Distributive Denial of Service Attack Detection Using Cat: Boost Algorithm in Machine Learning

Parvathi M., Dharanidharan A., Kirubasankar K. R.,
Mohamaad Ishaaq S. and Navaneetha Krishnan S.
Department of AI&DS, Nandha Engineering College, Erode, Tamil Nadu, India

Keywords: Distributed Denial of Service (DDOS) Attacks, Machine Learning, Cat-Boost Algorithm, DDoS Attack Detection.

Abstract: DDOS (Distributive denial of service) attacks is worked by overloading the server with multiple connections at the same time, making the site unable to take any more requests which in turn causes overloading of the server, triggering response time error's. This work is done on ubuntu operating system with tools such as mininet installed inside virtual machine and controller which helps by acting as a framework of the virtual machine which specializes in detection of the attack. The current existing system is done on the algorithm named as "random forest" (K. G. Maheswari, et.al., 2023). which utilities the concept of decision trees whereas in this project the current algorithm is replaced with a different one to operate which is "cat-boost". It works on the principles of categorizing the data's by assigning values of the same class together.

1 INTRODUCTION

Network security is seriously threatened by distributed denial of service (DDoS) attacks which overload servers with so many connections at once that they are unable to process valid requests in the end this leads to issues with response times and server overload which reduce service availability the project intends to improve the DDoS assault detection system.(K. G. Maheswari, et.al., 2023) In order to solve this issue to detect fraudulent traffic the current system uses the random forest approach which makes use of decision trees this study suggests using a more sophisticated algorithm called cat-boost to increase detection efficiency and accuracy by grouping values of the same class together cat-boost enhances model performance and works well with categorical data it accomplishes this by applying the gradient boosting approach (Y. Yu, et.al., 2018). DDoS attack detection needs to installs technologies such as the Ryu controller and mininet onto pcs running ubuntu in order to simulate network (H. S. Abdulkarem and A. Drawod, 2020). circumstances instead of using random forest to identify attacks the ryu controller framework uses catboost this study intends to improve network security and attack mitigation strategies in addition to providing a more dependable and efficient DDoS detection system (Y. Xiang, et.al., 2011).

1.1 Technological Achievement

In their early iterations traditional DDoS detection methods mostly depended on rule-based systems and basic statistical analysis which regularly fell behind the increasing complexity and reach of attacks the introduction of machine learning algorithms such as random forest was a significant breakthrough enabling decision trees to recognize and decide on increasingly complex patterns however these methods were still limited in their capacity to manage category data and scale for large dynamic networks(K. G. Maheswari, et.al., 2023).It includes some Modern Innovations The CatBoost approach, a cutting- edge machine learning technique that is excellent at handling categorical data and improving classification accuracy, enables a more precise identification of fraud traffic by efficiently combining and evaluating data of the same type. This has led to considerable advancements in the detection of DDoS attacks (K. G. Maheswari, et.al., 2023). The random forest approach, which uses decision trees, is a more traditional approach that might not be able to handle complicated data-patterns (J. Ye, et.al., 2018). And it has some limitations like its effectiveness, the

M., P., A., D., R., K. K., S., M. I. and S., N. K.
Distributive Denial of Service Attack Detection Using Cat: Boost Algorithm in Machine Learning.
DOI: 10.5220/0013900600004919

In *Proceedings of the 1st International Conference on Research and Development in Information, Communication, and Computing Technologies (ICRDICCT'25 2025) - Volume 3*, pages 496-503
ISBN: 978-989-758-777-1

suggested approach has a number of drawbacks that must be noted and we can make the Controlled Environment of DDoS detection using Mininet and the Ryu controller in an Ubuntu virtual machine, the system is put into practice and evaluated in a simulated setting (Y. Yu, L. et.al., 2018). The intricacies and dynamism of actual network settings could not be adequately captured by this controlled arrangement, which could restrict how broadly the findings can be applied (N. M. Saravana Kumar, et.al., 2016).

