A Lightweight Framework for DDoS Detection & Mitigation in SDN

PROJECT TEAM

Muhammad Owais Siyal	20k-0201
Muhammad Ammar Siddique	20k-0342
Murtaza Salman	20k-0398

PROJECT SUPERVISOR

Mr. Shoaib Raza

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

FAST SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES KARACHI CAMPUS
May 2024

Project Supervisor	Mr. Shoaib Raza
Project Team	Muhammad Owais Siyal 20k-0201 M. Ammar Siddique 20k-0342 Murtaza Salman 20k-0398
Submission Date	May 16, 2024

Mr. Shoaib Raza ________________________________

Supervisor

Dr. Atif Tahir ________________________________

Head of Department

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

Acknowledgement

We extend our heartfelt gratitude to our esteemed supervisor, Mr. Shoaib Raza, for his unwavering support and invaluable guidance throughout the journey of this project. His deep knowledge and expertise in computer networks have been instrumental in shaping the trajectory of our research and refining our methodology. Mr. Raza's dedication to fostering a collaborative and innovative environment has inspired us to push the boundaries of our work and explore novel approaches.

Furthermore, we wish to acknowledge the tremendous support and resources provided by our university, which laid the foundation for our research. The access to state-of-the-art facilities, insightful lectures, and a wealth of academic literature have played a crucial role in our progress. We also wish to recognize the instrumental technical assistance provided by our dedicated research team. Their diligence, ingenuity, and collaborative spirit have been vital to our success, allowing us to surmount challenges and maintain momentum throughout the project.

Collaboration has been at the heart of our journey, and the camaraderie among our team members has enabled us to achieve goals that would have been impossible individually. Each member's unique skills and perspectives have enriched our work, and the collective effort has led to meaningful insights and significant progress.
We are profoundly grateful for the opportunity to contribute to the important field of car damage detection systems. We believe our research holds promise in advancing the accuracy and efficiency of such systems, potentially benefiting the automotive industry and its customers. It is our hope that our work will serve as a stepping stone for future innovations, paving the way for further advancements in this vital area of research.

Table Of Contents

List of Tables

Abstract

This project presents a lightweight framework for detecting and mitigating Distributed Denial of Service (DDoS) attacks in Software-Defined Networking (SDN) environments. SDN enhances network management but introduces vulnerabilities that make it susceptible to DDoS attacks. The framework includes flow collection, feature extension, anomaly detection, and mitigation modules. The Naïve Bayes model achieved 93.67% accuracy, with a recall of 1.00 and precision of 0.91. The logistic regression model showed 97.08% accuracy, with a recall of 0.99 and precision of 0.97. The framework was validated using Mininet and the Ryu controller, with traffic data collected via the SDN controller. A React-based dashboard provides real-time monitoring for administrators. This framework contributes to network security by offering an effective solution for DDoS detection and mitigation in SDN environments. Future work will enhance the mitigation module and refine the user interface.

.

1. Introduction

Software-Defined Networking (SDN) stands at the forefront of a technological revolution, reshaping the landscape of network architecture by decoupling control logic from forwarding logic. Through the abstraction of control logic into a central controller and communication facilitated by southbound Application Programming Interfaces (APIs), notably employing the OpenFlow protocol, SDN provides a comprehensive view of the network, thereby enhancing decision-making capabilities. This transformative approach allows network administrators to program and manage the entire network efficiently, addressing issues that once consumed substantial time with remarkable speed [1].

