesystem.

  Vulnerabilities related to the use of CODESYS as a programmable logic controller framework have been documented [14] and exploited to decompile the control logic [7]. While CODESYS has

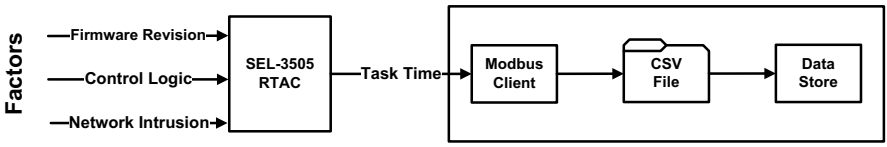


Figure 2. Experimental data collection system.

recently implemented an authentication mechanism to establish connections, this feature is not activated in a SEL-3505 RTAC and credentials are not required on port 1217 itself. Instead, TCP connections to port 1217 are blocked by default and the port can be opened and closed by accessing the PostgreSQL database.

SEL-3505 RTAC filesystem access is a concern because it uses TLS 1.1 for PostgreSQL traffic and web server configuration. The private server key used for encryption can be exfiltrated from a SEL-3505 RTAC using a CODESYS connection and then used to decrypt any communication and extract information such as a username and password. Additionally, an attacker could insert or modify SEL-3505 RTAC files that may enable further exploitation.

In order to emulate this network intrusion, a passive CODESYS connection was kept active during data collection. The passive connection had the minimum workload generated by a CODESYS connection with no files being read from the SEL-3505 RTAC and no process values changed from the engineering workstation.

4.2 Data Collection

A total of 30,000 task time samples were collected for each trial. The data was collected by creating a Modbus server on the SEL-3505 RTAC and polling the server via 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 modifications, as well as its sensitivity to changes in network traffic as described in previous research [4, 6, 13].

Each dataset was saved as a CSV file for future analysis using the selected decision algorithms. Future research will focus on real-time detection using a continual data collection technique and eventually employ a SEL-3505 RTAC function block that goes beyond Modbus data collection. Figure 2 shows a notional data collection process and the experimental factors.

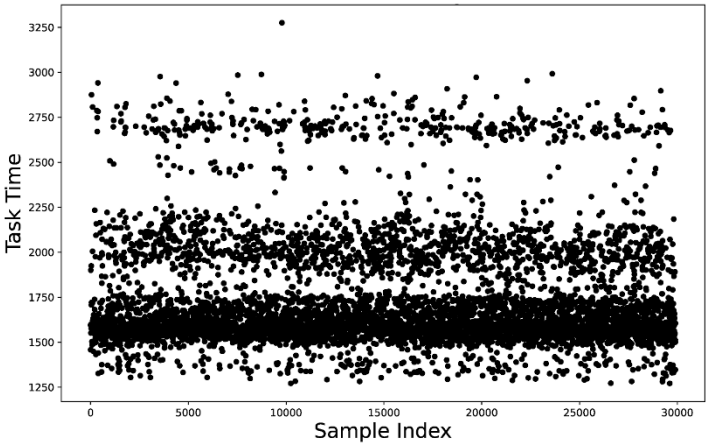


Figure 3. Scatter plot of task time distribution.

4.3 Discriminator Selection

Pilot studies confirmed previous research results that task times do not have normal distributions [4, 13]. Figure 3 shows a scatter plot of the sampled task times from a pilot study with no network intrusions. Three distinct clusters are discernible and no apparent correlation exists between when a sample was taken and its associated task time. The three clusters are seen in more detail in the kernel density estimation plot in Figure 4.

Because the task time population does not have a normal distribution, the following three non-parametric statistical tests were selected as discriminators to evaluate the efficacy at detecting network intrusions using task time:

- **Permutation Test:** This re-sampling test investigates the probability that two sample populations are from the same distribution. The test can be performed without making any assumptions about the sample distributions [25]. It has been used to detect control logic and firmware modifications [4]. The `mlxtend` Python module was used to perform this test [17].