Computational Resources has effectiveness with categorical data, the Cat-Boost method may necessitate a large amount of computing power and a great deal of hyper- parameter adjustment. Large-scale deployments or situations with limited resource provide difficulties (K. G. Maheswari, et.al., 2023). Some Dataset Dependency need to add to get better performance of the system is highly dependent on the quality and diversity of the training dataset. If the dataset lacks sufficient representation of various types of DDoS attacks, the model's detection accuracy may be adversely affected (N. Zhang, et.al., 2019). Also, it includes some Lack of Mitigation Strategies this is because of the current implementation focuses solely on the detection of DDoS attacks and does not address the mitigation of detected attacks. A comprehensive solution would require integrating mitigation strategies to effectively neutralize threats (W. Zhijun, et.sl., 2020). It also focusses on the Scalability Concerns to achieve system's scalability to larger and more complex networks has not been thoroughly tested. Further research is needed to evaluate its performance in high- traffic and heterogeneous network environments (S. Dong, et.al., 2019). We can use this system for Real-Time Detection by improving the system's ability to detect DDoS attacks in real-time has not been extensively evaluated. Real- world deployment would require ensuring minimal latency and high accuracy in real-time detection scenarios (N. Agrawal and S. Tapaswi, et.al., 2019).

1.2 Impact on Customer Experience and Efficiency

Impact on customer satisfaction and productivity the proposed technique for detecting distributed denial of service has a substantial influence on both operational efficiency and user experience. The system ensures continuous service availability by preventing server overload, which reduces downtime and disruptions that could otherwise annoy users. This directly improves customer satisfaction as users can rely on services without experiencing delays or outages. DDoS attacks are successfully detected and mitigated by the cat- boost algorithm, which also reduces latency and maintains optimal response times. Because customers can depend on the platform for consistent performance, this is particularly crucial for time- sensitive applications like financial services and e-commerce games. Increased reliability of the network infrastructure fosters client confidence and loyalty (Y. Xiang, t.al., 2011).

DDOS detection automation operationally removes the need for human intervention by establishing a safe, efficient, and customer-Caused work environment (M. Baskar, et.al., 2013). This allows employees to focus on other crucial tasks and more efficiently allocate resources. The solution also lowers the costs related to human monitoring and response, increasing operational efficiency. Enhances organizational performance and user experience, and the system's proactive feature makes early threat detection possible. Last but not least, its scalability allows it to adjust to shifting network requirements and sustain peak performance even as traffic levels rise, reducing harm and guaranteeing business continuity (K. G. Maheswari,et.al , 2023).

2 EXISTING SYSTEM

The Random Forest method, which uses the idea of decision trees to categorize and identify harmful traffic patterns, is the foundation of the current system for detecting Distributed Denial of Service (DDoS) assaults. In order to increase accuracy and decrease overfitting, this method entails building many decision trees during training and combining their output. Utilizing technologies like Mininet, which mimics network settings, and the Ryu controller, which serves as a framework for controlling the virtual machine and identifying threats, the system is deployed on the Ubuntu operating system. In order to avoid server overload brought on by too many concurrent connections, the Random Forest algorithm analyzes network traffic data to separate malicious from legal requests (O. Osanaiye, et.al., 2016).

3 PROPOSED SYSTEM

The system that is being suggested by using the CatBoost algorithm, which is an advancement over the conventional Random Forest-based detection technique, the suggested system seeks to improve the detection of Distributed Denial of Service (DDoS) assaults (M. H. Bhuyan and E. Elmroth, 2018). The Ryu controller acts as the SDN (Software-Defined Networking) framework, and the system is set up on an Ubuntu- based environment utilizing Mininet within a virtual machine. By effectively managing categorical data and enhancing decision- making in real-time assault scenarios, the gradient boosting algorithm CatBoost improves detection accuracy. To more accurately categorize and detect malicious activity, the model is trained using network traffic data. CatBoost exhibits better flexibility to a variety of attack patterns and lowers false positive rates when compared to current Random Forest techniques (Saied, et.al, 2016).

3.1 Packets

Table 1: D/B random forest and CatBoost.