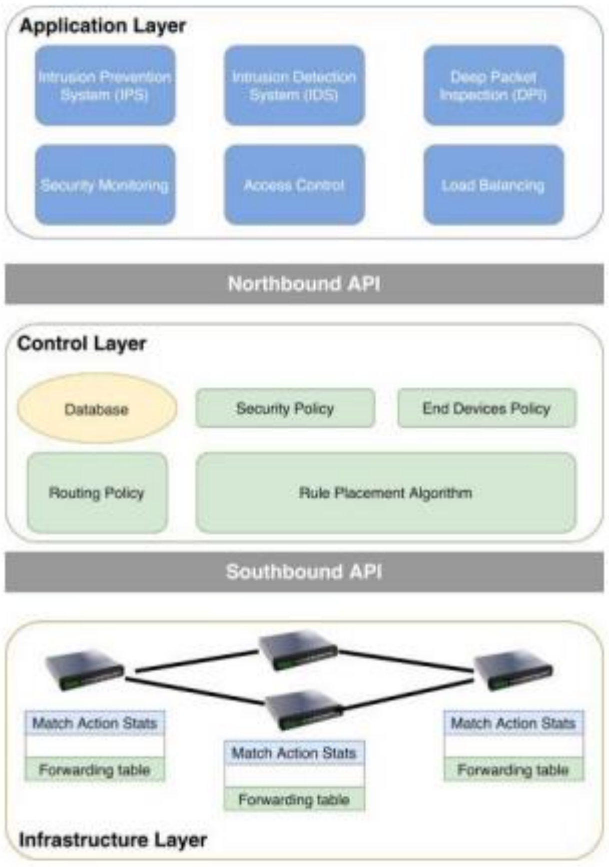


Figure 1: Simplified SDN Architecture

While industry behemoths like Google, Microsoft, and HP explore and adopt SDN integration, the benefits it brings to network management are indisputable. However, amid the positive transformations, challenges emerge, with Distributed Denial of Service (DDoS) attacks standing out as a persistent threat. Despite the evolution of detection and mitigation solutions, DDoS attacks continue to escalate in power, frequency, and severity, creating an urgent need for a highly efficient Intrusion Detection System (IDS) framework [1]. A Distributed Denial of Service (DDoS) attack represents a malicious attempt to disrupt normal traffic, overwhelming a targeted server, service, or network with a flood of internet traffic. This disruptive tactic utilizes multiple compromised computer systems as sources of attack traffic, including computers and networked resources like IoT devices. Analogous to an unexpected traffic jam on a highway, a DDoS attack impedes regular traffic from reaching its destination. Immediate action upon detection is imperative, as delayed response can lead to server crashes and prolonged recovery times.

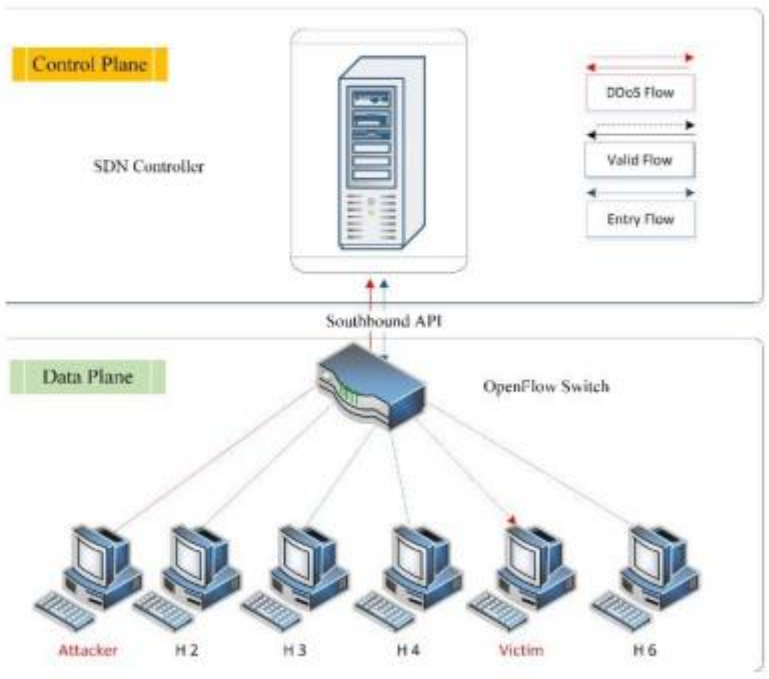


Figure 2: A simplified DDoS attack in an SDN environment.

