Scientific Eminence Group

Journal of Information Security and its Applications

Research Article

Open Access

jisa@scientificeminencegroup.com

# Cyber-Eye: A Novel Web Extension for Real-Time Detection and Mitigation of DDoS and DoH Threats

**Mohamed Khaled, Abdelrahman Alaa, Mariam Maged, Shady Amr, Tarek Talaat and Hossam Abdelrahman***

*Misr International University, KM 28 Cairo – Ismailia Road, Ahmed Orabi District, Egypt*

## *Corresponding Author

Hossam Abdelrahman, Misr International University, KM 28 Cairo – Ismailia Road, Ahmed Orabi District, Egypt, E-mail: hossam.csc0225@miuegypt.edu.eg

## Citation

Mohamed Khaled, Abdelrahman Alaa, Mariam Maged, Shady Amr, Tarek Talaat, et al. (2024) Cyber-Eye: A Novel Web Extension for Real-Time Detection and Mitigation of DDoS and DoH Threats. J. Inf. Secur. Appl. 2: 1-14

## Publication Dates

Received date: September 15, 2024
Accepted date: October 15, 2024
Published date: October 18, 2024

## Abstract

In today's digital landscape, unexpected website downtimes are often the result of DDoS attacks—malicious attempts to overload sites with fake traffic. An equally sinister threat comes from DNS over HTTPS (DoH) attacks, where attackers conceal nefarious activities within encrypted DNS queries. CyberEye, an innovative web extension, leverages advanced machine learning techniques to discern between benign and malicious traffic, thus enhancing cybersecurity measures. It effectively mitigates the risks associated with DDoS and DoH threats, offering website owners and cybersecurity experts a robust solution to maintain site integrity and operational continuity.

**Keywords:** DDoS; DoH; Decision Tree; Gaussian Naive Bayes; KNN

## I. Introduction

In the quickly changing digital world of today, unplanned website outages are becoming more common. These attacks involve Distributed Denial of Service (DDOS), where malicious actors launch coordinated efforts to overload websites with fake traffic, thereby disrupting services and causing inconvenience to users. Furthermore, DNS over HTTPS (DoH) attacks represent a growing threat to cybersecurity. These attacks use encrypted DNS queries to mask malicious activity, making it difficult to identify and stop such intrusions. The first documented DDoS-style attack occurred on February 7, 2000, when "mafiaboy," a 15-year-old Canadian hacker enthusiast, developed a series of Denial of Service attacks against several e-commerce sites. These attacks used computers at multiple locations to overwhelm the vendors' computers and shut down their sites [1].

Many attackers use spoofing techniques to modify their source address, (which is used to conceal the attacker's IP or MAC address via generating a random IP/MAC address or using a trusted device source identification [2]. This ends up resulting in a lack of awareness of the threat and preventing any counterattack preparations. Attacks will not be detected unless they are published by the author or identified by a third party. DoH is recommended to secure the connection between endusers and recursive resolvers to address privacy concerns in DNS. It uses HTTPS to encrypt DNS requests. DoH uses TCP port 443, just like HTTPS. DNS requests are delivered as URI templates. The domain name in the URI is used for both finding the IP address of the DoH resolver (using unencrypted DNS resolution) and verifying its identity (using SSL certificate verification). Browsers commonly include DoH as an integrated module [3].

To mitigate these threats and protect DNS's authenticity, confidentiality, and integrity, we present in this paper a detailed roadmap for CyberEye, which is a web extension to help mitigate the effects of botnet attacks on websites from potential cyber threats, including the detection of DDoS and DOH attacks. It describes how the extension functions, from monitoring web traffic to distinguishing between harmless visitors and malicious attacks using machine learning models, all through a user-friendly interface that doesn't require advanced technical knowledge to get the hang of. The extension aims to resolve issues regarding the malicious activities being carried out on botnet network devices that utilize DDoS and DoH attacks through fundamental.

We introduce a web extension designed to safeguard websites against Botnet DDoS and DoH attacks through the innovative use of machine learning and AI. Its core functionality revolves around the real-time analysis of web traffic, which is part of its revenue model, effectively distinguishing between legitimate packets that cause no harm and those that would otherwise be labelled as harmful. By leveraging advanced algorithms and utilizing the most efficient ones out of them, like Random Forest and Logistic Regression. The tool doesn't just identify potential botnet DDoS and DoH attacks, but also learns from them, the more it learns, the better it gets, and in turn enhances its defense mechanisms over time.