RANDOM FOREST	CATBOOST
Random Forest is a bagging algorithm that combines decision trees trained on different subsets of data)	CatBoost is a boosting algorithm (sequentially builds trees, where each tree corrects the errors of the previous one)
Random Forest requires manual encoding which has to be processed by the user onto the algorithm while training	Cat-Boost automatically handles categorical feature without requiring preprocessing.
Random Forest grows trees independently and in parallel that can be used for decision making	CatBoost grows trees sequentially, with each tree focusing on the mistakes of the previous one
Random Forest is less prone to overfitting due to averaging predictions from multiple trees.	CatBoost is more prone to overfitting but includes regularization techniques to mitigate it.
Random Forest is generally faster to train because trees are built independently.	CatBoost is slower to train due to its sequential nature and handling of categorical features

Table 1 shows the D/B Random Forest and Catboost. The program checks a total of 44 various features before confirming the status of the network The 44 different features are namely: 'Protocol', 'Flow Duration', 'Total Fwd Packets', 'Total Backward Packets', 'Fwd Packets Length Total', 'Bwd Packets Length Total', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean', 'Fwd Packet Length Std', 'Bwd Packet Length Max', 'Bwd Packet Length Min', 'Bwd Packet Length Mean', 'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max', 'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags', 'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s', 'Packet Length Min', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance', 'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count', 'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio', 'Avg Packet Size', 'Avg Fwd Segment Size', 'Avg Bwd Segment Size', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk', 'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate' (Z. Li, et.al., 2020).

4 METHODOLOGY

There are three techniques the current system uses a random forest approach to create a large number of decisions during the training phase and increase results, increase prediction accuracy existing random forests do not work well in dynamic network environments with complex traffic patterns and high data collections even if it is successful the gradient boost technology is based on catboost created this way beyond traditional methods in relation prediction accuracy with minimal preparation. And in this methodology, we are using this catboost technology in windows. As traditional Techniques and other Algorithms mainly works on the Ubuntu and Linus. But by using this methodology we can work the catboost algorithm in both Windows and Linus Operating System. By using the Catboost Algorithm it achieves the success rate of 99 percentage on both windows and Linus (M. H. Bhuyan,et.al., 2014). Figure 1 shows the bar chart of cat-boost and random forest comparison.

Mathematically, CatBoost can be represented as follows:

Given a training dataset with N samples and M features, where each sample is denoted as (x_i, y_i), as x_i is a vector of M features and y_i is the corresponding target variable, CatBoost aims to learn a function F(x) that predicts the target variable y.

$$F(x) = F0(x) + \sum m = 1M \sum i = 1Nfm(xi)F(x) = F0(x) + \sum m = 1M \sum i = 1Nfm \qquad (1)$$

where, F(x) represents the overall prediction function that CatBoost aims to learn. It takes an input vector x and predicts the corresponding target variable y.

F0() F(x) is the initial guess or the baseline prediction. It is often set as the mean of the target variable in the training dataset. This term captures the overall average behavior to the target variable. Σm=1MΣm=1M represents the summation over the ensemble of trees. M denotes the total number of trees in the ensemble. Σi=1 NΣi=1N. Represents summation over training samples. N denotes the total number of training samples.fm (xi) fm (xi) represents the prediction of the m-th tree for the i-th training sample. Each tree in the ensemble contributes to the overall prediction by making its own prediction for each training sample (O. Osanaiye, et.al., 2016).

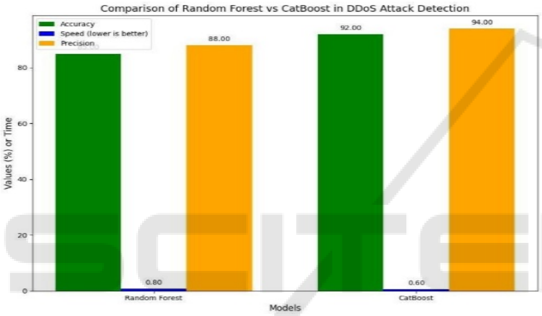


Figure 1: Bar Chart of CatBoost and random forest comparison.

Table 2: Tools used in this system.

Tool	Purpose
Ubuntu (OS)	Provides a secure environment for execution.
Mininet	Simulates a virtual network for testing.
Ryu Controller	Acts as an SDN controller for traffic monitoring.
Virtual Machine	Hosts the network simulation setup.
CatBoost Algorithm	Implements machine learning for attack detection.
Python	Used for programming and model implementation.
Windows (OS)	An Operating System used for testing of the proposed system.

