Detection and Prevention of Cyberattack

Leo Oh, Riley Wilkinson, Seong Won Park

April 4, 2025

1 Executive Summary

Given the broad objective of developing models to assist various entities in defending against cyber threats, we chose to narrow our project's focus to Advanced Persistent Threats (APT). APTs are a form of cyberattack, characterized by their prolonged duration, stealthy nature, and the involvement of well-funded adversaries. Unlike more opportunistic attacks, APTs often aim to infiltrate a network undetected, maintain long-term access, and gather sensitive information over time. Recognizing the complexity and real-world significance of these threats, we aimed to build models sophisticated enough to detect anomalies in network traffic and accurately identify the point of attack, enabling effective response and mitigation.

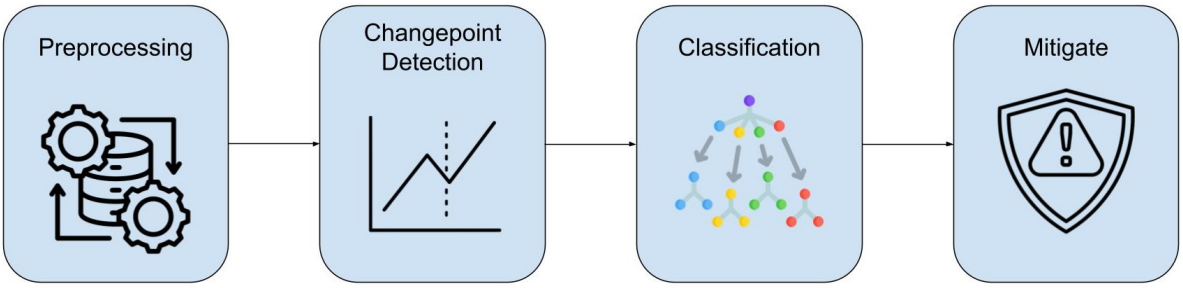

Figure 1: Detection Pipeline

Utilizing the CIC APT IIoT 2024 dataset, we modeled network traffic log data as a graph, where nodes represented IP addresses and edges captured the interactions between them. Our approach focused on monitoring temporal changes in edge attributes to identify features that were indicative of anomalous behavior, with the ultimate goal of isolating potentially malicious IP addresses. Since the raw dataset contained all interactions across the network, we preprocessed the data to examine traffic on a per-IP basis and aggregated it accordingly to isolate individual network behavior.

Once relevant features were identified, specifically those exhibiting a shift in distribution over time, we applied changepoint detection techniques to pinpoint the exact moments at which these distributional changes occurred. To better simulate real-world scenarios, we emphasized the development of an online changepoint detection model capable of identifying anomalies in real time as data streams in. Because the feature distributions were neither normal nor stationary, we prioritized the use of non-parametric methods, which are discussed in detail in the methodology section.

After detecting potential changepoints, our next objective was to identify nodes that might pose a threat and to classify the corresponding stages of attack. Thus, we isolated each node's activities and compared the individual IP addresses' changepoint to the aggregate's. Further, we utilized a Random Forest model fitted with the network traffic data as our features and different stages of attacks as our response variable. The main idea was to run the classification model trained on data near the changepoint, so we could identify the stages of attack and respond accordingly.

We hope our work provides actionable insights to decision makers where they can incorporate our detection pipeline into their cybersecurity measures. With an interpretable graphical concept, our framework of APT identification and classification contributes to the dynamic and adaptive defenses against evolving cyber threats.

2 Datasets and Data Exploration

We utilized the CIC APT IIoT 2024 Dataset, specifically focusing on the Network Traffic data. The Network Traffic data consists of two parts: Phase 1 and Phase 2. We utilized both phases, given prior knowledge that the Phase 1 dataset contained network traffic captured by NS3 for all devices in the experiment conducted over four days. This phase simulated normal system operations to establish a baseline behavior for the testbed components, including VMs, sensors, and Raspberry Pis.

The Phase 2 dataset, captured over three days, contains the same type of network traffic data but during a simulated APT attack. The attack was executed using APT29 tactics through Kali VM1 with MITRE Caldera. During the simulated attack, Phase 2 followed APT's "low and slow" approach, where attack steps were executed at random intervals of 45 to 75 minutes to mimic the stealth and persistence typical of APT attacks. This strategy was employed to realistically reflect APT attack characteristics in the Phase 2 data.