The structure of the remaining paper is structured as follows: Section II provides a comprehensive overview of the work related to our research, exploring various studies and their methodologies. In Section III, we delve into our Proposed Methodology, detailing the datasets we utilized in subsection 1.1. This includes a description of the data sources, their characteristics, and why they are suited for this study. Subsection 1.2 discusses the algorithms we employed, outlining each method's principles and its relevance to our research. Section IV is dedicated to presenting our Results and Analysis, where we interpret the data findings and evaluate the effectiveness of the applied algorithms. Section V wraps up the paper with our conclusions, summarizing the key insights and outcomes of our research. Finally, Section VI outlines potential future work, suggesting directions for further research and possible enhancements to our methodology.

## II. Related Work

In recent years, the landscape of DDoS and DoH detection tools has evolved significantly, with numerous studies exploring advanced methodologies to enhance detection accuracy and efficiency. This section reviews contemporary research and compares various tools and techniques.

### A. DDoS Detection Tools

- **Deep Learning Approaches:** Mittal et al. (2023) conducted a systematic review of deep learning approaches for detecting DDoS attacks. The study categorizes existing literature into different types of

DDoS attack detection methods, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The review highlights the strengths and weaknesses of these approaches, emphasizing the need for large datasets and extensive computational resources.

- **Software-Defined Networking (SDN) Based Solutions:** Jain et al. (2024) provided a comprehensive survey on DDoS detection, mitigation, and defense strategies in SDN environments. The study discusses various detection techniques, including entropy-based methods and machine learning models, and compares their effectiveness in real-time scenarios. The survey also identifies open challenges and future research directions.

- **Comparative Analysis of Machine Learning Techniques:** A recent study by Kumar et al. (2024) compared six machine learning techniques—Random Forest, Decision Tree, AdaBoost, Extreme Gradient Boosting, Multilayer Perceptron, and Dense Neural Network—for classifying DDoS attacks. The results indicated that Random Forest and Extreme Gradient Boosting provided the highest accuracy.

## B. DoH Detection Tools

- **Statistical Pattern Recognition:** Niktabe et al. (2023) proposed two statistical pattern recognition models based on logistic and linear regression for detecting and profiling malicious DoH traffic. The study utilized the CIRA-CIC-DoHBrw-2020 dataset and demonstrated that logistic regression outperformed linear regression in identifying malicious patterns.

- **Behavioral Profiling with Interpretable Machine Learning:** Another study by Niktabe et al. (2023) introduced a behavioral profiling model using inherently interpretable machine learning methods, such as Decision Trees and Random Forest. The model aimed to profile malicious and benign DoH traffic, achieving high accuracy with a balanced dataset.

- **Deep Learning for Encrypted Traffic:** A study by Jerabek et al. (2023) focused on detecting DoH traffic tunnels using deep learning techniques. The research employed eight base learner classifiers and the CIRA-CIC-DoHBrw-2020 dataset, highlighting the effectiveness of deep learning in handling encrypted traffic.

**Table 1:** Comparision of the Discussed Tools and Methodolgies

| Study | Methodology | Dataset | Key Findings | Strengths | Weaknesses |
|---|---|---|---|---|---|
| Mittal et al. (2023) | Deep Learning (CNNs, RNNs) | Various | High accuracy with large datasets | Handles complex patterns | Requires extensive computational resources |
| Jain et al. (2024) | SDN-based, Entropy, ML | Various | Effective in real-time | Centralized control, flexibility | Security concerns in SDN |
| Kumar et al. (2024) | ML (RF, DT, ADA, XGB, MLP, DNN) | Various | RF and XGB highest accuracy | Comprehensive comparison | Computationally intensive |
| Niktabe et al. (2023) | Statistical Models (Logistic, Linear Regression) | CIRA-CIC-DoHBrw-2020 | Logistic regression outperforms | Low computational complexity | Lower accuracy than ML/DL models |
| Niktabe et al. (2023) | Interpretable ML (DT, RF) | Balanced CIRA-CIC-DoHBrw-2020 | High accuracy, interpretable | Robust to noise | Requires feature engineering |
| Jerabek et al. (2023) | Deep Learning | CIRA-CIC-DoHBrw-2020 | Effective for encrypted traffic | High accuracy | High computational cost |

In [4]. Through network behaviour analysis, a lightweight malware detection system is proposed in this system that watches the network activity of questionable apps. The suggested lightweight network-based malware detection system is seen in the Fig. 1, below. It uses machine learning to differentiate between malware and goodware APKs based on how they behave on the network over time. Their proposal creates a feature vector for every network packet and uses a flow-based method to extract numerical characteristics from the TCP/IP packet header. For the dataset, they created a script for the good ware group to get 500 most downloaded APKs from the Google Play Store using the gplayclitool. On the other hand, samples from the Android Malware Dataset are broken down into 71 malware families, totaling 24,553 malware samples. A prototype of the system was built using the Random Forest and AdaBoost machine learning algorithms in a Samsung Galaxy S9+. The system results demonstrate that, in terms of classification performance, it is possible to detect malware variants using network features. Even though the classification performance per packet only reaches about 80% accuracy, the Anomaly Score allows them to distinguish between malware and goodware in APK classifications with almost 90% accuracy. In order to accurately categorize an application and expand processing assessment to a large number of applications, the minimum traffic volume required should be optimized.