4.1 Tools

The following tools (table 2) are used for implementing the DDoS attack detection system:
These tools are essential for simulating and detecting DDoS attacks effectively. Kali-Linux, Windows, RyuController, CatBoost Algorithm and Python. Where this tool is used to detect DDoS and helps and each working property are explanations are Ubuntu is an operating system based on Linux that may be used to execute the machine learning model and simulation in a secure and reliable setting. It is the perfect solution for our project because it is open-source and works with different networking technologies. And mininet is a network emulator made to simulate a network and assess how well it works. It facilitates the modeling of extensive network topologies and aids in the controlled study of DDoS attack behavior. The main tool for the DDoS detections is Ryu Controller it is a software-defined networking (SDN) controller controls network traffic and makes it easier to identify attacks in real time. It offers an adaptable framework for creating unique network applications and putting security measures in place (J. Cui, et.al., 2019). The network simulation environment is housed in a virtual machine (VM), which makes it possible to run the Ryu controller, Ubuntu, and Mininet. This guarantees isolated and repeatable DDoS attack detection testing circumstances. And CatBoost is a gradient boosting technique designed for high- performance machine learning applications. It is used in this project to precisely classify network traffic and identify potential DDoS attacks. Its ability to handle categorical data well makes it suitable for intrusion detection systems. The language used for this is Python. It is the primary computer language utilized in the development of the detection model. To detect threats in real time, it analyzes data, trains the CatBoost algorithm, and communicates with the Ryu controller. And it also works on Windows. It is an Operating System. Here we can use this operating system for testing and implementation of the project (S. Dong,et.al., 2019).

4.2 Modules Descriptions

Mininet-Based Network Simulation is a virtual network environment is created using Mininet, a network emulation program. The system can examine the behavior of DDoS assaults in a controlled environment by simulating different network topologies and traffic patterns. Integration of Ryu Controller to control the Network traffic and to dynamically managed and monitored by the Ryu Software-Defined Networking (SDN) controller (T. A. Pascoal, et.al.,2017). It serves as a foundation for network packet capture and analysis inside the virtual machine, guaranteeing efficient communication and early identification of unusual traffic. And the main Process includes in data preprocessing and feature extraction is to extract key characteristics that differentiate legitimate traffic from possible DDoS assaults, network traffic data is gathered and evaluated. Preprocessing methods like data transformation and normalization are used to increase classification accuracy. Putting the Cat-Boost Algorithm into Practice and it is used in the suggested system in place of the Random Forest method. Cat-Boost is an effective gradient boosting method that improves classification accuracy for DDoS detection by classifying and processing data by clustering related class values. The best possible network performance (H. Kumawat and G. Meena, 2014).

4.3 Work Flow Chart

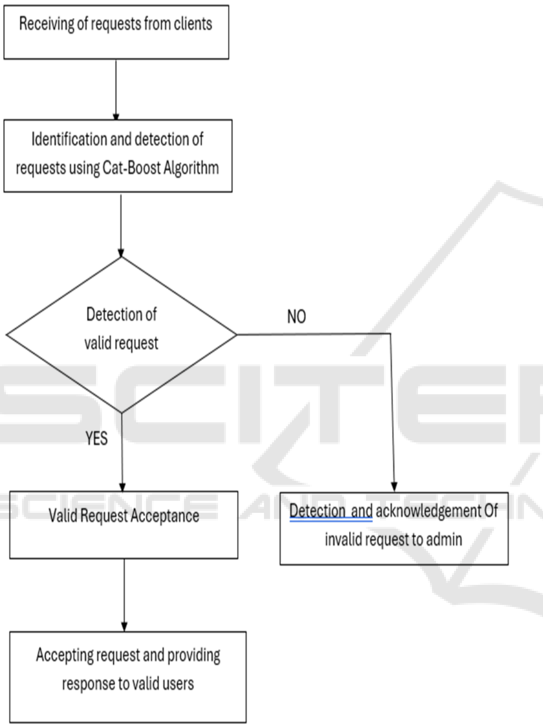
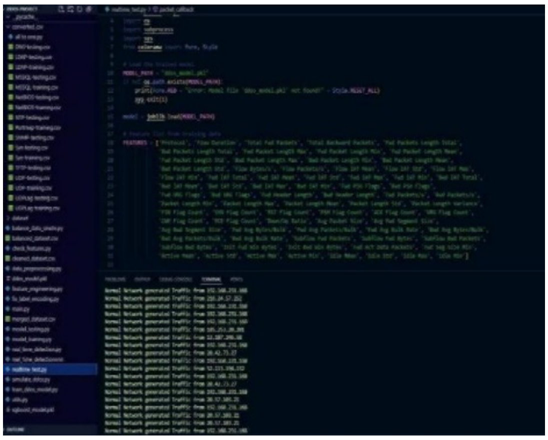

Receiving of requests from clients
Identification and detection of requests using Cat-Boost Algorithm
Detection of valid request
NO
YES
Valid Request Acceptance
Detection and acknowledgement Of invalid request to admin
Accepting request and providing response to valid users

Figure 2: Ddos Process Using Cat-Boost Algorithm.

