University of Cyprus

Department of Electrical and Computer Engineering

Packet Analysis & Anomaly Detection with Machine Learning

Supervisor

Prof. Angelos Marnerides

Undergraduate Student

Lazaros Lazarou

Academic Year 2024-2025

1 Introduction

Programmable Logic Controllers (PLCs) are primary targets of cyberattacks against industrial environments due to their role in automating necessary processes. These attacks can seriously disrupt operations by halting production lines, damaging equipment and causing safety hazards to employees and the environment. A mechanism that can identify possible intrusions is crucial to ensure the secure operation of these systems. This documentation outlines the development of a network traffic anomaly detection system using machine learning techniques. The system is designed to monitor the network traffic and analyze packets exchanged between devices, including PLCs, in order to detect attackers that exploit vulnerabilities in these systems. Our initial goal was to apply both packet-based and flow-based analysis methods to detect deviations from normal network behaviour, ensuring industrial operations' safety and reliability. However, testing this approach revealed limitations, it was found to be incapable of detecting several intrusions. This led us to look into alternative methods, for instance, analyzing delayed responses and inspecting packet payloads, to enhance detection accuracy.

1.1 Devices Setup

In the first test setup, there was only one laptop with Windows running both OpenPLC runtime, in order to simulate a working PLC, and ScadaBR for HMI visualization. Indeed, this setup became inefficient, as both applications were running on the same device with no need for actual network communication. To overcome this limitation, we have reorganized the system by using two different devices where one is assigned to run OpenPLC and the other on ScadaBR. In this distributed configuration, the ScadaBR sends requests for coil values and holding register values, and OpenPLC responds through the network, therefore, generating the necessary network traffic. This setup more closely represents real-world conditions and allows us to log the network traffic between the PLC and HMI in a way that represents a fairly realistic scenario within the industrial world.

2 Initial Methodology: Packet and Flow-Based Analysis

Packet analysis plays an important role in network security, especially in environments where real-time communication between critical systems, such as PLCs, must be defended. PLCs tend to continuously exchange data with other devices that manage operations, including supervisory systems such as Human Machine Interfaces (HMIs). This constant communication makes it difficult to estimate which packets are sent from authorized devices. Analyzing the contents of each packet separately raises significantly the complexity of the detection process. Therefore, this work focuses more on identifying patterns in communication such as unexpected spikes in traffic, abnormal packet sizes and other anomalies. To achieve more accurate results, we separate the analysis into both packet-based and flow-based packet analysis. These techniques complement one another and are employed when searching for anomalies in network traffic, with each offering distinct advantages. Although both methods share the same goal, the various features of each can detect different types of attacks. Since our aim is to create a detection system tailored specifically for industrial environments, and to recognize adversaries that could exhibit enough stealth to avoid traditional network detection techniques, we narrow our packet captures to only the communication between the PLC and the SCADA HMI. By focusing on this critical data exchange, we can carefully inspect any changes in their communication patterns that occur as an attack is taking place. This focused approach enables the detection of even slight anomalies, as even the slightest deviation in the communication between these critical devices could mean potential intrusions or disruptions.

2.1 Flow-Based Analysis

This method focuses on aggregated network flows, which represent a series of packets exchanged during a communication session between two endpoints. In the experiment, packets are captured in time periods from one to five minutes and saved in files in pcap format. The captured packets are divided into groups of packets that have common source and destination IP addresses and ports. To start the analysis each of these sets of packets, also commonly referred to as packet flows, represents a session between two devices. Once these flows are separated, we compute the following key features for each flow:

- Flow duration: Total time span for the flow, which is estimated based on the time difference between the first and last packets of the session.
- Average inter-arrival time: Elapsed time between consecutive packet arrivals. It gives us valuable information about the communication rate.
- Total packet size: The sum of all packet sizes in bytes within the flow. This helps identify the volumes of data trends.
- Average payload size: This is the calculated average size of the payload across all packets in the flow. This provides details about how much data is being transmitted.

After determining these metrics for each flow, general statistics are calculated across all flows. These include the mean, standard deviation, minimum, maximum, and Shannon entropy for each feature. The entropy metric can be useful in assessing the level of unpredictability or randomness inside the network flows, which may be linked to anomalies.