Mitigating DDoS attacks poses a unique challenge, as attackers employ sophisticated techniques to masquerade fake traffic as legitimate. The criticality of swift response is emphasized for enterprises operating at the edge, engaging in mission-critical activities that cannot afford downtime. DDoS mitigation emerges as a protective layer within SDN, ensuring the ongoing availability of essential activities and services. SDN not only revolutionizes network architecture but also plays a pivotal role in fortifying network security. The dynamic nature of SDN allows for the design, construction, and operation of networks, yet it also exposes vulnerabilities. Particularly, the emergence of new security concerns, exemplified by the frequent launch of DDoS attacks against SDN networks, underscores the need for robust security measures. As SDN transforms the network landscape, the persistent threat of DDoS attacks necessitates a proactive approach to network security. The integration of SDN and the development of efficient IDS frameworks become imperative in countering evolving threats. While SDN empowers network administrators with unprecedented control and efficiency, the journey towards its effective deployment requires a comprehensive understanding of emerging challenges, particularly those posed by DDoS attacks. The intricate dance between innovation and security underscores the ongoing evolution of network management in the era of Software-Defined Networking.

2. Problem Statement

The rapid growth of data transfer on the internet has led to the widespread deployment of high- speed networks in commercial and educational institutions. In today's network environments, the identification of potential Distributed Denial of Service (DDoS) attacks is crucial to ensure the availability and integrity of network services. Traditional Intrusion Detection Systems (IDS) struggle to cope with the challenges posed by heavy traffic loads, resulting in packet drops and reduced detection accuracy. This limitation impedes the use of IDS in high-speed networks. Existing studies have addressed the performance challenges of IDS in high-speed networks, emphasizing the importance of efficient packet capturing and data processing approaches. While these studies have improved IDS performance, challenges persist in ensuring low latency for legitimate traffic and maintaining high detection accuracy in the face of evolving DDoS threats.
To address the aforementioned challenges, this research aims to leverage machine learning (ML) models for rapid detection of Distributed Denial of Service (DDoS) attacks in high-speed networks. By integrating ML algorithms into the Intrusion Detection System (IDS), we intend to enhance the system's ability to identify and respond to malicious traffic patterns promptly and accurately.

Furthermore, this study proposes the development of a user-friendly interface for real-time packet visualization and analysis. This interface will provide network administrators with intuitive tools to monitor network traffic, detect anomalies, and take immediate action when necessary. Through the visualization of packet flows and the utilization of ML-based anomaly detection techniques, our system aims to empower network operators to effectively manage network security threats while minimizing disruption to legitimate network traffic.

By combining efficient packet capturing techniques, advanced ML models, and a user-friendly interface for visualization and analysis, this research endeavors to address the performance limitations of traditional IDS in high-speed networks. Ultimately, the goal is to ensure the availability, integrity, and security of network services in the face of evolving DDoS threats.

3. Related Work

In response to the exponential surge in internet data transmission, organizations and academic institutions have increasingly embraced high-speed network connections, ushering in an era where the role of Intrusion Detection Systems (IDS) is nothing short of pivotal. These systems play a crucial role in identifying and mitigating potential network threats within the dynamic and evolving digital landscape. However, the relentless growth in network traffic volume presents formidable challenges for IDS, manifesting in packet losses and a notable reduction in accuracy. These challenges arise from the intricate processing demands imposed by the diverse and high-volume nature of network traffic, creating significant roadblocks to the effective deployment of IDS in high-speed network environments.

Addressing these challenges head-on, Software-Defined Networking (SDN) has emerged as a robust and adaptive solution. By segregating the control plane and data plane, SDN offers a comprehensive set of advantages, including enhanced manageability, control, dynamic rule updating, comprehensive network analysis, and a unified global view facilitated by a centralized controller. While these advantages position SDN as a promising solution, it is essential to acknowledge the concomitant challenges, encompassing security vulnerabilities and deployment intricacies [14].

The research landscape has witnessed extensive efforts [3]-[6] focused on assessing the efficacy of IDS in high-speed networks, with a specific emphasis on the challenges posed by heavy traffic loads. A seminal study by Hu et al. [4] meticulously outlines the hurdles associated with packet capture systems, proposing innovative solutions through the implementation of multithreaded architectures. This architectural approach aims to optimize IDS performance by mitigating overload, thereby reducing packet losses, and enhancing CPU utilization. Furthermore, additional studies [7], [8] accentuate the intricate correlation between packet loss rates and IDS effectiveness, shedding light on the dual influence of packet capture and inspection mechanisms [2].