In [5]. Two techniques for identifying malware have been proposed: Static Analysis and Dynamic Analysis. The two most often utilized detection attributes are permissions and network traffic. Malicious programs download malware during runtime to avoid detection based on static permissions. Malware that does not need to be connected to a network tends to avoid network traffic-based detection which can be detected by permissions analysis. Using the FP-Growth algorithm, they train and test the suggested model to produce frequent patterns made up of permissions and traffic data. NTPDroid is their proposed hybrid Android malware detection solution. It is a prototype tool they used to extract traffic features and permissions from the applications through two phases. In the Analysis phase, using a combination of traffic attributes and permissions, they create regular malware patterns and normal datasets. This regular pattern creation can assist us in analyzing the patterns that are considerably present in both malware and healthy samples. These patterns comprise both traffic attributes and permissions. Throughout the detection phase, they identify the malicious applications by utilizing these recurring patterns. The Genome Malware dataset is used in their system. The results which show a detection accuracy of 94.25% demonstrated that, in comparison to employing either the traffic features or permissions, combining the two improved the detection rate.

In [6]. Their proposal is about presenting a two-layer Android malware analyzer based on static and dynamic features and enhancing their previous network-flow analyses with appending extracted network-gram sequential relations of API calls. The dataset was divided into two parts. First is putting out a comparison of the Android malware datasets that were previously accessible, taking into account 15 crucial factors. The second part used the CICAndMal2017 dataset, which consists of API calls as dynamic features and Permission and Intent as static features. The analysis consists of two layers Static Binary Classification (SBC) and Dynamic Malware Classification (DMC). For DMC, it attempts to classify the input malware samples into four malware categories (Adware, ransomware, SMS malware, and scareware) and 39 malware families. Their results show that they were successful in attaining 83.3% precision in dynamic-based malware category classification, 59.7% precision in dynamic-based malware family classification, and 95.3% precision in static-based malware binary classification at the first layer. Despite maintaining relatively low false-positive rates, they were able to improve results to 95.3% recall in malware binary classification, 81.0% recall in malware category classification, and 61.2% recall in malware family classification.

In [7]. The proposed solution was to use popular and effective Machine Learning models such as Decision Tree, and Naïve Bayes. To increase the performance, they used Random Forest, Stochastic Gradient Boosting, and AdaBoost. Sklearn framework was used to train their models. There are 28879 samples in the dataset, comprising 12290 benign applications and 16589 Android malware instances. Android malware is gathered from Virusshare and Koodous. They collected 65 features concerning behavior and the permissions of the application. Based on the obtained results, they determined that the Random Forest algorithm yields the best accuracy of 98.66%. In order to assess the viability of their framework, they are planning to install it on Android smartphones and measure the speed at which apps scan on various Android devices.

## III. Proposed Methodology

We attempt to test each of Logistic Regression, and Random Forest Classifiers on our datasets for both the DDoS dataset and the DoH dataset which is later used after the DDoS one. Firstly, the DDoS dataset contains 199,916 sample records. The dataset has 11 features, and whether or not the packet is malicious. We built a testing framework for the previously mentioned models, establishing datasets and testing scenarios. Datasets with categorical values need preprocessing. To enable machine learning models to exploit the information, it must be encoded into numerical values. The pre-processed dataset was separated into training and testing records. The model's performance is evaluated by comparing the predictions to the actual label values and presenting results in tables. Secondly, the DoH one was tested and trained to provide the highest accuracy with the highest detection rate using the most profound machine learning classifier algorithms and utilizing each and every column to come up with the best results in order to provide the user with safe web surfing experience. The total number of columns in this dataset are the same number as the previous one totaling at 11 columns 10 of which are features and one that's exclusively for classification. We've conducted multiple tests using multiple different classifiers like Logistic Regression, Naive Bayes, and Random Forest, we've settled on Random Forest as it proved to be superior in our case and had the highest accuracy compared to the other two proposed classifiers.