After reducing the scope of packet capture to only the two important devices, flow-based analysis became ineffective. Since there was only one communication flow between these two devices, flow-based analysis could not provide meaningful information, as it depends on multiple flows to start identifying anomalies or patterns in the different interactions. This limitation made the flow-based metrics redundant in this scenario. Consequently, we decided to abandon the flow-based analysis approach and, alternatively, promote the packet-based analysis method incorporating particular flow-based measurements to maintain an optimal amount of detection accuracy.

2.2 Packet-Based Analysis

Packet-based analysis consists of examining each packet passing through the network. Every packet contains information such as source and destination IP addresses, protocols, payload data and more header fields, but once again we focus on the patterns of the traffic like abnormal packet frequency. This approach provides a different set of features, by focusing on individual packets.

For the same packet capture samples as before, we collect time-based features as listed below:

- Packets sent per given period: The count of packets transmitted within a given time interval. This feature helps by detecting sudden spikes or drops in packet frequency.
- Bytes per given period: Total packet size in bytes sent during a period. By monitoring the amount of data transmitted, we can identify data exfiltration.
- Average packet size per given period: By combining the feature above, we calculate the average packet size during the time interval. This helps us detect if some packets contain a higher volume of data than expected.

For this experiment the time period is set to 1 second, meaning that we count the packets sent per second etc. After removing flow-based analysis, the following features were adapted and added to packet-based analysis to maintain the depth of detection:

- Inter-Arrival packet time: Time between consecutive packet arrivals
- Payload Size: The calculated size of the payload of all packets in the captured file.

We thus compute the average, standard deviation, minimum, maximum, and Shannon entropy for each feature. This way we have a more clear understanding of traffic patterns over time, allowing us to spot irregularities in network behavior.

Standard deviation is a statistical measure that shows the amount of variability or dispersion of a set of data points. Simply put, it describes how spread out the values are from the mean value. A low standard deviation means that, on average, most of the data points tend to be near the mean value, on the other hand, a high standard deviation implies that data points are spread over a wide range of values. In the case of examining the inter-arrival times of packets, a low standard deviation would indicate consistency in the timing between packets and thus represent normal communication. The standard deviation σ of a data set may be computed by:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{1}$$

- N is the number of data points (sample size).
- x_i is each individual data point in the dataset
- μ is the mean (average) of the dataset.
- (x_i - μ) represents the difference between each data point and the mean, indicating how far each value is from the mean.

The Shannon Entropy metric measures the unpredictability or randomness of packet data, providing information about network behavior variability. To calculate the Shannon Entropy $H(X)$ of a set of values $X = \{x_1, x_2, \ldots, x_N\}$, we use:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{2}$$

where $N = |X|$ represents the number of unique values in X and p_i is the probability $P[X = x_i]$, measuring the expected amount of information conveyed by the data set. For anomaly detection purposes, further improvements are achieved through normalization of values, resulting in:

$$H'(X) = -\frac{1}{\log_2 N}\sum_{i=1}^{N} p_i \log_2(p_i) \tag{3}$$

This normalized entropy $H'(X)$ scales the entropy value between 0 and 1, facilitating comparative analysis across datasets.

3 Analysis Results

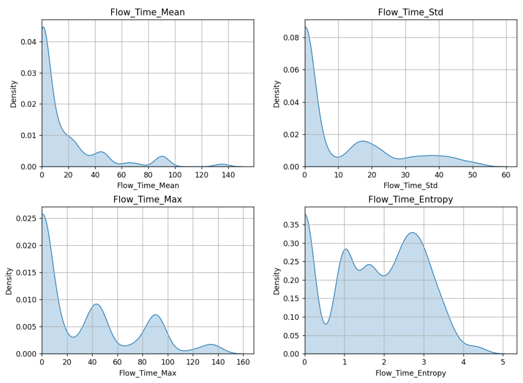
The metrics and features are written to a CSV file where each row represents the result of a capture sample, and each column corresponds to the feature. These features include the metrics that were previously mentioned. Each capture sample reflects packets collected over a fixed duration, typically some minutes.