A more granular exploration conducted by Hu et al. [6] delves into Suricata and Snort, both open-source IDSs, with the primary objective of augmenting their performance in high-speed networks. The study meticulously examines various factors impeding IDS utilization, offering valuable insights into memory usage, CPU utilization, packet loss rates, and detection accuracy. These findings provide a nuanced understanding that significantly contributes to the ongoing development of novel IDS systems tailored explicitly for high-speed networks.

In the realm of attack detection, our proposed Model transcends conventional features by incorporating additional metrics aimed at enhancing precision. These supplementary metrics encompass the average flow packet size, counts of flows directed to the same host within the past 5 seconds, and counts of flows targeting the same host and port within the last 5 seconds. The effectiveness of attack detection is further augmented by the utilization of six machine learning algorithms [13], introducing a multifaceted approach to fortify network defenses against evolving threats.

Introducing the Scattered Denial-of-Service Mitigation Tree Architecture (SDMTA), we propose a pioneering DDoS mitigation strategy tailored explicitly for hybrid cloud environments. The SDMTA seamlessly integrates network monitoring into detection procedures, providing a comprehensive evaluation of detection rates over diverse input datasets [15]. This integrative approach seeks to address the multifaceted nature of modern cyber threats and fortify network security in hybrid cloud environments.

While SDN introduces groundbreaking changes to networking paradigms, it simultaneously introduces new security threats, most notably distributed denial-of-service (DDoS) attacks. The centralized controller in SDN becomes a potential single point of attack and failure [16]. Additionally, the integration of the Internet of Things (IoT) into networks poses unprecedented security challenges, prompting the proposal of a Security Information and Event Management based IoT botnet DDoS attack detection and mitigation system [17]. This system stands as a testament to the proactive approach required to safeguard network integrity in the face of emerging threats associated with IoT integration.

The transition to Internet Protocol version six (IPv6), while alleviating the issue of IPv4 address depletion, brings forth new challenges, particularly in the form of ICMPv6-based Denial of Service (DoS) and DDoS attacks. IDSs tailored to combat these specific security issues [18] are instrumental in fortifying network security, contributing to the continual evolution and enhancement of network defense mechanisms. As the digital landscape evolves, it becomes imperative to continually refine and bolster security frameworks to effectively counter emerging threats and vulnerabilities.

Machine learning techniques give a compelling performance for the detection of DDoS attacks in SDN. The ML techniques effectively detect the attack against the control plane of the SDN controller. This section briefly discusses the current works to detect DDoS attacks in SDN using machine learning techniques. Furthermore, features selection-based ML models and techniques presented by the researchers in recent years are analyzed in the section. In [19], a statistical and machine learning-based method is proposed. K-mean and K nearest neighbors (KNN) based hybrid model is proposed in [20]. Support vector machine (SVM) based DDoS detection in SDN was performed in [21]. Kernel principal component analysis (KPCA), Genetic algorithm (GA) and SVM based method is presented in [22]. In [24], an entropy-based technique that uses Flow samples are presented for traffic classification, and it just focuses on a standard distribution of the traffic. A COFFEE model that extracts the features from the flow for the detection of attack is present in [25]. For the extraction of more features, the suspected flow is sent to the controller. In [26], many features are utilized by machine learning techniques to detect the attack. Furthermore, in [27], traffic features based on a lightweight DDoS attack detection algorithm are presented.

The Self-organizing map (SOM) is used for the analysis and extraction of traffic information. After the extraction of features, the Artificial Neural Network is used for the detection of DDoS attacks. In [28], the researchers proposed a k-nearest neighbor-based algorithm that uses the abstract distance between the traffic features to detect the attack. This algorithm gives effective results for the detection of abnormal flow and reduces the false alarm rate. Although the researchers proposed various machine learning-based solutions for detecting DDoS attacks, these solutions have some limitations in terms of optimal feature selection, low accuracy, and efficiency. In [29], Naïve bayes (NB) and K-mean clustering-based method was proposed to detect DDoS attacks. The K-mean cluster method clusters the traffic data that show similar behaviors, and the Naïve Bayes algorithm classifies the cluster data into standard and attacks traffic. In [30], Artificial Neural Network-based methods are proposed to detect known and unknown DDoS attacks.

