DDoS Attack Detection Using Machine Learning (XGBoost Algorithm)

Arnav Dham 2022A7PS1182P
Rakshita Goel 2022B1A71108P
Under the guidance of Prof. Haribabu K and Prof. Smriti Arora

Acknowledgement

We wish to extend our deepest gratitude to Dr. Haribabu K., Associate Professor at BITS Pilani, and Prof. Smriti Arora for their unwavering support and mentorship throughout our work on DDoS Attack Detection Using Machine Learning (XGBoost Algorithm). Their profound knowledge of DDoS attacks and existing protocols has been invaluable in shaping our understanding of this complex and critical domain. We are profoundly thankful for the opportunity to undertake this research under their guidance. Their insightful feedback, thoughtful direction, and constant encouragement have been pivotal in navigating the challenges of this project and enriching our academic journey.

1. Introduction

This research explores the application of XGBoost, a powerful machine learning algorithm, for detecting and mitigating Distributed Denial of Service (DDoS) attacks. As digital infrastructure becomes increasingly critical to organizations worldwide, Distributed Denial of Service (DDoS) attacks have become a significant cybersecurity threat, as they can severely disrupt operations and lead to considerable financial losses and reputational damage. This report examines how XGBoost can provide superior detection capabilities compared to traditional security measures and other machine learning approaches, offering a more adaptive, efficient, and accurate defense against these evolving threats. [10].

1.1 Background

In recent years, Distributed Denial of Service (DDoS) attacks have become a major cybersecurity threat, creating significant challenges for organizations across various industries. Unlike traditional Denial of Service (DoS) attacks which originate from a single source, DDoS attacks harness multiple compromised devices, often organized into botnets, to flood targeted servers, services, or networks with overwhelming volumes of traffic. This distributed approach makes these attacks particularly difficult to mitigate as they can utilize hundreds of thousands or more attacking sources simultaneously, making it impossible to stop by simply blocking individual IP addresses.

The primary objective of such attacks is to render online services inaccessible to legitimate users by depleting bandwidth, overwhelming server resources, or exhausting network capabilities. As technology continues to evolve, these attacks have become increasingly sophisticated, with perpetrators employing various techniques targeting different layers of network infrastructure, from application-level attacks to volumetric bandwidth consumption.

The consequences of successful DDoS attacks extend beyond immediate service disruption, often resulting in significant financial losses, operational damage, and long-term reputational harm to affected organizations. According to recent research, DDoS attack volume has remained consistently high, with some of the most prominent global companies falling victim despite their substantial security resources. Traditional security measures such as firewalls and conventional intrusion detection systems, while valuable for general security, often prove inadequate against the scale and complexity of modern DDoS attacks, highlighting the need for more advanced detection and mitigation strategies. [6].

1.2 Motivation

The limitations of traditional security approaches have driven significant interest in machine learning-based solutions for DDoS attack detection and mitigation. Conventional methods often struggle with accurately distinguishing malicious traffic from legitimate network activities, particularly when attackers employ techniques designed to mimic normal user behavior or when attack patterns evolve beyond predefined signatures. Machine learning algorithms offer a promising alternative by enabling systems to learn and recognize patterns in network traffic data, identifying anomalies that may indicate attack scenarios without relying solely on static rules or signatures.

The research community has increasingly recognized a significant gap in effectively detecting and mitigating

DDoS attacks, particularly in emerging network environments such as Software-Defined Networks (SDN), Internet of Things (IoT), and cloud infrastructures [2]. These modern network architectures present unique security challenges due to their distributed nature, programmability, and resource constraints, making them particularly vulnerable to sophisticated DDoS attacks.

1.3 Objectives

The primary objective of this research is to develop and evaluate an XGBoost-based approach for detecting and mitigating DDoS attacks with superior accuracy and efficiency compared to traditional methods. XGBoost (Extreme Gradient Boosting) represents a powerful advancement in machine learning technology, operating by constructing an ensemble of decision trees in a sequential manner where each new tree focuses on correcting the errors of its predecessors.

For DDoS detection specifically, the objective is to leverage XGBoost's ability to capture complex patterns and interactions between network traffic features that may indicate malicious activity, thereby enhancing detection precision while minimizing false positives. The research aims to exploit XGBoost's capacity to handle imbalanced datasets—a common challenge in network security where attack instances are typically rare compared to normal traffic—through techniques such as weighted loss functions and appropriate evaluation metrics.