### A.Datasets Descriptions

The study utilizes two distinct datasets for evaluating the performance of the CyberEye extension: one for DDoS attacks and another for DoH attacks.

### 1. DDoS Dataset

The DDoS dataset comprises 199,916 sample records, each representing a unique network flow characterized by various attributes such as source and destination IP addresses, port numbers, protocol types, flow duration, and packet counts. This dataset is meticulously designed to capture the nuances of DDoS attacks within network traffic, aiding in the accurate prediction and analysis of such attacks.

Features of the DDoS Dataset:

- Flow ID: Unique identifier for each flow.
- Src IP: Source IP address.
- Src Port: Source port number.
- Dst IP: Destination IP address.
- Dst Port: Destination port number.
- Protocol: Protocol number.
- Flow Duration: Duration of the flow.
- Tot Fwd Pkts: Total forward packets.
- Tot Bwd Pkts: Total backward packets.
- Flow IAT Min: Minimum inter-arrival time of packets.
- Fwd Seg Size Min: Minimum size of forward segments.
- Label: Indicates whether the flow is benign or malicious.

### 2. DoH Dataset

The DoH dataset contains 586,758 entries, capturing various aspects of network traffic generated from DoH activities. This dataset includes traffic from browsers like Google Chrome and Firefox, with malicious traffic extracted using tunneling tools such as DNS2TCP, DNSCat2, and Iodine.

Features of the DoH Dataset:

- SourceIP: Source IP address.
- DestinationIP: Destination IP address.
- SourcePort: Source port number.
- DestinationPort: Destination port number.
- TimeStamp: Timestamp of the flow's start.
- Duration: Duration of the flow.
- FlowBytesSent: Total bytes sent in the flow.
- FlowSentRate: Rate of bytes sent per second.

- FlowBytesReceived: Total bytes received in the flow.
- FlowReceivedRate: Rate of bytes received per second.
- Label: Indicates if the flow is benign or malicious.

### B. Dataset Diversity and Representativeness

While the datasets provide a substantial amount of data for training and testing machine learning models, it is crucial to acknowledge their limitations and potential biases:

#### Source Diversity

The DDoS dataset primarily focuses on specific types of network traffic, which may not encompass the full spectrum of DDoS attack vectors encountered in diverse network environments.

The DoH dataset includes traffic from popular browsers and specific tunneling tools, which might not represent all possible DoH attack scenarios.

#### Environmental Representativeness

Both datasets are collected from controlled environments, which may not fully capture the variability and complexity of real-world network conditions.

The datasets might lack representation from different geographical regions, network topologies, and organizational infrastructures, potentially limiting the generalizability of the findings.

#### Temporal Bias

The datasets might not account for the evolving nature of DDoS and DoH attacks, where new attack techniques and patterns emerge over time. This temporal bias can affect the model's ability to detect novel threats.

#### Class Imbalance

There could be an imbalance between benign and malicious samples within the datasets, which can skew the model's performance. Techniques such as oversampling, undersampling, or synthetic data generation might be necessary to address this issue.

### C. Addressing Dataset Limitations

To mitigate these limitations, future work should consider:

- Expanding Data Collection: Incorporating data from a wider range of sources, including different network environments, geographical locations, and organizational settings.
- Continuous Updating: Regularly updating the datasets to include the latest attack patterns and techniques, ensuring the models remain effective against emerging threats.
- Balancing Classes: Employing data balancing techniques to ensure a more equitable representation of benign and malicious samples, enhancing the model's robustness.

The first dataset which naively speaks about DDoS [II] as it incorporates many different columns that contribute to an accurate measurement, this dataset was meticulously and particularly designed for the prediction and analysis of DoS attacks within network traffic, as well as determining whether it's actually an attack or if it's just benign. It consists of 199,916 entries, each record in the dataset is a unique network flow, whether it's forwards or backwards (send or receive), characterized by a combination of attributes that describe the specifics of the traffic flow, including source and destination details, protocol used, and various metrics related to the flow's size and timing.

The second dataset [III], which we've discussed before to be DoH related, tends to highlight the different network aspects necessary to dissect and dig into the specific types of traffic that are generated from DoH traffic. The dataset contains major elements that could be used to flag a packet, whether it's benign or a DoH attack is comprised of and contains about 586,758 different entries that span from different sources that they originate from. They come from Google Chrome and Firefox, and the malicious traffic was extracted using DNS2TCP, DNSCat2, and Iodine, which are tunneling tools used to inject code into yet-to-be encrypted DoH traffic.