The researcher in the controller applies a dynamic Multilayer perceptron (MLP) that works with a feedback mechanism to detect DDoS attacks [31]. They use some selected features that cannot distinguish between standard and attack traffic flows. In [32], the authors introduced the trigger mechanism that detects the DDoS attack faster and reduces the switches' workload. The trigger mechanism applied on the controller's control plane effectively attacks but increases the controller's workload. Zang et al. [33] proposed a finer-grained method that uses the flow features to detect an attack. It extracts the 39 different traffic features from the flow and improves detection accuracy.

Table 1: Summary of papers and their contributions

Papers	**Focus**	**Contribution**

Hu et al. [4]	Challenges of packet capture Systems in high-speed networks.	● Proposes multithreaded architectures. ● Aims to optimize IDS performance. ● Reduces packet losses. ● Enhances CPU utilization.
Hu et al. [6]	Performance enhancement of Suricata and Snort in high-speed networks.	● Examines factors affecting IDS utilization. ● Memory usage ● CPU utilization ● Packet loss rates ● Detection accuracy
SDMTA [15]	DDoS mitigation strategy for hybrid cloud environments.	● Introduces SDMTA. ● Integrates network monitoring into detection procedures. ● - Aims for comprehensive evaluation of detection rates.
IPv6-specific IDS [18]	Security challenges in IPv6 networks, specifically ICMPv6-based attacks.	● Proposes IDSs tailored for IPv6 transition. ● Focuses on combating security issues related to ICMPv6-based attacks.
Machine Learning in SDN [19-33]	Various machine learning-based methods for DDoS detection in SDN.	● Includes multiple methods ● K-mean and K nearest neighbors (KNN) hybrid model ● Support Vector Machine (SVM) ● Kernel PCA ● Genetic Algorithm (GA) ● Self-organizing map (SOM) ● Naïve Bayes (NB) ● Artificial Neural Network-based methods

4. System Requirements

4.1 Functional Requirements

The system must fulfill the following functional requirements:

- **DDoS Attack Detection:** The machine learning model must accurately identify Distributed Denial of Service (DDoS) attacks based on network traffic data.
- **Controller Communication:** The system should maintain a robust communication protocol with the network controller to obtain traffic data as needed.
- **Traffic Mitigation:** Upon detection of a DDoS attack, the system must initiate a mitigation process that specifically targets malicious traffic without

affecting normal network operations.

4.2 Non-Functional Requirements

The system's performance is characterized by the following non-functional requirements:

- **Response Time:** The system must detect DDoS attacks within 20 seconds of their initiation.
- **Traffic Analysis Frequency:** It should analyze incoming traffic data every 5 seconds to classify it as normal or anomalous.
- **Mitigation Timeframe:** The system must be capable of mitigating identified malicious traffic within 2 minutes, including pinpointing the compromised port number.

4.3 Software and Hardware Requirements

To support the intended functionalities, the system requires the following software and hardware:

- **Operating Systems:** Compatible with Windows, Linux, and macOS.
- **Virtualization Software:** VMware or VirtualBox to simulate network environments.
- **Network Tools:**
  - **Ryu Controller:** For network management and control based on OpenFlow protocols.
  - **Mininet:** To create a realistic virtual network for testing and development.
- **Development Tools:**
  - **Node.js**: For building scalable network applications.
  - **Flask:** A lightweight WSGI web application framework used for service integration.

4.4 Libraries

The implementation will utilize various Python libraries to support machine learning and data manipulation functionalities:

- **Scikit-learn:** For implementing machine learning algorithms.
- **Numpy:** Essential for handling large, multi-dimensional arrays and matrices.
- **Pandas:** Provides high-performance, easy-to-use data structures.

5. Design

The design of the DDoS detection and mitigation framework involves four core components, each playing a critical role in ensuring effective detection and mitigation of malicious traffic within an SDN environment:

5.1 Flow Collector Module:

- **Purpose**: To gather traffic flow data from the network switches controlled by the SDN controller.
- **Process**: This module collects real-time traffic data, including flow identifiers (such as source/destination IP, source/destination ports, protocol) and flow counters (such as packet and byte counts).
- **Communication**: Utilizes the OpenFlow protocol to request flow statistics from switches via the SDN controller, which then aggregates this data and sends it to the Flow Collector.