Figure 2 shows the DDoS Process using cat-boost algorithm.

5 OUTCOMES

5.1 Screenshot of Output

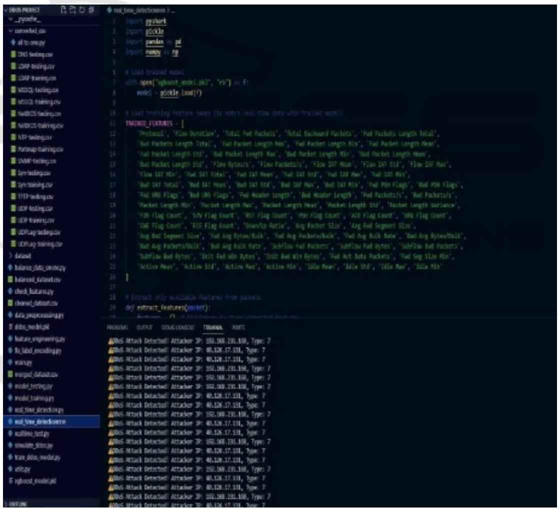


```
import os
import subprocess
import sys
from colorama import Fore, Style

# Load the trained model
MODEL_PATH = "ddos_model.pkl"
if not os.path.exists(MODEL_PATH):
    print(Fore.RED + "Error: Model file 'ddos_model.pkl' not found!" + Style.RESET_ALL)
    sys.exit(1)

model = joblib.load(MODEL_PATH)

# Feature list from training data
FEATURES = ['Protocol', 'Flow Duration', 'Total Fwd Packets', 'Total Backward Packets', 'Fwd Packets Length Total',
            'Bwd Packets Length Total', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean',
            'Fwd Packet Length Std', 'Bwd Packet Length Max', 'Bwd Packet Length Min', 'Bwd Packet Length Mean',
            'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
            'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min', 'Bwd IAT Total',
            'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags',
            'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
            'Packet Length Min', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance',
            'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count',
            'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio', 'Avg Packet Size', 'Avg Fwd Segment Size',
            'Avg Bwd Segment Size', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk',
            'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets',
            'Subflow Bwd Bytes', 'Init Fwd Win Bytes', 'Init Bwd Win Bytes', 'Fwd Act Data Packets', 'Fwd Seg Size Min',
            'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min']
```

```
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 210.24.57.152
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 185.251.38.88
Normal Network generated Traffic from 13.107.246.48
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 20.42.73.27
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 52.113.196.132
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 20.42.73.27
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 20.57.185.21
Normal Network generated Traffic from 192.168.231.168
Normal Network generated Traffic from 20.57.185.21
Normal Network generated Traffic from 20.57.185.21
Normal Network generated Traffic from 192.168.231.168
```

Figure 3: Output of Legal Request from Users.