Furthermore, this study seeks to optimize XGBoost's performance specifically for DDoS detection by identifying the most relevant features from network traffic data that contribute significantly to detection accuracy. By implementing dimensional reduction and feature selection techniques, the research aims to enhance both the efficiency and effectiveness of the detection model, potentially reducing computational overhead while maintaining or improving detection capabilities.

Another key objective involves comparing XGBoost's performance against other machine learning algorithms such as Logistic Regression, Naive Bayes, Decision Trees to establish a comprehensive benchmark of its effectiveness in the specific context of DDoS attack detection.

Recent experimental findings have shown that XGBoost can achieve remarkably high accuracy rates of up to 99.9% in identifying DDoS attacks while significantly reducing the total number of features required for effective detection. Through this systematic evaluation and optimization of XGBoost for DDoS detection, the research ultimately aims to contribute to the development of more resilient network security infrastructures capable of defending against increasingly sophisticated cyber threats in various network environments.

2. Related Work

2.1 Statistical-Based Approaches

Statistical-based DDoS detection methods analyze network traffic properties like packet rates, flow distributions, and entropy to identify anomalies. One approach, proposed by Xiao et al., uses correlation analysis to enhance detection while reducing computational overhead. Their r-polling method minimizes training data without compromising effectiveness. However, statistical models struggle with evolving attack patterns and often generate false positives in dynamic networks.

Entropy-based detection measures randomness in traffic attributes, identifying shifts during attacks. While effective for volumetric attacks, it struggles against stealthy threats and requires extensive baseline analysis, making it challenging in dynamic environments like SDN-based clouds.

2.2 Machine Learning Techniques

Machine learning has gained traction in DDoS detection due to its ability to learn complex traffic patterns and adapt to evolving threats. These methods fall into two main categories: traditional machine learning algorithms and ensemble techniques, each with distinct strengths and trade-offs.

Traditional methods like CKNN (KNN with correlation analysis) improve detection accuracy but struggle with scalability in high-throughput networks due to computational overhead. SVM, when combined with Self-Organized Ant Colony Networks (CSOACNs), enhances classification performance but requires careful feature selection and has longer training times (33.23s vs. 11.07s for XGBoost, per Chen et al.) [1]). ANNs, used by Saied et al., effectively model non-linear traffic patterns but demand significant training data and computational resources, making them less suitable for resource-limited environments.

Ensemble methods provide better robustness. Random Forest offers fast training (3.59s) but has a higher false positive rate (0.018). GBDT refines predictions by sequentially correcting errors, achieving higher accuracy (97.69%) at the cost of longer training (21.67s). These machine learning approaches present trade-offs between accuracy, false positives, and computational efficiency, with the optimal choice depending on deployment constraints and security priorities.

2.3 XGBoost Related Work

Chen et al.'s XGBoost Implementation for SDN-Based Cloud Environment. Chen et al. [9] proposed an XGBoost-based approach for DDoS detection in SDN cloud environments, focusing on protecting the SDN controller. Using POX for SDN control and Mininet for net-

work simulation, they created a realistic attack scenario where one cloud network targeted another. They selected nine key features from the KDD Cup 1999 dataset using information gain and chi-square statistics, preprocessing them for optimal model performance.

Leveraging XGBoost's parallel computation via OpenMP and its DMatrix class for preprocessing, their approach enhanced efficiency for real-time detection. Compared to GBDT, Random Forest, and SVM, XGBoost achieved the highest accuracy (98.53%) and the lowest false positive rate (0.008). It also demonstrated strong scalability, maintaining efficiency even as dataset size increased from 400,000 to 4 million entries, with detection time improving through parallelization.

Despite these impressive results, Chen et al.'s approach has several limitations. First, their reliance on the KDD Cup 1999 dataset, while widely used, raises concerns about the model's effectiveness against modern, more sophisticated attack patterns that have evolved significantly since this dataset was created. Second, their evaluation focused primarily on traditional flooding-based attacks (ICMP flooding, TCP flooding, TCP-SYN flooding, UDP flooding, and Smurf attacks), potentially limiting effectiveness against emerging low-rate or application-layer DDoS attacks that mimic legitimate traffic. Finally, the simulation-based evaluation, while methodologically sound, may not fully represent the diversity and complexity of traffic patterns in production environments, particularly those involving encrypted communications or sophisticated evasion techniques.