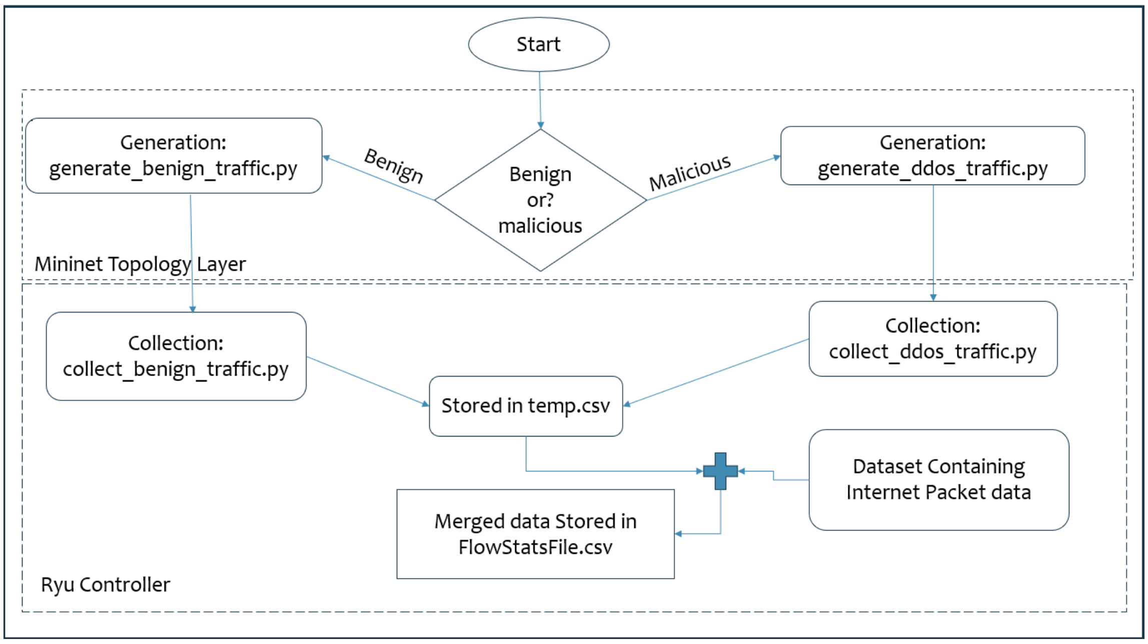

Start
Generation: generate_benign_traffic.py
Benign
Benign or? malicious
Malicious
Generation: generate_ddos_traffic.py
Mininet Topology Layer
Collection: collect_benign_traffic.py
Collection: collect_ddos_traffic.py
Stored in temp.csv
Dataset Containing Internet Packet data
Merged data Stored in FlowStatsFile.csv
Ryu Controller

Figure 3: Flow collection model diagram.

Feature Engineering and Model Training:

- **Purpose:** Enhance the ability of the machine learning model to accurately identify and respond to DDoS threats in network traffic and to refine the input data to improve model efficiency and accuracy.
- **Process:** The dataset is processed through several stages of transformation

to isolate the most informative features and prepare them for model training. This systematic refinement helps in reducing the dimensionality of the data, which is crucial for effective model training and performance.

- **Model Selection**: Logistic Regression was chosen for its high detection accuracy (97.08%) and a high recall (0.99) and precision (0.97) and efficient training time (~40 seconds). Naïve Bayes was also considered but lower accuracy (93.67%) had a relatively similar training time (30 seconds).