- **Mann-Whitney U Test:** This non-parametric test explores the null hypothesis that one population is stochastically larger than the other [3, 9]. While it is commonly used in the behavioral sciences, the test has been employed to evaluate statistical differences in sampled and forecasted network traffic [2, 12]. The test was con-

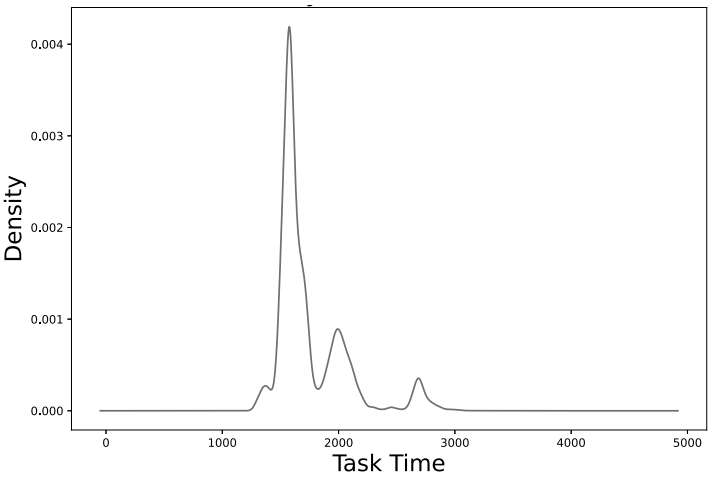


Figure 4. Kernel density estimation of task time.

ducted on task time data using the `scipy.stats.mannwhitneyu` Python function [24].

- **Kolmogorov-Smirnov Test:** This non-parametric goodness-of-fit test probes the hypothesis that two independent samples are drawn from the same distribution. The test has the advantage of considering the entirety of a distribution function instead of just the difference in a test statistic. For the test to be valid, the task time distribution is assumed to be continuous. The Kolmogorov-Smirnov test statistic is the maximum absolute difference between two distribution functions [3] and has been used to detect control logic modifications [6]. The test was conducted using the `scipy.stats.kstest` Python function [24].

4.4 System Evaluation

To test each discriminator, the datasets must be separated by firmware revision and project type. This reduces the buoying effect that inflates the true positive rate when testing a low task time dataset against a high task time dataset. Next, each intrusion-free dataset is tested against all the other datasets in its (firmware revision, project type) subset.

Figure 5 shows the experimental analysis process. The results of the tests are used to compute the true positive and false positive rates for each decision algorithm. The process is repeated, varying the decision threshold to generate a receiver operating characteristic (ROC) curve

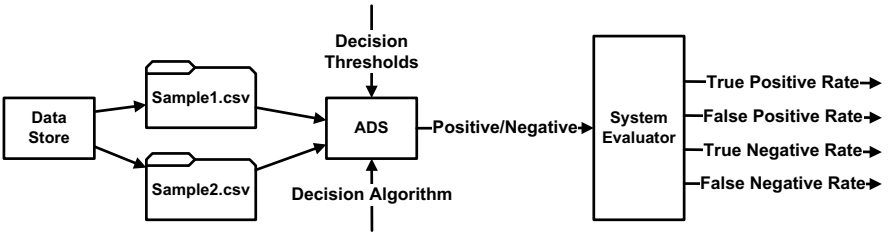


Figure 5. Experimental analysis process.

for each algorithm. The ROC curve helps explore the trade-off between the true and false positive rates [4]. Additionally, the area under the curve metric helps compare the performance of multiple algorithms [26].