**Table 2:** Features of Ddos Dataset [8]

| Feature | Type | Description |
|---|---|---|
| Flow ID | Numerical | Unique identifier for each flow, ranging from 350 to 850 |
| Src IP | Categorical | Integer representation of source IP addresses |
| Src Port | Numerical | Source port numbers, ranging from 0 to 65535 |
| Dst IP | Categorical | Integer representation of destination IP addresses |
| Dst Port | Numerical | Destination port numbers, ranging from 0 to 10 |
| Protocol | Numerical | Protocol numbers, typically ranging from 0 to 255 for common protocols |
| Flow Duration | Numerical | Duration of the flow in minutes, from -1 to 120 |
| Tot Fwd Pkts | Numerical | Total forward packets in a flow, ranging from 0 to 310,000 |
| Tot Bwd Pkts | Numerical | Total backward packets in a flow, ranging from 0 to 292,000 |
| Flow IAT Min | Numerical | Minimum inter-arrival time of packets in the flow, from -13 to 120 minutes |
| Fwd Seg Size Min | Numerical | Minimum size of forward segments, from 0 to 48 bytes |
| Label | Categorical | Whether the flow is benign or malicious |

**Table 3:** Features of Doh Dataset [9]

| Feature | Type | Description |
|---|---|---|
| SourceIP | Numerical | Integer representation of source IP addresses |
| DestinationIP | Numerical | Integer representation of destination IP addresses |
| SourcePort | Numerical | Source port numbers, ranging from 0 to 65535 |
| DestinationPort | Numerical | Destination port numbers, ranging from 0 to 65535 |
| TimeStamp | Numerical | Timestamp of the flow's start, in UNIX epoch time |
| Duration | Numerical | Duration of the flow in seconds |
| FlowBytesSent | Numerical | Total bytes sent in the flow |
| FlowSentRate | Numerical | Rate of bytes sent per second in the flow |
| FlowBytesReceived | Numerical | Total bytes received in the flow |
| FlowReceivedRate | Numerical | Rate of bytes received per second in the flow |
| Label | Categorical | Indicates if the flow is benign or malicious |

### B. Used Algorithms

Decision Tree, Gaussian Naive Bayes, KNN, Logistic Regression, and Random Forest were the five machine learning algorithm classifiers that were used for the purpose of training and testing, wherein we've conducted multiple comprehensive tests on each and every one of these classifiers so that both datasets can be utilized to their maximum whilst using any of the proposed machine learning algorithm classifiers. Several preprocessing steps were executed prior to conducting the process of machine learning in order to output correct, realistic, and fair scores for each one of the classifiers hence feeding all of the features into each classifier and converting all categorical values to numerical ones since some classifiers tend to only accept numerical ones.

### Decision Tree

Decision trees are a form of supervised learning used primarily for classification, though they also handle regression tasks. This method employs a hierarchical tree structure to systematically split the data, beginning at the root and progressing to binary outcomes at the leaves. At each node, the data is divided using criteria such as specific values, ranges, or thresholds based on probability distributions. Decision trees are valuable for pinpointing critical predictors in data, making them adaptable to various types of input data. The decision tree can be calculated using the following rule [10]

$$y = -\sum_{i=1}^{k} p_i \log_k(p_i) \qquad (1)$$

### Gaussian Naive Bayes

Naive Bayes is a classification algorithm widely utilized in machine learning for different types of classification tasks. It operates based on Bayes' theorem, a crucial theorem that calculates the probability of an event based on prior knowledge of conditions that might be related to the event. To determine the class of a new instance, Naive Bayes evaluates the probability of each class label based on the feature values, then selects the label with the highest probability. Nonetheless, it relies on an assumption of conditional independence among features, which isn't always valid, especially when features are strongly correlated. This can affect the algorithm's accuracy in some scenarios.

Gaussian Naive Bayes is a form of the Naive Bayes algorithm tailored for classification tasks involving features that follow a Gaussian distribution. This approach estimates the likelihood of each class based on the attributes of the features. It computes the product of these probabilities across all features and assigns to the instance the class label that has the highest resultant probability [11].

The Gaussian Naïve Bayes can be calculated Through the Bayes rule:

$$P(Y|X) = P(X|Y) * P(Y)/P(X) \qquad (2)$$

### K – Nearest Neighbor

K Nearest Neighbors (KNN) is a straightforward supervised classification algorithm. It extends the simple nearest-neighbor rule by considering the 'k' closest samples to the class label being tested, rather than just the nearest one. Unlike the basic nearest neighbor approach, KNN leverages the class labels of these 'k' neighbors during the decision-making phase, thus enriching the information available for making predictions. This method allows KNN to bypass the typical learning phase, setting it apart from many other classification algorithms by focusing directly on inference based on the closest examples.