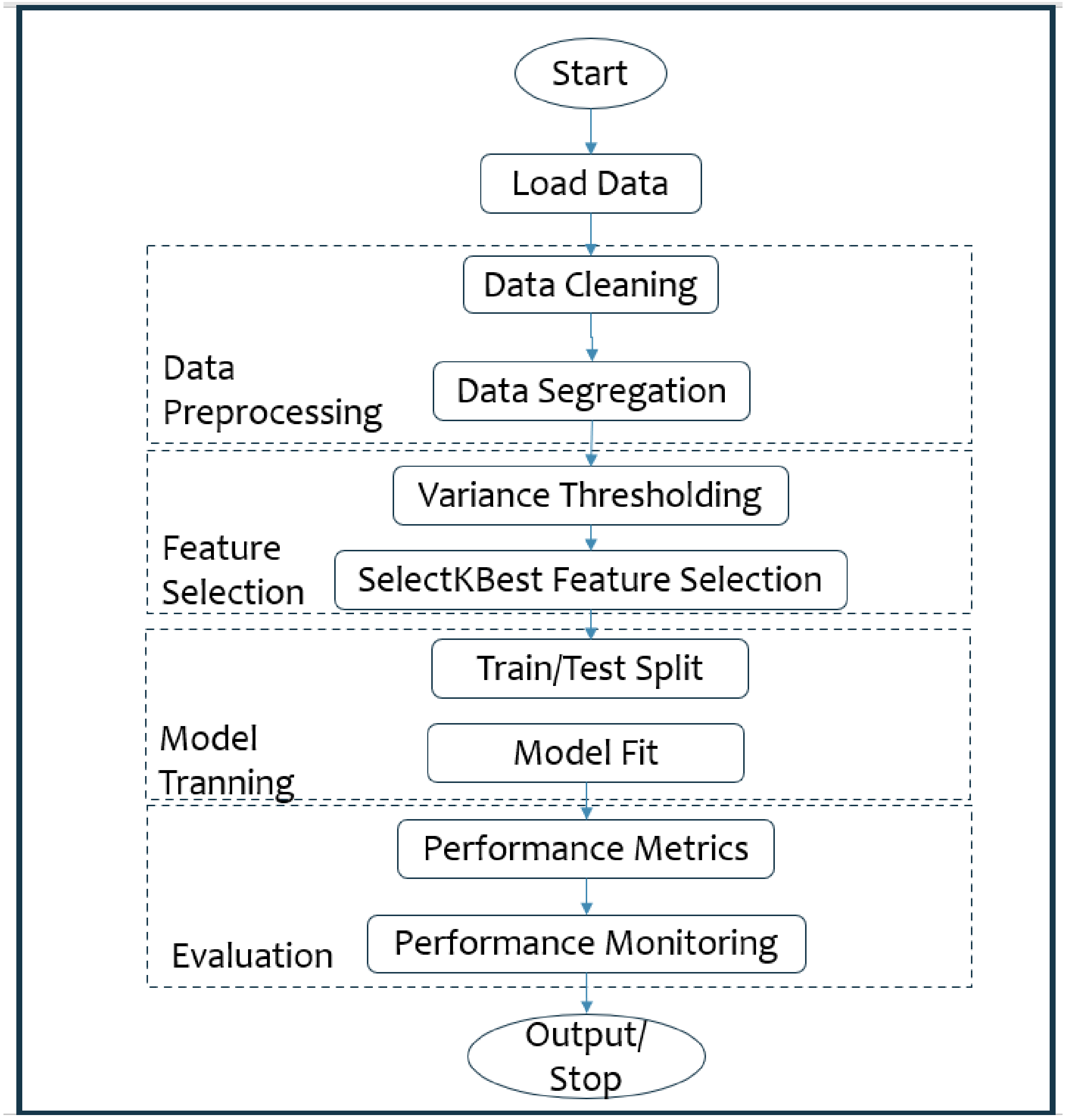


Figure 4: Feature Selection and Model Training Model.

Anomaly Detection Module:

- **Purpose**: To identify anomalies in network traffic indicative of DDoS attacks using machine learning.
- **Process**: The incomming traffic is collected on the switches at set intervals by the controller and is then passed to the trained model, the model first

preprocesses the packets and then labels them.

- **Detection**: Once the model is loaded into the controller, the model monitors incoming traffic in real-time, classifying each flow as either normal or anomalous. The model keeps a threshold value of 20%, i.e. for the traffic in the network it checks and labels all the traffic and sees if the Malicious traffic exceeds 20%, if it does it triggers a DDOS attack warning, this approach allows us to minimize false alarms.