5. Experimental Results

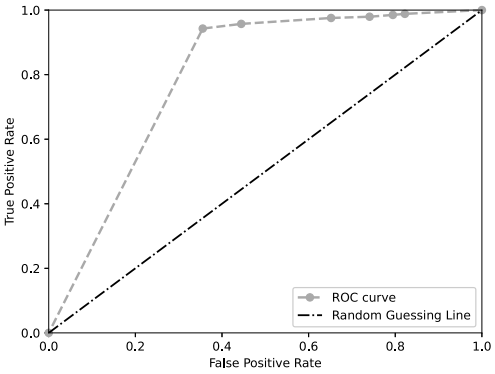
This section presents the anomaly detection results and discusses strategies for improving the anomaly detection rates.

5.1 Anomaly Detection Rates

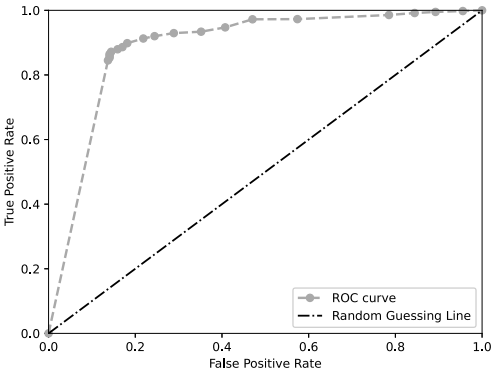
The compiled results from the 240 experimental trials are presented. Figure 6 shows the ROC curves for the three decision algorithms. The area under the curve (AUC) is also displayed for each algorithm. An ideal classifier would have an area under the curve of 1.0. The Mann-Whitney U test has the highest area under the curve of 0.89. The Kolmogorov-Smirnov test has a slightly lower value of 0.88. The permutation test has a value of 0.80. The ROC curves were employed to select decision thresholds for the algorithms to showcase their representative performance.

Tables 2, 3 and 4 show the results of the permutation test, Mann-Whitney U test and Kolmogorov-Smirnov test, respectively. Values shown in bold do not meet the performance thresholds of less than 0.1 for the false positive rate or greater than 0.9 for the combined true positive rate. These performance thresholds are based on previous research in anomaly detection systems [4]. Each (firmware revision, project type) pair is shown to highlight when an algorithm performed poorly overall or if a specific test case was more difficult to detect.

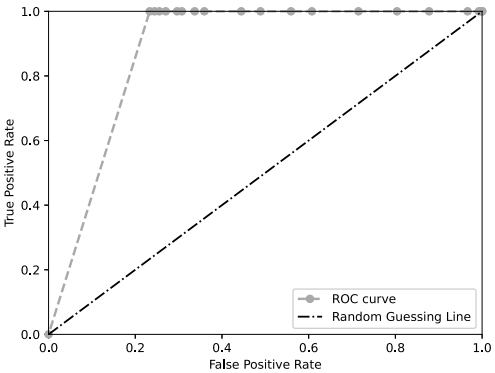
The permutation test failed to meet the false positive rate threshold, but it performed well in detecting network intrusions, with two (firmware revision, project type) pairs just missing the 0.9 cutoff. The Mann-Whitney U test performed well with regard to the overall false positive



(a) Permutation test (AUC = 0.80).



(b) Mann-Whitney U test (AUC = 0.89).



(c) Kolmogorov-Smirnov test (AUC = 0.88).

Figure 6. ROC curves.

Table 2. Permutation test results (decision threshold = 0.0001).

Firmware Revision	Task Time	FPR	TPR			
			Combined	CODESYS	ARP Spoof	PostgreSQL
R145	Low	**0.46**	0.94	0.96	0.86	0.99
R146	Low	**0.22**	0.97	1.00	1.00	0.92
R147	Low	**0.40**	**0.87**	0.81	0.91	0.90
R145	High	**0.20**	1.00	1.00	1.00	1.00
R146	High	**0.69**	**0.90**	0.92	0.92	0.85
R147	High	**0.17**	0.98	0.98	1.00	0.95

Table 3. Mann-Whitney U test results (decision threshold = 1×10^{-14}).