K-Nearest neighbor is measured by the following distance rules: [12].

$$\sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \qquad (3)$$

$$\sum_{i=1}^{k} |x_i - y_i| \qquad (4)$$

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q} \quad (5)$$

### Logistic Regression

Logistic regression is a type of supervised learning used primarily for predicting the outcome of a categorical dependent variable, such as Yes or No, 0 or 1. Rather than directly predicting these discrete outcomes, logistic regression estimates the probability of the occurrence of an event, assigning a value closer to 1 when an event is more likely to happen. This method is particularly useful in scenarios such as determining loan approvals, where the decision to approve or deny a loan is based on the predicted probability of a borrower defaulting [13].

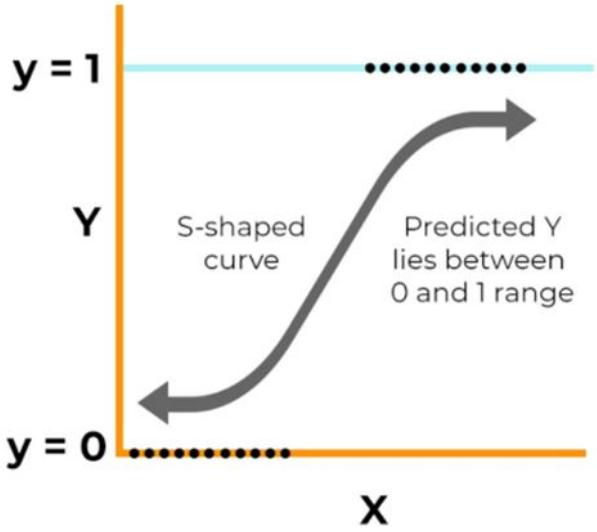


**Figure 1:** Logistic regression curve [14]

Logistic regression uses a sigmoid function to transform predictions and their probabilities, as shown in Fig 1. The sigmoid function is an S-shaped curve that maps any real value to a range between 0 and 1. The sigmoid function: [15].

$$y = \frac{e^{(b_0 + b_x x)}}{1 + e^{(b_0 + b_1 x)}} \quad (6)$$

### Random Forest

A random forest algorithm is made up of multiple decision trees and utilizes a technique known as bagging to enhance prediction accuracy. Bagging, or bootstrap aggregating, involves using different subsets of the training data to train each decision tree in the forest. By randomly selecting both features and observations from the dataset for each tree, the model generates a variety of predictions. The random forest then combines these individual tree results, typically by averaging, to produce a final prediction. The overall accuracy of the model generally increases with the number of trees in the forest, as this diversifies the predictions and mitigates overfitting [16].

## Results and Analysis

For the DDoS dataset, we deployed, implemented, and fully integrated two machine learning models, following extensive data preprocessing to cleanse the dataset and eliminate potential errors. The two machine learning classifiers chosen for this task are Random Forest and Logistic Regression, each offering unique advantages in identifying DDoS attacks:

- Random Forest: Combines multiple decision trees into an ensemble model, which enhances classification accuracy by reducing overfitting and managing diverse features in DDoS traffic. Its ability to handle imbalanced data makes it suitable for datasets with varying traffic patterns, like DDoS attacks.

- Logistic Regression: Provides a straightforward yet powerful classification mechanism well-suited for binary classification problems. Despite its simplicity, it can effectively distinguish between benign and malicious traffic, making it useful in scenarios where quick response time is essential.

With both models tested and validated, their high accuracy in detecting botnet DDoS attacks confirms the value of their application:

- Random Forest Classifier - Accuracy: 99%

- Logistic Regression Classifier - Accuracy: 93%

These models provide a reliable basis for detecting DDoS attacks across various devices and networks, ensuring consistent performance for practical implementation.

```
Logistic Regression Accuracy: 0.9352691538113987
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     79038
           1       0.92      0.90      0.91     44905
           2       0.00      0.00      0.00         1

    accuracy                           0.94    123944
   macro avg       0.62      0.62      0.62    123944
weighted avg       0.94      0.94      0.94    123944

Random Forest Accuracy: 0.9986687536306719
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     79038
           1       1.00      1.00      1.00     44905
           2       0.00      0.00      0.00         1

    accuracy                           1.00    123944
   macro avg       0.67      0.67      0.67    123944
weighted avg       1.00      1.00      1.00    123944
```

**Figure 2:** Machine Learning Accuracy – DdoS classifier

The reason why these two specifically were used is as follows:

- Random Forest: Combines multiple decision trees to improve accuracy and reduce overfitting, ideal for datasets with diverse features like DoH.