The 2024 Network Traffic Data also included the TS (Time Stamp) feature, recorded as a Unix timestamp, which represents the number of seconds elapsed since January 1, 1970 (UTC). Since the timestamp was recorded in fractions of a second, we aggregated the entire 2024 Network Traffic Data at the minute levels. During this process, we averaged all numeric columns. Additionally, we marked the packets aggregated over seconds or minutes as an attack if at least one packet within that time frame was classified as an attack.

Since one of our main goals was to detect anomalies in network traffic and successfully identify the attack phase, we plotted the density of packet counts per second for both Phase 1 and Phase 2. We also plotted the density of packet counts per second for each scenario, comparing cases where an attack occurred versus when it did not.

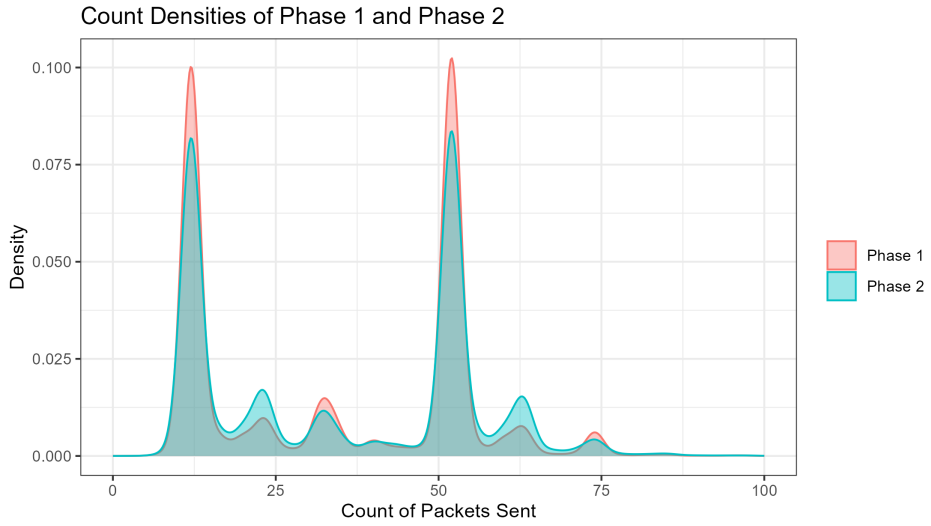

Figure 2: Counts of packets being sent per seconds for each of phase 1 and phase 2

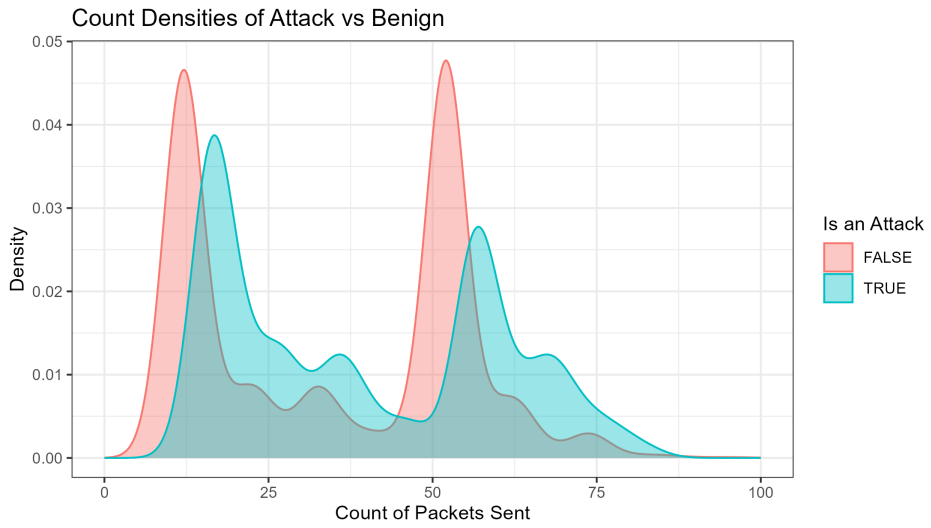

Figure 3: Counts of packets being sent per seconds under Attack or Benign

As we can see in Figure 3, there are some discrepancies in density between the two phases and the presence of an attack. We also plotted an edge graph to identify which edges between IP addresses were marked as part of the attack and which nodes (IP addresses) were involved. Specifically, we plotted two edge graphs—one for when an APT attack occurred and one for when the traffic was benign. We marked the packets sent between IP addresses during the attack with red arrows, allowing us to identify the two IP addresses, 172.16.65.128 and 172.16.63.128, that were used during the attack.