3. Discussion

3.1 DDoS Attacks

A Distributed Denial of Service (DDoS) attack is a malicious attempt to make an online service or network unavailable by overwhelming it with massive amounts of traffic from multiple sources. DDoS attacks target system resources, causing disruption or complete shutdown, preventing legitimate users from accessing services.

SYN Flood attacks represent one of the most widespread and severe forms of Distributed Denial of Service (DDoS) attacks. In such an attack, the adversary manipulates the TCP handshake protocol by transmitting a large volume of SYN packets to the target server without completing the three-way handshake. This causes the server to maintain a large number of half-open connections, ultimately draining its resources and hindering its ability to respond to legitimate traffic.

DDoS attacks can be categorized mainly as IP spoofing (impersonating trusted sources) and flooding attacks (sending excessive packets to overload the target). To effectively detect and prevent such attacks, Machine Learning (ML) techniques are used to classify network traffic as either normal (benign) or malicious (attack). In this study, we focus specifically on SYN attacks, using ML-based methods like XGBoost to analyze and classify traffic patterns, aiming for high accuracy in identifying and mitigating SYN Flood DDoS attacks.

3.2 XGBoost Classification Algorithm

XGBoost is a powerful machine learning algorithm based on gradient boosting, designed for speed, efficiency, and high predictive performance.

It is an ensemble learning method that builds multiple weak learners (decision trees) sequentially, where each new tree corrects the mistakes of the previous trees. It optimizes performance using gradient boosting, which minimizes the errors in predictions by adjusting model weights using gradient descent.

XGBoost uses gradient boosting, where trees are trained in sequence. Each new tree attempts to correct the errors of the previous trees by focusing more on misclassified data points. It automatically learns the best way to handle missing values. It supports weighted sampling to balance datasets during training.

Unlike traditional gradient boosting, XGBoost includes L1 (Lasso) and L2 (Ridge) regularization, preventing overfitting and improving generalization [3].

This is the objective function and optimization method of the XGBoost algorithm respectively. The objective function of XGBoost is defined as:

$$F_{\text{Obj}}(\theta) = L(\theta) + \Omega(\theta)$$

where

$$L(\theta) = l(y_i, y_i)$$

and

$$\Omega(\theta) = \gamma T + \frac{1}{2}\lambda ||w||^2$$

The objective function consists of two components: $L(\theta)$ and $\Omega(\theta)$, where θ denotes the model parameters. The term $L(\theta)$ is a smooth and convex loss function that quantifies the difference between the predicted value y_i and the actual target y_i, indicating how well the model fits the data.

However, the objective function defined in Equation (1) presents difficulties for traditional optimization methods due to the presence of regularization terms that involve functions as parameters. Consequently, it becomes essential to determine whether the target y_i can be learned using the formulation provided in Equation (2).

$$L(\theta) = \sum_{i=1}^{n} l\left(y_i, y_i^{(t-1)} + S_t(T_i)\right) + \Omega(\theta)$$

Equation (2) provides an optimal solution primarily when applied to the squared loss function. However, extending it to other loss functions introduces substantial complexity. To overcome this challenge, a second-order Taylor expansion is used to reformulate Equation (2) into Equation (3), thereby allowing the method to be applied to a wider variety of loss functions.

$$L(\theta) = \sum_{i=1}^{n} \left(l\left(y_i, \hat{y}^{(t-1)}\right) + g_i S_t(T_i) + \frac{1}{2} h_i S_t^2(T_i) \right) + \Omega(\theta)$$

4. Data Collection

To develop an effective DDoS attack detection model, we sourced real-world network traffic data from the **CICDDoS2019 dataset**, an open-access dataset specifically designed for DDoS attack research. Our primary focus is on **SYN flood attacks**, a type of TCP-based DDoS attack that overwhelms the target by sending numerous SYN requests, exhausting server resources.

The dataset contained traffic from both malicious (**SYN attacks**) and benign (**normal network activity**) sources, allowing us to train an ML model capable of distinguishing between attack and legitimate traffic.

The CIC DDoS dataset comprises two subsets collected on different days:

- **CICDDoS-Training Dataset** – Used to train the XGBoost model by learning attack patterns from labeled network traffic.
- **CICDDoS-Testing Dataset** – Used to evaluate the model's performance on unseen network traffic recorded on a different day.

5. Data Preprocessing

To ensure the dataset was structured and optimized for training, we performed several preprocessing steps using Python's `pandas` library.