- Gradient Boosting: Sequentially builds trees to correct previous errors, achieving high precision and recall for distinguishing between benign and malicious traffic.

Now that we've established a base of understanding. The machine learning code yielded exceptional results for both classifiers since they both got extremely similar accuracy's, after extensive decision-making we've settled on using Random Forest classifier to make it uniform between DDoS and DoH capturing and detection. To highlight the accuracy of both models, they achieved the following:

- Random Forest Classifier - Accuracy: 98.67%

- Gradient Boosting Classifier - Accuracy: 98.63%

```
Random Forest Performance:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99    108044
           1       0.86      0.99      0.92      9308

    accuracy                           0.99    117352
   macro avg       0.93      0.99      0.96    117352
weighted avg       0.99      0.99      0.99    117352

Random Forest Accuracy: 0.9867663099052424

Gradient Boosting Performance:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99    108044
           1       0.99      0.84      0.91      9308

    accuracy                           0.99    117352
   macro avg       0.99      0.92      0.95    117352
weighted avg       0.99      0.99      0.99    117352

Gradient Boosting Accuracy: 0.9863231985820438
```

**Figure 3:** Machine Learning Accuracy – DoH

```python
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
import pandas as pd

# Load the dataset
file_path = '/content/DoH.csv'
data = pd.read_csv(file_path)

# Define features and target variable
X = data.drop('Label', axis=1)
y = data['Label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)

# Initialize Random Forest model
rf_model = RandomForestClassifier(n_estimators=10, max_depth=3, random_state=42, class_weight='balanced')
rf_model.fit(X_train, y_train)

# Predict and evaluate Random Forest
y_pred_rf = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, y_pred_rf)
rf_report = classification_report(y_test, y_pred_rf)
print("Random Forest Performance:")
print(rf_report)
print(f"Random Forest Accuracy: {rf_accuracy}")

# Initialize Gradient Boosting model
gb_model = GradientBoostingClassifier(n_estimators=10, max_depth=3, random_state=42)
gb_model.fit(X_train, y_train)

# Predict and evaluate Gradient Boosting
y_pred_gb = gb_model.predict(X_test)
gb_accuracy = accuracy_score(y_test, y_pred_gb)
gb_report = classification_report(y_test, y_pred_gb)
print("\nGradient Boosting Performance:")
print(gb_report)
print(f"Gradient Boosting Accuracy: {gb_accuracy}")
```

**Figure 4:** Machine Learning Code - DoH

The study employs several machine learning classifiers, including Decision Tree, Gaussian Naive Bayes, K-Nearest Neighbor (KNN), Logistic Regression, and Random Forest. After extensive testing, Random Forest was chosen as the final model for both DDoS and DoH detection. This section provides a detailed comparison and justification for this

choice.

**1. Decision Tree**

- Strengths: Simple to understand and interpret, handles both numerical and categorical data.
- Weaknesses: Prone to overfitting, especially with noisy data.

**2. Gaussian Naive Bayes**

- Strengths: Fast and efficient, works well with small datasets.
- Weaknesses: Assumes independence between features, which is often not the case in real-world data, leading to lower accuracy.

**3. K-Nearest Neighbor (KNN)**

- Strengths: Simple and intuitive, effective with small datasets.
- Weaknesses: Computationally expensive with large datasets, sensitive to irrelevant features and the choice of k.

**4. Logistic Regression**

- Strengths: Good for binary classification, interpretable results.
- Weaknesses: Assumes a linear relationship between features and the log odds of the outcome, which may not hold true for complex datasets.

**5. Random Forest**

- Strengths:
- Handles Imbalanced Data: Random Forest can handle imbalanced datasets effectively by using techniques such as class weighting and bootstrapping.
- Reduces Overfitting: By averaging multiple decision trees, Random Forest reduces the risk of overfitting, which is a common issue with single decision trees.
- Handles High Dimensionality: It can manage datasets with a large number of features and complex interactions between them.
- Robustness: Provides high accuracy and robustness against noise in the data.
- Feature Importance: Offers insights into feature importance, helping to understand which features contribute most to the prediction.

**Comparison and Justification**

**Dataset Characteristics**

DDoS Dataset: The DDoS dataset contains 199,916 records with 11 features. The diversity and complexity of the features, such as flow duration, packet counts, and inter-arrival times, make Random Forest a suitable choice due to its ability to handle high-dimensional data and complex interactions.

DoH Dataset: The DoH dataset includes 586,758 entries with similar complexity. Random Forest's robustness and ability to handle large datasets with many features make it ideal for this context.