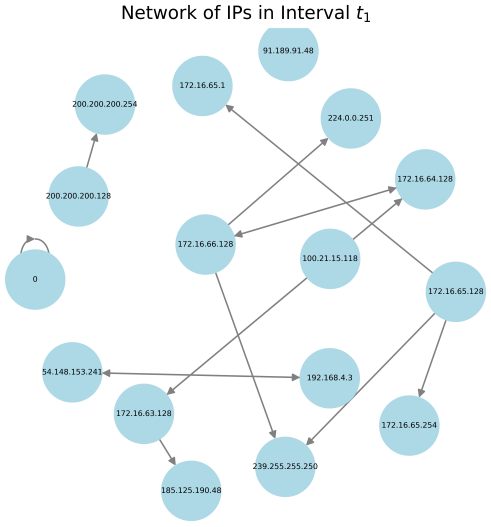

Figure 4: Benign

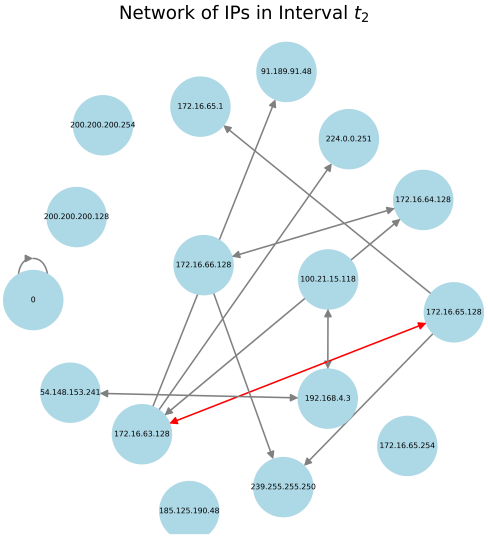

Figure 5: Attack

In addition to aggregating the CIC APT IIoT 2024 Dataset by timestamp in minutes, we also aggregated the data by IP address. We grouped each row depending on whether the IP address was included in the "from" or "to" columns. After aggregating the data by IP address, we conducted separate changepoint analyses to identify which IP addresses exhibited significant changes.

During our exploratory data analysis, we conducted a LASSO regression using $I(ts > ts_{firstattack}$ as the target in order to determine variable importance. Using the top 10 most important variables identified through the LASSO, we determined that the flow idle time was the most important for determining a difference between pre and post attack states.

3 Online Changepoint

As discussed in the previous section, we decided to utilize online changepoint detection to better reflect real-world scenarios in which live network traffic logs are accessible. We considered changepoint detection methods to be a robust approach for identifying unforeseen anomalies, which traditional classification models may fail to detect. Consequently, our task was to identify the online changepoint method most appropriate for our dataset.