To visualize the data collected we plot the Probability Distribution Function (PDF) and Cumulative Distribution Function (CDF) for all the features. The PDF gives the probability of occurrences of different values for a certain feature, while the CDF gives the total probability of a feature being less than or equal to a particular value. Such visualization enables us to look at normal behavior in network traffic and recognise anomalies when patterns of behavior deviate from the expected ones.

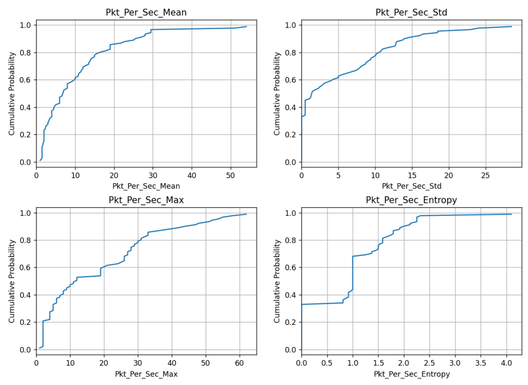
After running some tests, we let the data capture and analysis run for much longer periods of time, both in the normal state and under-attack states. Contrary to our initial hypothesis, the results showed no difference between the patterns of data in these two states, pointing out our approach failed.

In fact, after investigating more deeply, we realized that, although the values of the PLC's coils and holding registers do change much more during an attack, this change is not reflected in the communication data. The reason is that SCADA systems poll these values at fixed intervals, so no matter how often the values change, the communication pattern is always the same because the request-response cycle is driven by the polling rate of the SCADA system. This fixed sampling interval obscured the activity variations under attack, such that patterns for the normal and compromised states were nearly identical.

NOTE: the figure results will be updated with more figures to compare between the normal and under-attack differences proving that there is little to no difference



(a) PDF of Flow Time



(b) CDF of Packets Sent Per Second

Figure 1: Examples for CDF and PDF

4 Delayed Response Analysis

With an alternative approach, we suggested a new hypothesis: in the case of an attack, as the PLC values change at a much higher frequency, then the additional load of the PLC processing would result in delayed responses. Simply put, we are expecting that an attack causing high frequency of changes is able to slow down the PLC processor enough so that its responses to requests from the SCADA system are measurably delayed. To evaluate this, we implemented a basic detection mechanism that measured the time difference between each request from the SCADA system and its corresponding response from the PLC. We suggested this approach as a straightforward and potentially effective complementary detection technique.

Again, after extensive testing, we found this to be an ineffective approach. The delays in responses were not changed significantly, even when under attack. It appears that the attack was not introducing enough stress on the PLC, to affect the response timing and make this detection mechanism ineffective. This result means that their response delays are not a reliable signature of anomalous behavior in this setup as the response time from the PLC remained the same irrespective of the frequency at which its internal values were modified.

5 Payload Analysis

Considering the ineffectiveness of all prior methods, we decided to try a different approach. Instead of considering traffic patterns, we turned our attention to packet payloads and their variation across packets. Specifically, we initiated an analysis of the response packets, since they include the most relevant information with regard to the system state.

From our previous analysis, it was concluded that the communication between the two devices, Scada and PLC has a standardized structure of three main types of packets: request, response, and acknowledgment. The communication initiates with the Scada requesting the coil values from the PLC, then the PLC sends the values in a response packet, which is followed by an acknowledgment packet from the Scada, which verifies that the transmitted data was successfully received. Upon completion of this exchange, the SCADA system requests once again the values of the holding registers, and the cycle of communication starts all over again in the same way. This cyclical operation continues at regular intervals defined by the update rate predetermined in the SCADA system.

Now, to detect if the system is under attack, we compute the three different metrics of the encoded payload string in response packets. These metrcis are:

- Positional Entropy: This measure estimates the predictability of changes at specific positions in the string, indicating patterns or randomness in character changes.
- Change/No-Change Entropy: This measure is based on the probability of characters either changing or remaining unchanged between different strings, which gives insight into the stability of the payload over time.
- Hamming Distance: This measurement counts the total number of mismatches between two strings by measuring the number of non-matching characters, which results in an intuitive measure of difference.