5.1 Dataset Balancing

We balanced the dataset by ensuring a **9:1 ratio** between attack and benign traffic. We identified benign samples (with the label "Benign") and attack samples (all other labels) and then downsampled the attack samples to match the required proportion. After balancing, we shuffled the dataset to remove any inherent ordering bias.

5.2 Data Cleaning

We cleaned the data by:

- Removing duplicate records.
- Dropping columns that contained only missing values.
- Converting numeric columns that were mistakenly stored as strings into their proper numerical format.
- Replacing any infinite values with NaN and removing missing values.

These steps ensured the dataset was well-prepared for training an accurate and efficient XGBoost model.

6. Feature Selection

Feature selection is a crucial step in optimizing the performance of our XGBoost model for DDoS attack detection. It helps in reducing dimensionality, improving model efficiency, and enhancing interpretability.

6.1 Feature Importance

The most influential features were identified and visualized using a bar chart, allowing us to understand which network parameters play a key role in distinguishing SYN attacks from benign traffic.

We used the following XGBoost method to retrieve the feature importance:

```
initial_xgb.get_booster().get_score(
importance_type="weight"
)
```

The `importance_type="weight"` argument specifies that feature importance is measured by the number of times a feature is used in splits across all trees in the model. This returns a dictionary where:

- **Keys** = feature names (e.g., 'Flow Bytes/s', 'Fwd Packets/s').
- **Values** = importance scores (e.g., 15, 23, indicating how many times the feature was used in splits).

6.2 Selected Features

The following features were selected based on their importance scores:

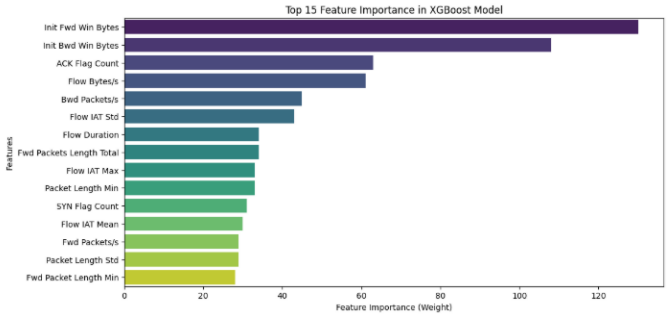

Figure 1: Selected Features based on Importance Scores

7. Hyperparameter Tuning

Hyperparameter tuning refers to the process of modifying the configuration settings that influence the learning behavior of a machine learning model. Unlike model parameters, which are derived during training, hyperparameters need to be defined prior to the training phase. Proper selection of these hyperparameters is vital for achieving high model performance.

7.1 Important Hyperparameters in XGBoost

XGBoost has several important hyperparameters that influence its performance:

- **learning_rate**: Controls the step size taken by the optimizer.
- **n_estimators**: Determines the number of boosting trees.
- **max_depth**: Sets the maximum depth of each tree.
- **subsample**: Defines the percentage of rows used for each tree construction.
- **colsample_bytree**: Determines the fraction of features used per tree.

7.2 Grid Search for Hyperparameter Tuning

To find the best combination of hyperparameters, we used `GridSearchCV`, which evaluates all possible combinations of hyperparameters and selects the best-performing set.

7.3 Best Hyperparameters

After performing `GridSearchCV`, the optimal hyperparameters were identified as:

- **colsample_bytree**: 0.8
- **learning_rate**: 0.2
- **max_depth**: 3
- **n_estimators**: 100
- **subsample**: 0.8

These values were used for training the final model in subsequent steps.

8. Accuracy and Results

We evaluated our model using two different approaches to assess its performance.

In the first approach, we split the original training dataset into 80% training and 20% testing. The model was trained on the 80% subset and then evaluated on the remaining 20%. This yielded an impressive accuracy of 99.95%, with a ROC-AUC score of 0.99999, indicating a highly effective classification capability. The confusion matrix showed minimal misclassifications, reinforcing the model's robustness in distinguishing between the two classes.