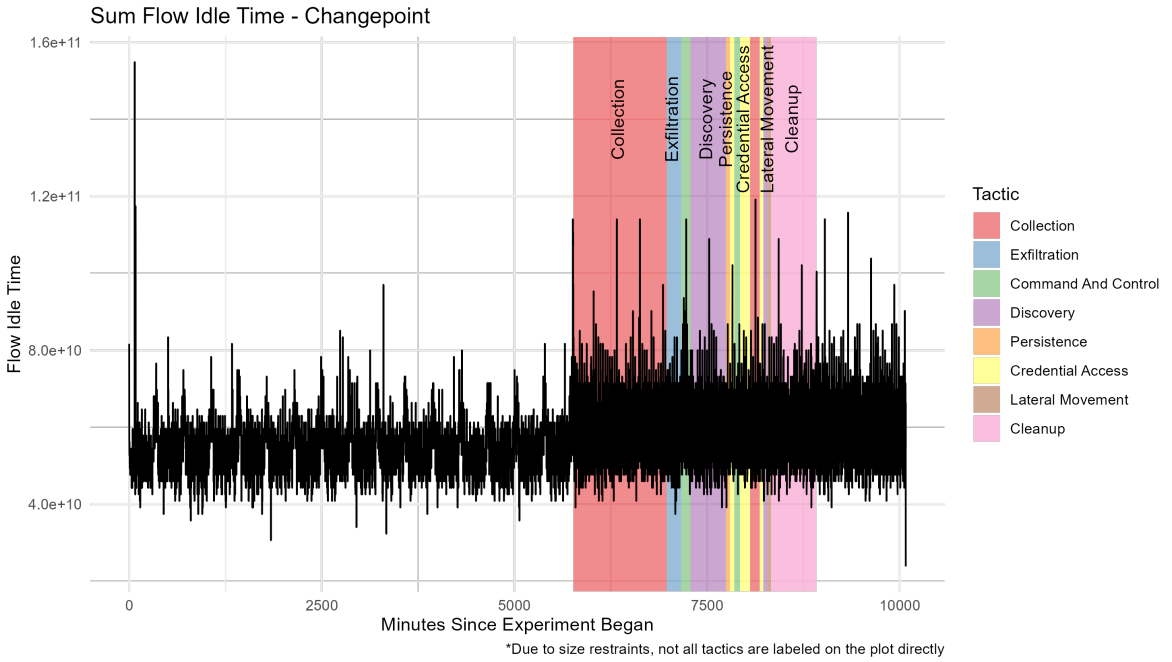

Figure 6: Main variable used for anomoly detection

The CUSUM (Cumulative Sum) algorithm, introduced by Page in 1954 [Pag54], is one of the most widely utilized alogrithm for online changepoint. It detects changes by calculating the cumulative sum of deviations from a reference value, typically under the assumption that the distributions before and after the changepoint are known. This requirement for prior knowledge of both pre-change and post-change distributions posed a significant limitation for our setting, where the nature of anomalies is unpredictable. Therefore, most methods derived from CUSUM were not suitable for our application.

We also examined Bayesian Online Changepoint Detection (BOCPD), proposed by Adams and MacKay in 2007 [AM07], which treats changepoints probabilistically by introducing a hazard function that models the probability that a changepoint occurs at a given time. However, the original BOCPD framework assumes a constant hazard rate, which may not align well with the nonstationary behavior of cyberattacks, where the likelihood of changes can vary over time.

This led us to consider non-parametric methods, which estimate distributional changes without relying on strong parametric assumptions. These methods often utilize empirical cumulative distribution functions (CDFs) to detect distributional shifts. For instance, the Non-Parametric UNbounded Changepoint (NUNC) [AREF23] detection method focuses on identifying changepoints within a sliding window using rank-based or empirical measures. However, its sensitivity to hyperparameters, such as the window size, was an issue especially when we have small prior knowledge about the data.

3.1 NP-FOCuS

Romano developed NP-FOCus (NPF) [REF24] with the only assumption that the observations were iid. However, within the literature, the algorithm proved to be effective even with the violation of the iid assumption. Contrary to NUNC, NPF relies on pruning sequences of data stream based on its

likeihood generated by the binomial distribution from the empirical CDFs.

$$n\hat{F}_{1:n}(p) \sim \text{Binom}(n, \theta)$$

In this equation, θ is an unknown rate of the observations, n is the total length of the data sequence, and p is our fixed parameter of quantiles. We then can check the impact of proposed θ on the likelihood given that we have chosen the most likely sequence.

$$\max_{0 \leq \tau < n} [-L(y_{\tau+1:n}; \theta_0, p) + L(y_{\tau+1:n}; \theta_1, p)]$$

τ is the point of change and θ_0, θ_1 are the original and proposed rates of the distribution. Without getting into technical details of NPF's algorithm, the main idea is to represent the Likelihood ratio test statistic to be functional with respect to θ and find the value that maximizes the functional representation of the likelihood ratio test statistic. In short, we are finding the maximum of the maximum.