5.2 Terminal



```
import pyshark
import pickle
import pandas as pd
import numpy as np

# Load trained model
with open("xgboost_model.pkl", "rb") as f:
    model = pickle.load(f)

# Load training feature names (to match real-time data with trained model)
TRAINED_FEATURES = [
    'Protocol', 'Flow Duration', 'Total Fwd Packets', 'Total Backward Packets', 'Fwd Packets Length Total',
    'Bwd Packets Length Total', 'Fwd Packet Length Max', 'Fwd Packet Length Min', 'Fwd Packet Length Mean',
    'Fwd Packet Length Std', 'Bwd Packet Length Max', 'Bwd Packet Length Min', 'Bwd Packet Length Mean',
    'Bwd Packet Length Std', 'Flow Bytes/s', 'Flow Packets/s', 'Flow IAT Mean', 'Flow IAT Std', 'Flow IAT Max',
    'Flow IAT Min', 'Fwd IAT Total', 'Fwd IAT Mean', 'Fwd IAT Std', 'Fwd IAT Max', 'Fwd IAT Min',
    'Bwd IAT Total', 'Bwd IAT Mean', 'Bwd IAT Std', 'Bwd IAT Max', 'Bwd IAT Min', 'Fwd PSH Flags', 'Bwd PSH Flags',
    'Fwd URG Flags', 'Bwd URG Flags', 'Fwd Header Length', 'Bwd Header Length', 'Fwd Packets/s', 'Bwd Packets/s',
    'Packet Length Min', 'Packet Length Max', 'Packet Length Mean', 'Packet Length Std', 'Packet Length Variance',
    'FIN Flag Count', 'SYN Flag Count', 'RST Flag Count', 'PSH Flag Count', 'ACK Flag Count', 'URG Flag Count',
    'CWE Flag Count', 'ECE Flag Count', 'Down/Up Ratio', 'Avg Packet Size', 'Avg Fwd Segment Size',
    'Avg Bwd Segment Size', 'Fwd Avg Bytes/Bulk', 'Fwd Avg Packets/Bulk', 'Fwd Avg Bulk Rate', 'Bwd Avg Bytes/Bulk',
    'Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate', 'Subflow Fwd Packets', 'Subflow Fwd Bytes', 'Subflow Bwd Packets',
    'Subflow Bwd Bytes', 'Init Fwd Win Bytes', 'Init Bwd Win Bytes', 'Fwd Act Data Packets', 'Fwd Seg Size Min',
    'Active Mean', 'Active Std', 'Active Max', 'Active Min', 'Idle Mean', 'Idle Std', 'Idle Max', 'Idle Min'
]

# Extract only available features from packets
def extract_features(packet):
```

```
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
⚠DDoS Attack Detected! Attacker IP: 192.168.231.158, Type: 7
⚠DDoS Attack Detected! Attacker IP: 40.126.17.131, Type: 7
```

Figure 4: Output of Request from Users With Ipaddress.

Using the Ryu controller and mininet implementation on an Ubuntu-based virtual machine, the proposed system successfully enhances classification accuracy and ddos attack detection by moving from the random forest method to the catboost algorithm. Attack scenarios can be tested and evaluated in a controlled simulation environment by handling categorical data more effectively. The catboost technique also reduces false positive rates and boosts model efficiency. Experiments show a notable improvement in detection speed, which lowers the error of server response times caused by

DDoS assaults. Network resilience is significantly increased by the architecture's ability to differentiate between malicious and legitimate traffic. Performance metrics like accuracy and f1-score recall demonstrate how trustworthy the recommended method is in spotting various DDoS attack patterns. In contrast to traditional random forest detection Cat boost offers greater scalability and adaptability to evolving attack tactics. The proposed model demonstrates how it might be used in real-world network security systems. Software-defined networking (SDN) in the Ryu controller provides dynamic and real-time response capabilities to successfully thwart attacks. Strengthening the defenses overall Future advancements might focus on optimizing feature selection and integrating real-time anomaly detection techniques for enhanced security (E. Adi, et.al., 2015). We analyze the performance of the DDoS attack detection method in SDN simulation network and evaluates the effectiveness and feasibility of the method in terms of accuracy, recall and misjudgment rate. In addition, we select the detection method based on joint and SOM-based machine learning detection algorithm as comparison (H. S. Abdulkarem and A. Drawod,2020). Most DDoS attack detection algorithms over SDN can be classified as methods based on entropy and methods based on machine learning. The comparison method JESS selected in this work is a DDoS detection method based on joint entropy (J. Ye, X. Cheng,et.al., 2018). This method not only considers the entropy of the target IP address, but also pays attention to the combination of IP address and TCP attributes, and selects the appropriate dynamic attributes for the current attack during the detection process. Although the detection method based on entropy (X. Liu, et.al., 2020).

6 APPLICATIONS

The application of this study is very relevant to network security, namely in thwarting Distributed Denial of Service (DDoS) attacks, which pose a severe risk to online services and it can also use infrastructure.