8.1 Results on 20% Test Split

- **Accuracy:** 0.9995
- **ROC-AUC:** 0.99999

Table 1: Classification Report (20% Test Split)

Class	Precision	Recall	F1-Score	Support
0 (Benign)	1.00	1.00	1.00	5407
1 (Attack)	1.00	1.00	1.00	8661
Accuracy			**1.00**	**14068**

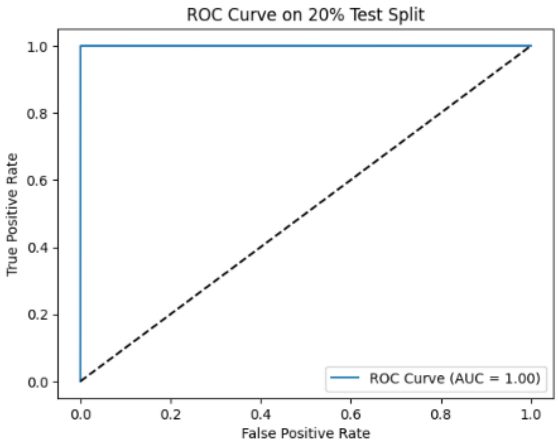

Figure 2: ROC-AUC curve for 20% split

8.2 Results on Testing Dataset

In the second approach, we trained the model on the full training dataset and then tested it on an entirely separate testing dataset, `syn-testing.csv`. This evaluation resulted in an accuracy of 99.78% and a ROC-AUC score of 0.9986, demonstrating the model's strong generalization to unseen data. The classification report confirmed high precision and recall, with the confusion matrix showing only a few false positives and negligible false negatives. These results highlight the model's reliability in real-world scenarios.

- **Accuracy:** 0.9978
- **ROC-AUC:** 0.9986

Table 2: Classification Report (Testing Dataset)

Class	Precision	Recall	F1-Score	Support
0 (Benign)	1.00	0.99	1.00	374
1 (Attack)	1.00	1.00	1.00	533
Accuracy			**1.00**	**907**

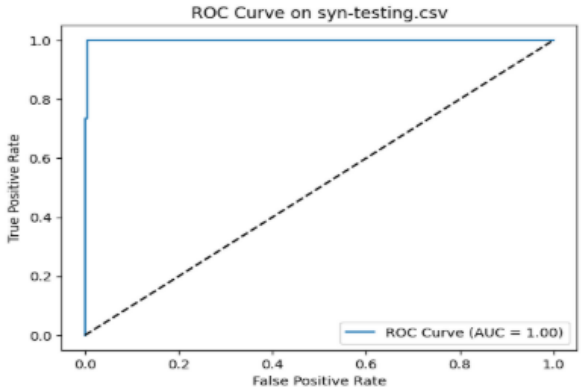

Figure 3: ROC-AUC curve for testing dataset

9. Evaluation Using BCCC-Mal-NetMem-2025 Dataset

To further validate the robustness and practical applicability of the DDoS detection model, a second dataset, BCCC-Mal-NetMem-2025 `cleaned_dataset2.csv`, was introduced. This dataset was derived from raw EV charging station traffic, cleaned, and engineered to represent realistic patterns of benign and SYN-based DDoS attack traffic. Its inclusion was critical for evaluating the generalizability, stability, and operational reliability of the detection system beyond the controlled training environment.

9.1 Feature Selection-Based Training (60-25-15 Split)

The dataset was partitioned into three parts:

- 60% for model training
- 25% (sampled from the remaining 40%) for feature selection
- 15% reserved as a final holdout test set

An initial XGBoost model was trained on the 85% (training + feature selection) subset to determine feature importance using the `weight` metric. The top 15 features were selected and used to train the final model, which was then evaluated on the 15% holdout set.

Performance on the 15% holdout test set:

- Accuracy: 99.93%
- ROC-AUC Score: 0.9999
- SYN Recall: >99.8% across multiple thresholds

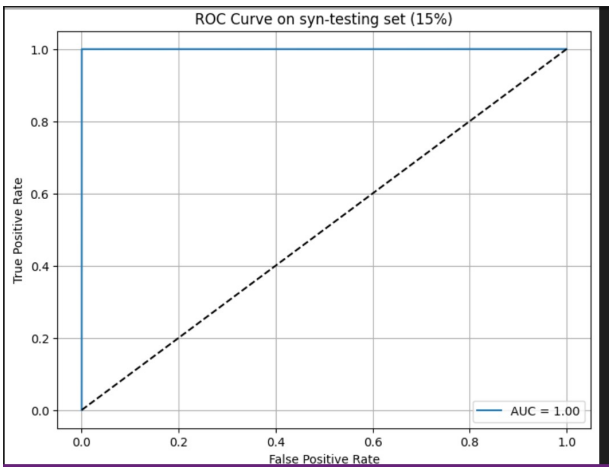

Figure 4: ROC-AUC curve for 15% holdout test set dataset

9.2 Threshold Tuning

Given the need to minimize false negatives in SYN attack detection, the model's decision threshold was varied between 0.1 and 0.5. Lower thresholds (e.g., 0.2) yielded nearly perfect SYN recall, enabling trade-offs between precision and sensitivity based on operational priorities.