Given that our functional test statistics are quadratics, we have n possible functions. Thus, when n gets large, we need a way to prune functions that are sub-optimal regardless of the value of θ to obtain a more optimal subset. Once we have obtained the pruned subset, we then find the maximum point, $\mathcal{Q}_n$, within the subset.

With our predetermined set of quantiles, $p \in \{p_1, ..., p_M\}$s, we detect a changepoint by either comparing

$$\sum_{m=1}^{M} \mathcal{Q}_n^m \quad \text{or} \quad \max_{m \in \{1,...,M\}} \mathcal{Q}_n^m$$

to a threshold. If the selected statistic is higher than the threshold, we declare a changepoint. This process is repeated as new data points are added, and various computational strategies have been proposed to improve the algorithm's efficiency, as detailed in Romano's paper. For this project, we used the sum-based statistic to detect changepoints.

4 Random Forest

The Random Forest model, developed by Breiman [Bre01] is one of the most versatile and well known machine learning algorithms. In many instances, random forests can even outperform other machine learning algorithms like boosted trees and deep neural networks [Roß18]. In addition, random forests can capture linear and non-linear relationships and are extremely robust.

The main building block of a random forest is a decision tree. Decision trees are easy to interpret and can handle multiple different data types; however, they almost always overfit the data [Roß18]. The solution is to use an ensemble of decision trees. In this case, different trees are created by selecting a bootstrapped sample of the data to train each tree and then further randomized by randomly selecting a subset of the predictors at each node. This randomness is what makes the ensemble possible [Die00]. The collection of weak learners that don't have access to all of the data creates a single strong learner that gives accurate predictions.

For this analysis, we chose to use a random forest because of its versatility and consistent high performance across different types of data.

5 Modeling and Analysis

5.1 Aggregated Changepoint

The most important variable that we found was the flow idle time, and Figure 6 clearly demonstrates why. In Figure 6, we take the sum of the flow idle time for each minute in the experiment. This provides a constant distribution that changes once the attacks begin in phase 2. The change is clear and our goal was to build a model that could detect the change.

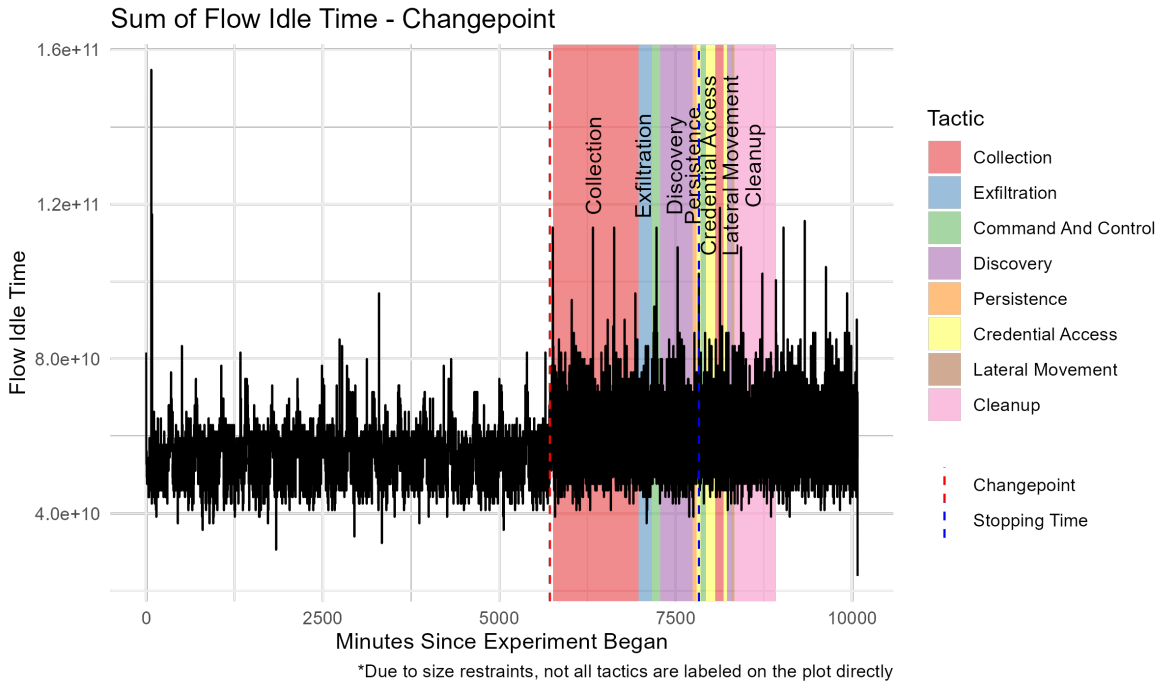

Figure 7: dashed lines represent the changepoint detected and the time it was detected