The assumption holds that when the PLC is under attack, the attacker commences manipulating the system by spamming wrong values to all Modbus addresses used by the PLC. This assumption is based on the idea that if the attacker only altered a limited number of operations or did so infrequently, then the system would likely recover. The PLC program is normally designed in such a way that it corrects erroneous values and can also contain failure detection mechanisms to identify abnormalities and restore normal operations. However, through the continuous flooding of the system with false values, the attacker prevents the system from returning to its correct state, as a result creating a continuous disruption.

This study tries to find such behavior through an analysis of the entropy based on the payload's string. High values of entropy might indicate an increase in the randomness or irregularity of the payload and could be a sign of an intrusion. This approach exploits the natural predictability and structured nature of normal Modbus communication to identify anomalies due to attacks.

5.1 Analysis Implementation

To conduct this analysis, we capture network packets sent/received by the SCADA and PLC, same way with the previous analysis, and store them in pcap files. Each pcap file contains packets captured in a time duration of 5 seconds. To identify whether a packet is a response for coils or registers, we check the payload size in bytes. For the PLC program used in this experiment:

- Coil response packets have a payload size of 14 bytes.
- Register response packets have a total payload size of 44 bytes, split into two packets, one with 11 bytes and the other with 33 bytes.
- Request packets have a payload size of 12 bytes.
- Acknowledgment packets have a payload size of 0 bytes.

To calculate the entropy of the payload strings for the coil and the register, we compare the payload of each response packet with the payload of the previous response packet of the same type. This allows us to quantify how much the payload changes between consecutive responses. By focusing on these changes, our goal is to detect anomalies that may indicate the presence of an attack.

The comparison of payload strings and identification of their differences is done by taking advantage of the fact that the encoded payload string values have always the same lengths as the payload sizes are fixed. The process involves comparing each character of the string with the character at the same position in the other string. If the characters differ, the mismatch and its position are noted. Here is an example of two payload strings and their comparison:

String 1: 03700000000b0103082269009205dc0dac
String 2: 03720000000b0103080596001717700 7d0
Differences: 14

After analyzing all the strings in the file, we process the mismatches and their positions to compute the metrics. In order to calculate positional entropy, we use the probability of a mismatch at every position in the string and the total number of mismatches for all positions. This gives us an insight into how predictable or random the mismatches are across the string's positions

$$H_{\text{positional}} = -\frac{1}{\log_2(L)} \sum_{i=1}^{L} \left(\frac{m_i}{M} \log_2 \left(\frac{m_i}{M} \right) \right) \tag{4}$$

Where:

- $H_{\text{positional}}$: Positional entropy.
- L: Length of the string (number of positions).
- m_i: Count of mismatches at position i
- M: Total number of mismatches across all positions

The Change vs. No Change entropy quantifies the uncertainty in the distribution of changed and unchanged characters in the string comparison process. This metric is calculated based on the proportion of characters that change versus those that remain unchanged between two payload strings. Since the payload sizes are fixed, the probabilities of "change" and "no change" are complementary: $P_{\text{changed}} = 1 - P_{\text{unchanged}}$ The entropy is then calculated using the standard entropy formula, summing over these probabilities: The entropy is calculated using the standard entropy formula:

$$H_{\text{change}} = -P_{\text{changed}} \log_2(P_{\text{changed}}) - P_{\text{unchanged}} \log_2(P_{\text{unchanged}}) \tag{5}$$

Where:

- H_{change}: Change vs. No Change entropy.
- $P_{\text{changed}} = \frac{\text{changed\_count}}{\text{total\_chars}}$: Probability of a character being changed.
- $P_{\text{unchanged}} = \frac{\text{unchanged\_count}}{\text{total\_chars}}$: Probability of a character being unchanged.
- total_chars: Total number of characters in the payload string.

The *Hamming Distance* measures the number of differing characters between two strings of equal length. It provides a straightforward metric for assessing how much the payload string in a response packet differs from the previous one. The average Hamming distance is normalized to account for the string length, ensuring consistency across varying payload sizes.