**Imbalanced Data**

Both datasets may have an imbalance between benign and malicious samples. Random Forest's ability to handle imbalanced data through techniques like bootstrapping and class weighting ensures better performance compared to other models.

**Performance**

- Accuracy: Random Forest achieved the highest accuracy in both DDoS (99%) and DoH (98.67%) detection, outperforming other models like Logistic Regression and KNN.
- Generalization: The ensemble nature of Random Forest helps in generalizing well to unseen data, reducing the risk of overfitting.

Given these advantages, Random Forest was chosen as the final model for both DDoS and DoH detection, providing a balance between accuracy, robustness, and interpretability.

## V. Conclusion

To summarize and conclude our work and what steps were taken to come up with CyberEye in its market-ready form, FlaskAPI, Scapy, and Streamlit were utilized in order to manipulate packets, track them, and generate the web extension. They were also used by means of integrating the machine learning code via a .PKL file with the backend code that sniffs incoming and outgoing packets and provides the user with live, accurate and time-serious results of packets that are sent from or to them. The CyberEye web extension aims to put the user's in as less of a threat as possible by detecting whether that packets is malicious or benign thus making it safer and easier for them to surf the web without restrictions and without fear.

## VI. Future Work

For the future work that we're planning on doing is we're planning on getting an actual server that we can then send incoming traffic to and it sort of acts like an IDS system where it scan incoming DoH traffic, signals it and flags it whether it's benign or an attack through decryption of the actual packet before this packet is further into the user's network to avoid any DoH-based attack that could compromise the user's PC.. Furthermore, we're looking forward to employing a model that serves free users and users that are willing to pay extra, providing them with more security through subscription of our premium services that monitor the user's network continuously to make sure they're always safeguarded against DDoS, DoH, or any other attack that could be carried over the Internet. Furthermore, we wish to make the design more user-friendly and have multiple interfaces that serve people of different ages or different technological background that provides them with more control over what they want to be protected from or what information specific network do they want to be protected.

## VII. References

1. Encyclopædia Britannica (2024) Denialof-service attack. [Online]. Available: https://www. britannica.com/technology/-denial-of-service-attack

2. R Abubakar, A Aldegheishem, MFMajeed, A Mehmood, H Maryam, et al. (2020) An effective mechanism to mitigate real-time ddos attack, IEEE Access, 8: 126215-27.

3. T Zebin, S Rezvy, Y Luo (2022) An explainable aibased intrusion detection system for dns over https (doh) attacks, IEEE Transactions on Information Forensics and Security, 17: 2339-49.

4. IJ Sanz, MA Lopez, EK Viegas, VR Sanches (2020) A lightweight network-based android malware detection system, 695-703.

5. A Arora, SK Peddoju (2018) Ntpdroid: a hybrid android malware detector using network traffic and system permissions, 808-13.

6. L Taheri, AFA Kadir, AH Lashkari, (2019) Extensible android malware detection and family classification using network-flows and api-calls, 1-8.

7. NC Lê, TM Nguyen, T Truong, ND Nguyen, T Ngô (2020) A machine learning approach for real time android malware detection, 1-6.

8. DEVENDRA. Available: https://www.kaggle.com/datasets/devendra416/ddos-datasets

9. P Friedrich (2020) DNS Test Traffic (DoH/BRW2020), https://www.kaggle.com/datasets/peterfriedrich1/dns-test-traffic-dohbrw2020

10. S Pathak, I Mishra, A Swetapadma (2018) An assessment of decision tree based classification and regression algorithms, in 2018 3rd International Conference on Inventive Computation Technologies (ICICT). IEEE, 92-5.

11. R Vats (2021) Gaussian naïve bayes, https://www.upgrad.com/blog/gaussian-naivebayes/?msclkid=658123f7d04811ec8608a267e841a654

12. CS Lee, PYS Cheang, M Moslehpour (2022) Predictive analytics in business analytics: decision tree, Advances in Decision Sciences, 26: 1-29.

13. edX (2022) What is logistic regression? https://www.mastersindatascience.org/learning/machinelearning-algorithms/logistic-regression/

14. vijaya kanade (2022) What is logistic regression? equation, assumptions, types, and best practices, https://www.spiceworks.com/tech/artificialintelligence/articles/what-is-logisticregression/lg=1slide=0

15. L Connelly (2020) Logistic regression, Medsurg Nursing, 29: 353-4.

16. O Mbaabu (2020) Introduction to random forest in machine learning, Berreskuratua-(e) tik https://www.Section.io/engineering-education/introduction-to-random-forestin--machine-learning.