Using the Cat-Boost algorithm, this system offers a comprehensive way to detect DDoS attacks are characterized by overloading servers with concurrent connections, leading to response time problems and service unavailability. The implementation is carried out on the Ubuntu operating system using technologies such as Mininet and the Ryu controller in a virtual machine environment. The mininet facilitates network topology simulation, while the Ryu controller serves as a framework with an emphasis on attack detection. By classifying data according to class similarities, cat-boost substitutes the conventional random forest technique, improving attack detection speed and accuracy while increasing system efficiency. protecting web servers, cloud-based apps, and vital network infrastructure against DDoS attacks Two essential uses for this project are preserving peak performance and guaranteeing continuous service availability [15]. The development of resilient and intelligent cybersecurity systems has been significantly accelerated by the use of machine learning techniques, especially the cat-boost algorithm. By doing the following programming code. We get to determine the frequency of our network system whether its currently stable or is under any attack or is vunerable to attacks that may cause some anomalies on the network flow of its packets. It has a total of 44 different packets that needs to be checked each and individually. Even if one value mismatches it comes out as an anomaly in the network flow stream [24].

7 CHALLENGES AND FUTURE DIRECTIONS

The execution of this project ensures improved security for network systems by making a substantial progress in the detection of Distributed Denial of Service (DDoS) assaults. By substituting the Cat-Boost algorithm for the conventional Random Forest method, the system is able to identify fraudulent traffic with more accuracy. Because DDoS assaults can negatively affect availability and performance, this initiative is especially helpful for businesses that depend on cloud-based services, data centers, and web hosting platforms. Real-time monitoring and attack prevention are made possible by the scalable and effective network simulations made possible by the integration of Mininet within virtualized environment (Y. Yu, et.al., 2018). Future studies might concentrate on the endurance and scalability of the detection systems. The issue of acquiring labeled datasets might be resolved by using unsupervised or semi-supervised learning techniques. Utilizing distributed systems and edge computing for real- time detection might save computational overhead. Lastly, the deployment of sophisticated threat intelligence and anomaly detection systems might provide a more thorough defense against changing DDoS assault plans. A range of machine learning techniques, including ensemble approaches, could increase detection accuracy and decrease false positives. All of these enhancements would contribute to the development of more resilient and adaptable systems that can protect networks from more attackers (Saied, R. E. et.al., 2016).

8 CONCLUSIONS

The project's main goal is to identify Distributed Denial of Service (DDoS) assaults, which take advantage of server weaknesses by flooding them with connections at once, causing server overload and response time issues. Utilizing technologies like Mininet in a virtual machine environment and the Ryu controller, a framework with a focus on attack detection, the implementation was done on the Ubuntu operating system. This project presents the Cat-Boost method, which categorizes data by assigning values that belong to the same class, in contrast to the current system that uses the Random Forest algorithm based on decision trees. This paper offers a fresh technique to DDoS assault detection by substituting the CatBoost algorithm for the conventional Random Forest methodology (N. Zhang et.al., 2019). It is built with Mininet and the Ryu controller, which serves as an attack detection framework, in a virtualized Ubuntu environment. CatBoost improves detection accuracy over the traditional method by efficiently applying gradient boosting to decision trees for data categorization. This technique strengthens the network's security and resilience by enhancing the system's ability to distinguish between malicious and legal traffic (3. N. M. Saravana Kumar, et.al. 2016) The findings show that the suggested approach offers a more dependable and efficient way to guard against DDoS assaults, guarantee peak network performance, and lower the frequency of reaction time mistakes. Further optimization and practical implementation in future studies to enhance scalability and flexibility. The Figure 3 & 4 shows the output of the project where it has a process to identify the IP Address of every users who are requesting to the website and using this system we can identify the legal and illegal requests and its IP Address. And we can give the response to the legal requesting users.

REFERENCES

A. Saied, R. E. Overill, and T. Radzik, "Detection of known and unknown DDoS attacks using artificial neural networks," Neurocomputing, vol. 172, pp. 385–393, Jan. 2016.

E. Adi, Z. Baig, C. P. Lam, and P. Hingston, "Low-rate denial-of-service attacks against HTTP/2 services," in Proc. 5th Int. Conf. IT Convergence Security (ICITCS), Aug. 2015.

H. Kumawat and G. Meena, "Characterization, detection and mitigation of low-rate DoS attack," in Proc. Int. Conf. Inf. Commun. Technol. Competitive Strategies, 2014.

H. S. Abdulkarem and A. Drawod, "DDoS attack detection and mitigation at SDN data plane layer," in Proc. 2nd Global Power, Energy and Communication Conf. (GPECOM), Oct. 2020, pp. 322–326.