The Hamming distance for two strings S_1 and S_2 of length L is defined as:

$$d_H(S_1, S_2) = \sum_{i=1}^{L} \delta(S_1[i], S_2[i]) \tag{6}$$

Where:

- $d_H(S_1, S_2)$: Hamming distance between strings S_1 and S_2.
- L: Length of the strings.
- $\delta(S_1[i], S_2[i])$: Indicator function, defined as:

$$\delta(S_1[i], S_2[i]) = \begin{cases} 1 & \text{if } S_1[i] \neq S_2[i] \\ 0 & \text{if } S_1[i] = S_2[i] \end{cases}$$

To calculate the *normalized average Hamming distance* across all pairs of strings in the dataset, the formula is:

$$\text{Hamming\_Dist\_Avg} = \frac{\frac{1}{N}\sum_{k=1}^{N} d_H(S_{1,k}, S_{2,k})}{L} \tag{7}$$

Where:

- Hamming_Dist_Avg: Normalized average Hamming distance.
- N: Number of string pairs compared.
- $d_H(S_{1,k}, S_{2,k})$: Hamming distance for the k^{th} pair of strings.
- L: Length of each string.

In the implementation, the Hamming distance is computed for all string pairs, and the normalized average is derived using the formula above.

The calculated metrics for each file are stored in a table within a HDF file. HDF (Hierarchical Data Formats) files are used to store large amounts of data in the form of multidimensional arrays.

To gather sufficient data for training and testing a machine learning model, we collected data of coil and register payloads over a period of two days. These values were calculated by analyzing the differences between strings in the packet payloads. The packet captures, spanning durations of 5 seconds each, were saved in multiple pcap files. These files were subsequently processed to extract and compute entropy metrics for each communication session. The resulting dataset provides a robust representation of the communication patterns between the SCADA and PLC devices for both normal and under-attack operating conditions.

6 Training and Testing the Detection System

The main objective of the machine learning model is to consistently be able to determine whether the system is under attack or operating normally. Previous approaches, which were dependent on traffic pattern analysis based on packet-based metrics, completely failed to detect intrusions in communications between the SCADA system and the PLC. As a result, the new approach became focused on using payload-based entropy metrics, which work to capture variations in the payload of packets, especially in the response packets.

By training a machine learning model on these entropy values, it aims to recognize anomalous characteristics of malicious activity such as spamming incorrect values or disrupting normal operations. The model is expected to generalize well to unseen data, allowing for real-time intrusion detection.

6.1 Overview of the Data for Machine Learning

The dataset for training and testing the machine learning model contains entropy values taken from packet payloads used in SCADA-PLC communication. The data are collected over a week, which gives a complete picture of normal operations and attack situations.

Each data sample represents a particular time period, normally ranging from 30 to 90 seconds of network packets collected. Entropy values were calculated for every capture session using the coil and register payloads in the response packets.

The dataset is structured in a tabular format, where rows represent captures sessions and columns present calculated features. Additionally, labels that indicate whether the session represents an attack state or a normal state are included, enabling supervised learning. This dataset serves as the basis for both training and testing the model, allowing it to effectively differentiate between normal and abnormal system behavior.

6.2 Initial Approach: One-Class SVM for Anomaly Detection

In the effort to develop an anomaly detection algorithm, the One-Class Support Vector Machine (One-Class SVM) was chosen as the initial approach. One-Class SVM is an unsupervised machine learning method, which learns from normal data and identifies any outliers that deviate from the norm as anomalies. Its capabilities are especially suitable for our particular application in anomaly detection.

In this approach, the model was developed based only on the normal communication data, representing typical interaction between the PLC and SCADA systems. It was expected that the model would learn the features inherent in such typical communication in a way that any important deviation would be identified as anomaly. The one-class SVM is based on constructing a decision boundary that encapsulates the normal data points while maximizing the margin between these usual examples and the rest of the feature space at the same time. In this way, outliers such as anomalous or attack data are usually outside the decision boundary, making them possible to detect.

The One-Class SVM contains some important hyperparameters that influence its performance, including 'gamma' and 'nu'. The hyperparameter gamma is crucial for the formation of the decision boundary and hence influences the data fitting process of the algorithm. A high value of gamma creates a complex and flexible boundary while a low value of gamma creates a simpler and wider boundary. On the other hand, the nu parameter sets the upper bound on the fraction of margin errors and support vectors. It acts like a regularization component that helps to control the balance between false positives and false negatives. One can tune the precision and recall of the model by changing the nu parameter in order to reach the optimal trade-off.