Firmware Revision	Task Time	FPR	TPR			
			Combined	CODESYS	ARP Spoof	PostgreSQL
R145	Low	0.09	0.96	0.99	0.97	0.91
R146	Low	0.00	0.96	1.00	0.90	0.99
R147	Low	0.09	**0.81**	0.81	0.81	0.81
R145	High	0.00	0.97	1.00	0.92	1.00
R146	High	**0.67**	**0.70**	0.70	0.73	0.67
R147	High	0.00	**0.78**	0.74	0.75	0.86

Table 4. Kolmogorov-Smirnov test results (decision threshold = 1×10^{-17}).

Firmware Revision	Task Time	FPR	TPR			
			Combined	CODESYS	ARP Spoof	PostgreSQL
R145	Low	**0.11**	1.00	1.00	1.00	1.00
R146	Low	0.00	1.00	1.00	1.00	1.00
R147	Low	**0.16**	1.00	1.00	1.00	1.00
R145	High	**0.22**	1.00	1.00	1.00	1.00
R146	High	**0.91**	1.00	1.00	1.00	1.00
R147	High	0.00	1.00	1.00	1.00	1.00

rate, but it did not yield adequate true positive rates in half of the test cases. The Kolmogorov-Smirnov test yielded true positive rates of 1.0 across all the test cases, but it struggled with false positive rates, failing to meet the threshold for four of the six (firmware revision, project type) pairs.

A potential outlier in the data is firmware revision R146 with a high project task time. All the tests have markedly different false positive

rates for this test case. This could be due to some unknown behavior in the firmware revision or automated conversion process that renders an RTAC AcSELerator project compatible with a new firmware revision.

Figure 7 shows the ROC curves for the three tests without the R146 outlier. The area under the curve values for the Mann-Whitney U test and Kolmogorov-Smirnov test are boosted to 0.94 and 0.97, respectively. However, the permutation test has a modest improvement to just 0.83.

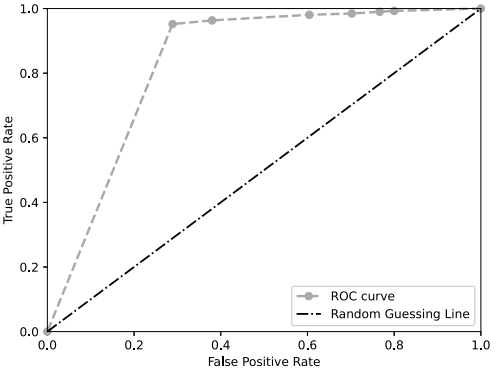
5.2 Improving Detection Rates

The results presented above were collected using all 30,000 samples in each dataset without performing any pre-test data treatments. Future research will explore the effect of sample size on the performance of each algorithm. Additionally, data conditioning methods such as removing outliers or only using specific percentiles of sorted data for detection will be employed. With a compiled dataset and data collection framework in place, further discrimination strategies may be employed, including multiple data features such as tracking memory use in addition to task time. Since multi-dimensionality increases the computational complexity, it should be balanced against the performance gain.

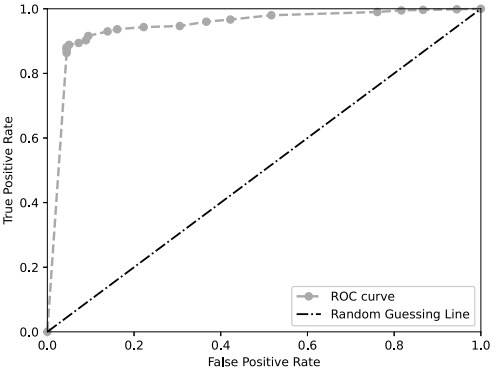
Candidate algorithms that are vetted should be implemented in the CODESYS environment for deployment in a real-time SEL-3505 RTAC. By offloading detection from a central system to the real-time automation controller itself, the burden of network traffic would be removed and the potential for malicious tampering of data in transit would be reduced. This would also enable integration with pre-existing capabilities such as syslog to enable a single point of security auditing for the SEL-3505 RTAC. With multiple detection tests implemented in discrete function blocks, voting schemes could be used and tuned. Additionally, a wider array of attack types could be evaluated to demonstrate the robustness of the anomaly detection solution and provide detailed comparisons with anomaly detection strategies described in the literature.