9.3 Stratified K-Fold Cross-Validation (K=5)

To further test generalization, 5-fold stratified cross-validation was performed. This approach ensured each fold preserved the original class distribution while providing a complete evaluation across all samples.

Table 3: 5-Fold Cross-Validation Results

Fold	Accuracy	ROC-AUC	SYN Recall
1	0.9997	1.0000	0.9998
2	0.9998	1.0000	0.9999
3	0.9998	1.0000	0.9998
4	0.9998	1.0000	0.9999
5	0.9996	1.0000	0.9999
Average	**0.9998**	**1.0000**	**0.9999**

9.4 Feature Importance Analysis

A bar chart was generated using Seaborn to visualize the top 15 most influential features. Metrics related to flow size, timing variance, and skew were observed to be the most informative in distinguishing between benign and malicious traffic, confirming the model's interpretability and data-driven structure.

9.5 Conclusion

The introduction of the second dataset played a pivotal role in confirming the XGBoost model's adaptability, precision, and generalization. Unlike synthetic datasets, this dataset reflected real-world traffic behavior and potential noise. Through diverse evaluation methods—feature-selected training, threshold tuning, and cross-validation—the model demonstrated state-of-the-art DDoS detection performance, with high SYN recall and ROC-AUC. These findings confirm the model's readiness for deployment in real-time cybersecurity systems.

Table 4: Holdout Test Set Performance Comparison

Metric	**Dataset 1**	**Dataset 2**
Data Source	CICDDoS2019	BCCC-Mal-NetMem-2025
Accuracy (holdout test set)	99.89%	99.93%
ROC-AUC (holdout)	0.9997	0.9999
SYN Recall (holdout)	0.9989	0.9998
Evaluation Strategy	80/20 train-test split	60-25-15 split (train-feature-test)
Cross-Validation	Not performed	5-Fold Stratified Cross-Validation

References

[1] Chen et al., "Network Intrusion Detection with XGBoost," Distributed Systems Group.

[2] Alqahtani, H.M. and Abdullah, M., "A Review on DDoS Attacks Classifying and Detection by ML/DL Models," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 2, 2024.

[3] "XGBoost Machine Learning Model-Based DDoS Attack Detection," *International Journal of Engineering Trends and Technology*, vol. 71, no. 2, 2023.

[4] Sahosh, Z.H., Faheem, A., Tuba, M.B., Ahmed, M.I., and Tasnim, S.A., "A Comparative Review on DDoS Attack Detection Using Machine Learning Techniques," *Malaysian Journal of Science and Advanced Technology*, vol. 4, no. 2, pp. 75–83, March 2024.

[5] "XGBoost Machine Learning Model-Based DDoS Attack Detection," *International Journal of Engineering Trends and Technology*, February 2023.

[6] "Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review," *Applied Sciences*, vol. 13, no. 5, 2023.

[7] Dhahir, Z.S., "A Hybrid Approach for Efficient DDoS Detection in Network Traffic Using CBLOF-Based Feature Engineering and XGBoost," *Future Technologies and Scientific Innovations*, 2024.

[8] "An Efficient Real Time DDoS Detection Model Using Machine Learning Algorithms," *arXiv*, 2019.

[9] Chen, Z., Jiang, F., Cheng, Y., Gu, X., Liu, W., & Peng, J. (2018). "XGBoost Classifier for DDoS Attack Detection and Analysis in SDN-based Cloud," in *IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 273–276.

[10] Yan, Q., Yu, F.R., Gong, Q., & Li, J. (2016). "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 602–622.

[11] Wang, W., & Gombault, S. (2009). "Efficient Detection of DDoS Attacks with Important Attributes," in *International Conference on Risks and Security of Internet and Systems*, pp. 61–67.

[12] Feng, W., Zhang, Q., Hu, G., & Huang, J. (2014). "Mining Network Data for Intrusion Detection through Combining SVMs with Ant Colony Networks," *Future Generation Computer Systems*, vol. 37, pp. 127–140.