Our final model was able to detect a changepoint at time 5718 represented by the dashed red line in Figure 7. The "stopping time" is the moment that the model was able to detect the changepoint and it is referenced in Figure 7 as a blue dashed line. Our model "stopping time" was time 7829 (minutes since the experiment began).

Ideally, the changepoint should be detected as soon as possible. We decided to use an online changepoint method in order to simulate a real life scenario. The implication is that every minute that the changepoint is not detected is another minute that the malicious user has access to the system. We were able to detect the change before a lot of the damage was done; however, the sooner that we can detect the change, the better.

5.2 IP Address Changepoint

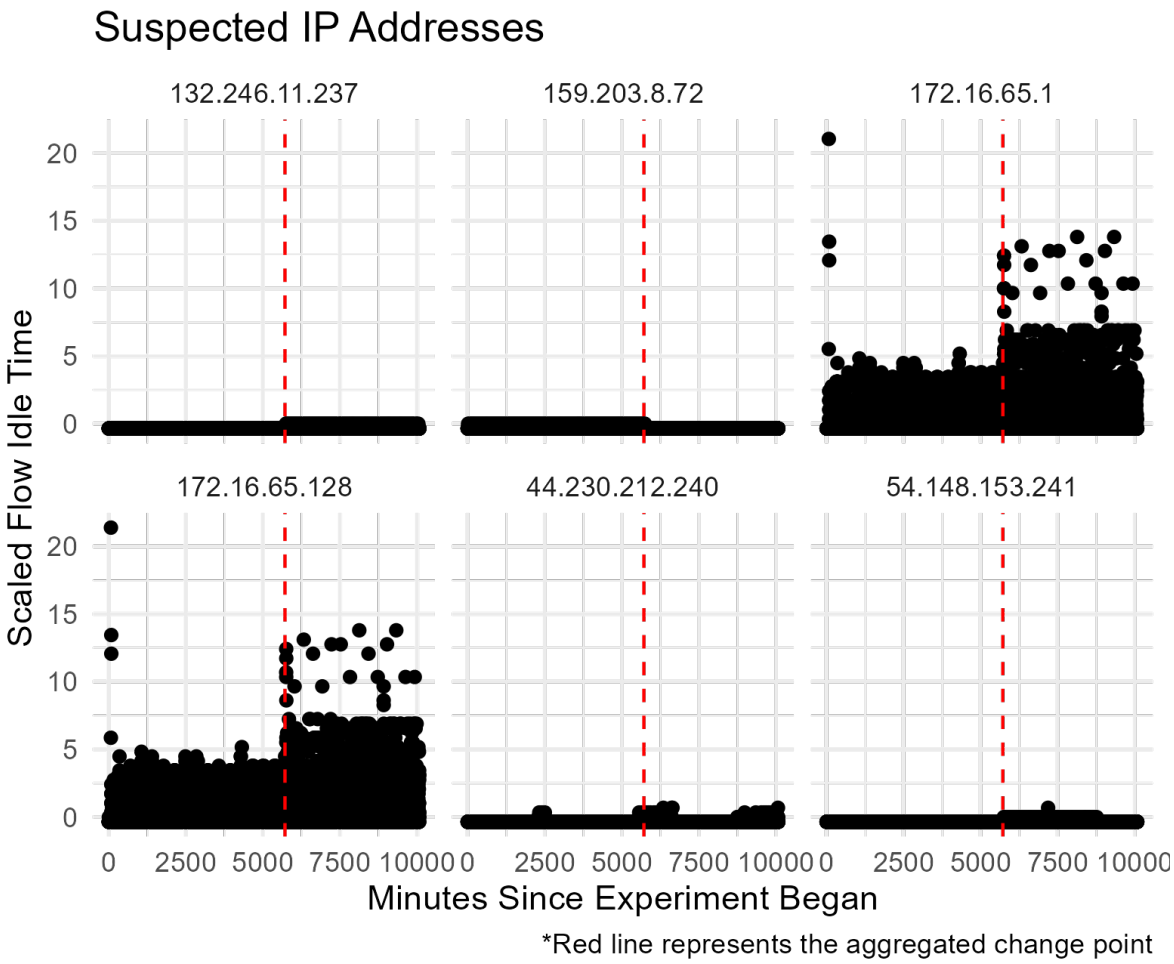

Figure 8: The suspected IP Addresses

The next step in the analysis is to determine the source of the malicious activity. Our goal is to determine what caused the changepoint to occur. In order to achieve this, we subsetted the data by whether or not the interaction included a specific IP address, and then we used the same online change point detection algorithm (NPF) to determine which IP Addresses contained the source of the variation.

The detection algorithm would create a flag for each changepoint detected in each of the IP Address subsets. The moment that a changepoint is detected in the aggregated data, the IP addresses that had the closest detected changepoints were observed and plotted to determine which of them was the cause of the aggregated changepoint. As can be seen from Figure 6, the IP Addresses 172.16.65.1 and 172.16.65.128 are highly likely to be malicious and have been identified as the cause of the aggregated changepoint.

5.3 Random Forest Predictions

Once we have identified that there is unusual activity happening, we would like some way to know what the attacker did so that we can reverse the actions and expunge the attacker from the network.

To accomplish this, we constructed a random forest to predict the attack type (known as sublabel in the dataset). These are the same types that are referenced in Figure 6 and Figure 7. Once the malicious interactions from the network log data are identified, then it is possible to counteract the

effects.

To fit the random forest model, we first processed the data. Since the model is intended to be used near the attack points detected through changepoint analysis, we sampled 2,000 data points—1,000 before and 1,000 after each attack point. After aggregating these 2000× (number of attacks) data points, we removed any duplicate entries. Using this processed data, we trained the random forest model to predict the sublabel using all available features. We used 10 fold cross validation to pick optimal number of features to be randomly selected at each node. After fitting the model, we plotted and ranked the features by variable importance. We then selected the 25 most important features and refitted the random forest model using the same dataset, as shown in Figure 9. This refined model yielded improved performance, as presented in Table 1.