6. Conclusions

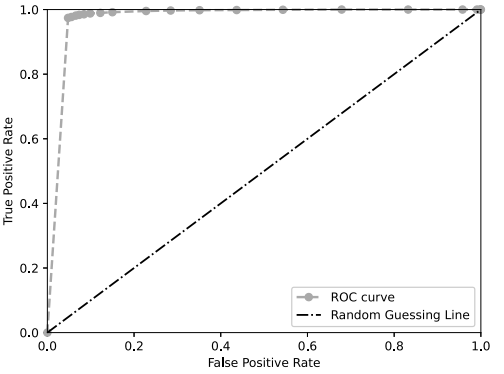
This research has focused on the development of an anomaly detection system for a SEL-3505 RTAC using task time as the data feature. The ability to detect network intrusions using task time data was evaluated using three statistical tests. While the permutation test has been used previously to detect control logic and firmware modifications, the experiments demonstrated that it was unable to discriminate between normal variations in device behavior and the burden created by network intrusions; specifically, the overall false positive rate was unacceptably



(a) Permutation test (AUC = 0.83).



(b) Mann-Whitney U test (AUC = 0.94).



(c) Kolmogorov-Smirnov test (AUC = 0.97).

Figure 7. ROC curves without R146 data.

high. In contrast, the Mann-Whitney U and Kolmogorov-Smirnov tests yielded good anomaly detection results, failing to meet the performance thresholds for only a few experimental treatments. The Kolmogorov-Smirnov test was able to detect network intrusions in all the trials and, when the performance associated with the outlier (firmware revision R146 and high task time) was discounted, the test yielded only 44 false positives out of 450 baseline-to-baseline comparisons, a strong showing for a proof-of-concept test.

Future work will focus on improving anomaly detection performance using data conditioning and exploring the effects of sample size. The data collection framework also provides the flexibility to test future strategies against many data features and project types.

The SEL-3505 RTAC has several security features, but it is not designed to detect the network intrusions tested in this research. ARP spoofing in the absence of a network-based intrusion detection system goes undetected, exposing the industrial control system and the physical system it monitors and controls to disruption or damage. While anomaly detection is not the only alternative, task time is a promising data feature for detecting network intrusions. Moreover, anomaly detection can be implemented in existing installations without adding new devices by deploying the detection functionality in programmable logic controller function blocks. Future research will focus on identifying vulnerabilities and mitigations for the SEL-3505 RTAC as well as other Schweitzer Engineering Laboratories equipment.

The views expressed in this chapter are those of the authors, and do not reflect the official policy or position of the U.S. Air Force, U.S. Department of Defense or U.S. Government. This document has been approved for public release, distribution unlimited (Case #88ABW-2020-3828).

References

[1] T. Alves, R. Das and T. Morris, Embedding encryption and machine learning intrusion prevention systems in programmable logic controllers, *IEEE Embedded Systems Letters*, vol. 10(3), pp. 99–102, 2018.

[2] J. Bernacki and G. Kolaczek, Anomaly detection in network traffic using selected methods of time series analysis, *International Journal of Computer Network and Information Security*, vol. 7(9), pp. 10–18, 2015.

[3] Y. Dodge, *The Concise Encyclopedia of Statistics*, Springer, New York, 2008.

[4] 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, 2016.

[5] S. East, J. Butts, M. Papa and S. Shenoi, A taxonomy of attacks on the DNP3 protocol, in *Critical Infrastructure Protection III*, C. Palmer and S. Shenoi (Eds.), Springer, Berlin Heidelberg, Germany, pp. 67–81, 2009.

[6] 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.