J. Ye, X. Cheng, and J. Zhu, "A DDoS attack detection method based on SVM in software-defined network," Secur. Commun. Netw., vol. 2018, Apr. 2018, Art. no. 9804061.

J. Cui, M. Wang, and Y. Luo, "DDoS detection and defense mechanism based on cognitive-inspired computing in SDN," Future Gener. Comput. Syst., vol. 97, pp. 275–283, Aug. 2019.

K. G. Maheswari, C. Siva, and G. Nalinipriya, "Optimal cluster-based feature selection for intrusion detection system in web and cloud computing environment using hybrid teacher learning optimization enables deep recurrent neural network," 2023.

K. G. Maheswari, C. Siva, and G. N. Priya, "An optimal cluster-based intrusion detection system for defense against attack in web and cloud computing environments," 2023.

M. Baskar, T. Gnanasekaran, and S. Saravanan, "Adaptive IP traceback mechanism for detecting low-rate DDoS attacks," in Proc. IEEE Int. Conf. Emerg. Trends Comput., Commun. Nanotechnol. (ICECCN), Mar. 2013.

M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: Methods, systems and tools," IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 303–336, 1st Quart. 2014.

M. H. Bhuyan and E. Elmroth, "Multi-scale low-rate DDoS attack detection using the generalized total variation metric," in Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA), Dec. 2018.

N. M. Saravana Kumar, S. Deepa, C. N. Marimuthu, T. Eswari, and S. Lavanya, "Signature-based vulnerability detection over wireless sensor network for reliable data transmission," 2016.

N. Zhang, F. Jaafar, and Y. Malik, "Low-rate DoS attack detection using PSD based entropy and machine learning," in Proc. 6th IEEE Int. Conf. Cyber Security and Cloud Comput., Jun. 2019.

N. Agrawal and S. Tapaswi, "Defense mechanisms against DDoS attacks in a cloud computing environment: State-of-the-art and research challenges," IEEE Commun. Surveys Tuts., vol. 21, no. 4, pp. 3769–3795, Oct. 2019.

O. Osanaiye, K.-K. R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework," J. Netw. Comput. Appl., vol. 67, pp. 147–165, May 2016.

S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments," IEEE Access, vol. 7, pp. 80813–80828, 2019.

T. A. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow TCAM exhaustion DDoS attack," in ICT Systems Security and Privacy Protection, S. De Capitani di Vimercati and F. Martinelli, Eds. Cham, Switzerland: Springer, 2017.

T. Apostolovic, N. Stankovic, K. Milenkovic, and Z. Stanisavljevic, "DDoSSim—System for visual representation of the selected distributed denial of service attacks," in Proc. Zooming Innovation Consumer Technol. Conf. (ZINC), May 2018.

W. Zhijun, X. Qing, W. Jingjie, Y. Meng, and L. Liang, "Low-rate DDoS attack detection based on factorization machine in software defined network," IEEE Access, vol. 8, pp. 17404–17418, 2020.

X. Liu, X. Yin, and Y. Hu, "CPSS LR-DDoS detection and defense in edge computing utilizing DCNN Q-learning," IEEE Access, vol. 8, pp. 42120–42130, 2020.

Y. Xiang, K. Li, and W. Zhou, "Low-rate DDoS attacks detection and traceback by using new information metrics," IEEE Trans. Inf. Forensics Security, vol. 6, no. 2, pp. 426–437, Jun. 2011.

Y. Yu, L. Guo, Y. Liu, J. Zheng, and Y. Zong, "An efficient SDN-based DDoS attack detection and rapid response platform in vehicular networks," IEEE Access, vol. 6, pp. 44570–44579, 2018.

Y. Tarasov, E. Pakulova, and O. Basov, "Modeling of low-rate DDoS attacks," in Proc. 12th Int. Conf. Security Inf. Netw., 2019.

Y. Wang, T. Hu, G. Tang, J. Xie, and J. Lu, "SGS: Safe-guard scheme for protecting control plane against DDoS attacks in software-defined networking," IEEE Access, vol. 7, pp. 34699–34710, 2019.

Z. Li, H. Jin, D. Zou, and B. Yuan, "Exploring new opportunities to defeat low-rate DDoS attack in container-based cloud environment," IEEE Trans. Parallel Distrib. Syst., vol. 31, no. 3, pp. 695–706, Mar. 2020.