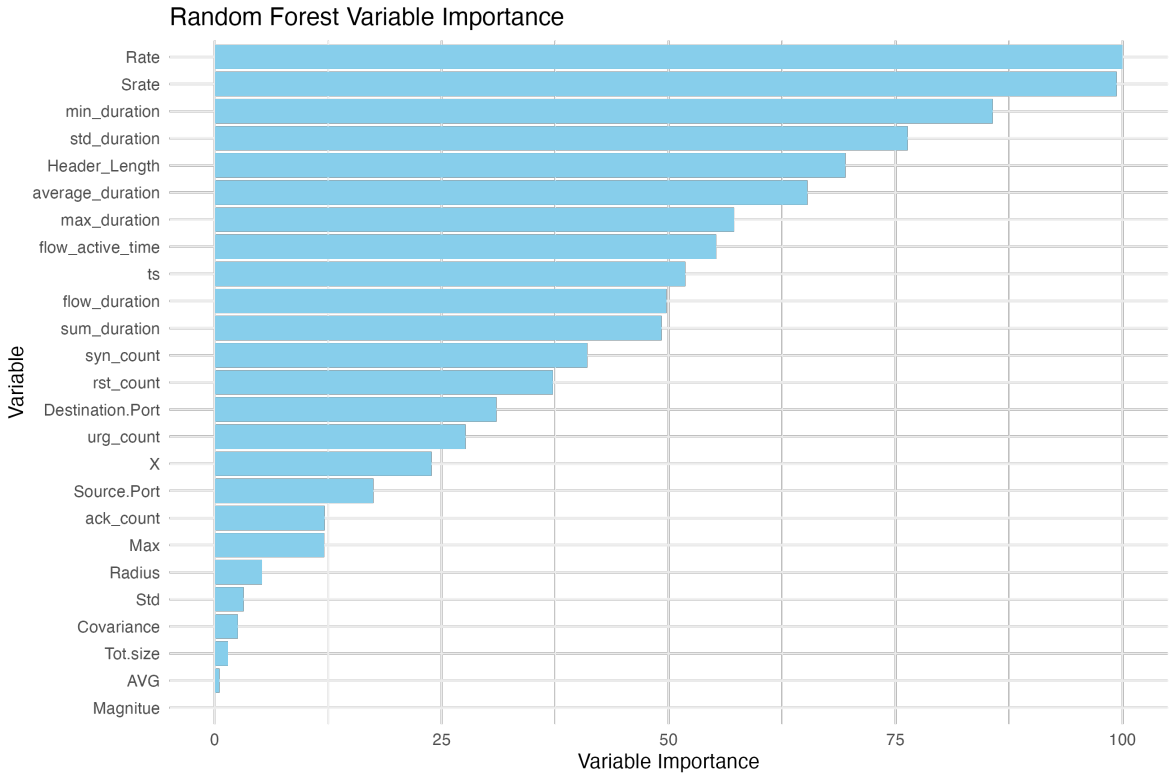

Figure 9: The suspected IP Addresses

Recall	Specificity	F1 Score	Accuracy
0.992	0.9996	0.99588	0.99998

Table 1: Performance Metrics of Random Forest

6 Alternative Methods

Our original idea was to utilize the relationships between all of the IP addresses, treating each as a node, and the interactions between them as edges. We created a feature for every edge between each node and aggregated over all of them to create a full feature set totaling of about 3000 different features.

We then used principle component analysis for dimension reduction and attempted to run a changepoint analysis on the first principle component or a multivariate changepoint analysis on the first 5 principle components. However, the changepoint did not perform well as only one of the 5 principle components proved useful in determining the correct changepoint.

Additionally, we attempted to use LDA to predict the attack type; however, it did not perform nearly as well as the random forest did, so we decided to go with the random forest as our model.

7 Discussion

Within our changepoint detection model, we decided to use flow idle time as it displayed the clearest shift between phase 1 and 2. However, if we wanted our model to be more robust, we would need a way to incorporate more features in detecting our changepoint. Thus, without the time constraint, we could have explored multivariate changepoint methods along with additional dimension reduction techniques. Furthermore, we could have utilized external data to tune our hyperparameters, such as the threshold value and number of quantiles, for our online changepoint model.

After detecting the attack, determining the suspect IP addresses, predicting which interactions are malicious, and classifying what type/stage of attack is being performed, the next step is prevention. Using the information that we gathered about the attackers and the attack, we can implement a prevention plan for each attack type identified.