[7] A. Keliris and M. Maniatakos, ICSREF: A framework for automated reverse engineering of industrial control system binaries, *Proceedings of the Twenty-Sixth Annual Network and Distributed System Security Symposium*, 2019.

[8] D. Kite, Leveraging Security-Using the SEL RTAC's Built-In Security Features, SEL White Paper LWP0018-01, Schweitzer Engineering Laboratories, Pullman, Washington, 2016.

[9] H. Mann and D. Whitney, On a test of whether one of two random variables is stochastically larger than the other, *The Annals of Mathematical Statistics*, vol. 18(1), pp. 50–60, 1947.

[10] L. Martin-Liras, M. Prada, J. Fuertes, A. Moran, S. Alonso and M. Dominguez, Comparative analysis of the security of configuration protocols for industrial control devices, *International Journal of Critical Infrastructure Protection*, vol. 19, pp. 4–15, 2017.

[11] R. Mitchell and I. Chen, A survey of intrusion detection techniques for cyber-physical systems, *ACM Computing Surveys*, vol. 46(4), article no. 55, 2014.

[12] 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(1), pp. 13–20, 2008.

[13] M. Niedermaier, J. Malchow, F. Fischer, D. Marzin, D. Merli, V. Roth and A. von Bodisco, You snooze, you lose: Measuring PLC cycle times under attacks, *Proceedings of the Twelfth USENIX Workshop on Offensive Technologies*, 2018.

[14] A. Nochvay, Security Research: CODESYS Runtime, A PLC Control Framework, Version 1.0, Kaspersky, Woburn, Massachusetts (`ics-cert.kaspersky.com/media/KICS-CERT-Codesys-En.pdf`), 2019.

[15] C. Parian, T. Guldimann and S. Bhatia, Fooling the master: Exploiting weaknesses in the Modbus protocol, *Procedia Computer Science*, vol. 171, pp. 2453–2458, 2020.

[16] D. Pliatsios, P. Sarigiannidis, T. Lagkas and A. Sarigiannidis, A survey of SCADA systems: Secure protocols, incidents, threats and tactics, *IEEE Communications Surveys and Tutorials*, vol. 22(3), pp. 1942–1976, 2020.

[17] S. Raschka, MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack, *Journal of Open Source Software*, vol. 3(24), pp. 638–639, 2018.

[18] Schweitzer Engineering Laboratories, SEL ranked top protective relay manufacturer in industry survey, Pullman, Washington (`www.selinc.com/company/news/126347`), June 12, 2019.

[19] Schweitzer Engineering Laboratories, Customer Highlights, Pullman, Washington (`www.selinc.com/solutions/success-stories`), 2020.

[20] Schweitzer Engineering Laboratories, SEL-3505 SEL-3505-3 Real-Time Automation Controller Instruction Manual, Pullman, Washington, 2020.

[21] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed and V. Roussev, Denial of engineering operations attacks on industrial control systems, *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pp. 319–329, 2018.

[22] 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, 2017.

[23] C. Vargas Martinez and B. Vogel-Heuser, A host intrusion detection system architecture for embedded industrial devices, *Journal of the Franklin Institute*, vol. 358(1), pp. 210–236, 2021.

[24] P. Virtanen, R. Gommers, T. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. van der Walt, M. Brett, J. Wilson, K. Millman, N. Mayorov, A. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, I. Polat, Y. Feng, E. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. Quintero, C. Harris, A. Archibald, A. Ribeiro, F. Pedregosa, P. van Mulbregt and SciPy 1.0 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in Python, *Nature Methods*, vol. 17, pp. 261–272, 2020.

[25] R. Wilcox, *Applying Contemporary Statistical Techniques*, Academic Press, Burlington, Massachusetts, 2003.

[26] D. Zhu and Y. Cui, Understanding the random guessing line in a ROC curve, *Proceedings of the Second International Conference on Image, Vision and Computing*, pp. 1156–1159, 2017.