8 Conclusion

This paper examined methods where we could detect anomalies within a data stream while classifying potential malicious IP addresses and stages of attacks. With ever adapting nature of cyberattacks, we searched for models that would be robust enough to be incorporated into different pipelines of various cybersecurity measures.

To accomplish this goal, we created a model that successfully detects a changepoint when an attack begins in real time. Our model also identifies suspicious IP Addresses to determine where the malicious activity stems from and prevent it. After detecting the significant change point, we were able to successfully identify the specific stage of an attack using a Random Forest model.

Our changepoint model correctly identified changepoint by time 7829 (minutes since experiment began) and then was able to determine which IP addresses were malicious. The Random Forest model correctly classified the malicious network activity with an average recall of 0.992.

9 Github Repository

Our code can be found at `https://github.com/quintbro/data-dudes`.

References

[AM07] Ryan Prescott Adams and David J. C. MacKay. Bayesian online changepoint detection, 2007.

[AREF23] Edward Austin, Gaetano Romano, Idris A. Eckley, and Paul Fearnhead. Online non-parametric changepoint detection with application to monitoring operational performance of network devices. *Computational Statistics Data Analysis*, 177:107551, 2023.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[Die00] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[Pag54] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

[REF24] Gaetano Romano, Idris A. Eckley, and Paul Fearnhead. A log-linear nonparametric online changepoint detection algorithm based on functional pruning. *IEEE Transactions on Signal Processing*, 72:594–606, 2024.

[Roß18] Peter Roßbach. Neural networks vs. random forests–does it always have to be deep learning. *Germany: Frankfurt School of Finance and Management*, pages 1–8, 2018.