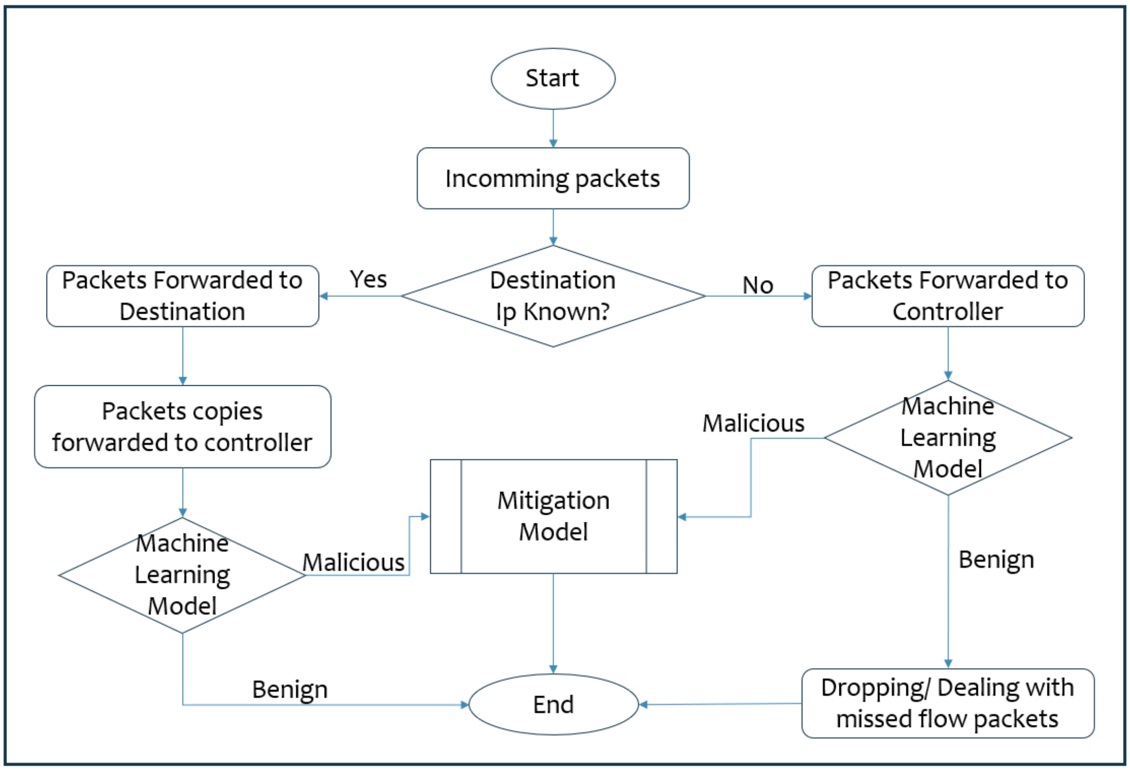

Start
Incomming packets
Destination Ip Known?
Yes
No
Packets Forwarded to Destination
Packets Forwarded to Controller
Packets copies forwarded to controller
Machine Learning Model
Malicious
Mitigation Model
Machine Learning Model
Malicious
Benign
Benign
End
Dropping/ Dealing with missed flow packets

Figure 5: Anomaly Detection Module.

Mitigation Module:

- **Purpose**: To act against detected anomalies to protect the network from DDoS attacks.
- **Process**: Based on the classification from the Anomaly Detection module, this module takes immediate action to mitigate the detected attack. Mitigation strategies include isolating or rerouting suspicious traffic, updating flow tables to block malicious traffic, or adjusting network policies dynamically.
- **Customization**: Allows administrators to define specific mitigation rules and strategies based on the network's security requirements, ensuring minimal impact on legitimate traffic.

6. Overall System Design

The overall design integrates these modules into a cohesive framework that operates in real-time to detect and respond to DDoS attacks efficiently:

- **System Integration**: Each module communicates seamlessly with others, allowing for a continuous flow of data from collection to mitigation.
- **Scalability and Efficiency**: The design leverages lightweight machine learning models and modular architecture to ensure scalability and efficiency, crucial for large-scale SDN deployments.
- **Flexibility**: The framework is adaptable to various network configurations, supporting a wide range of feature sets and attack scenarios.

The design ensures that the framework provides a robust solution for DDoS detection and mitigation, tailored specifically for the dynamic requirements of SDN environments.

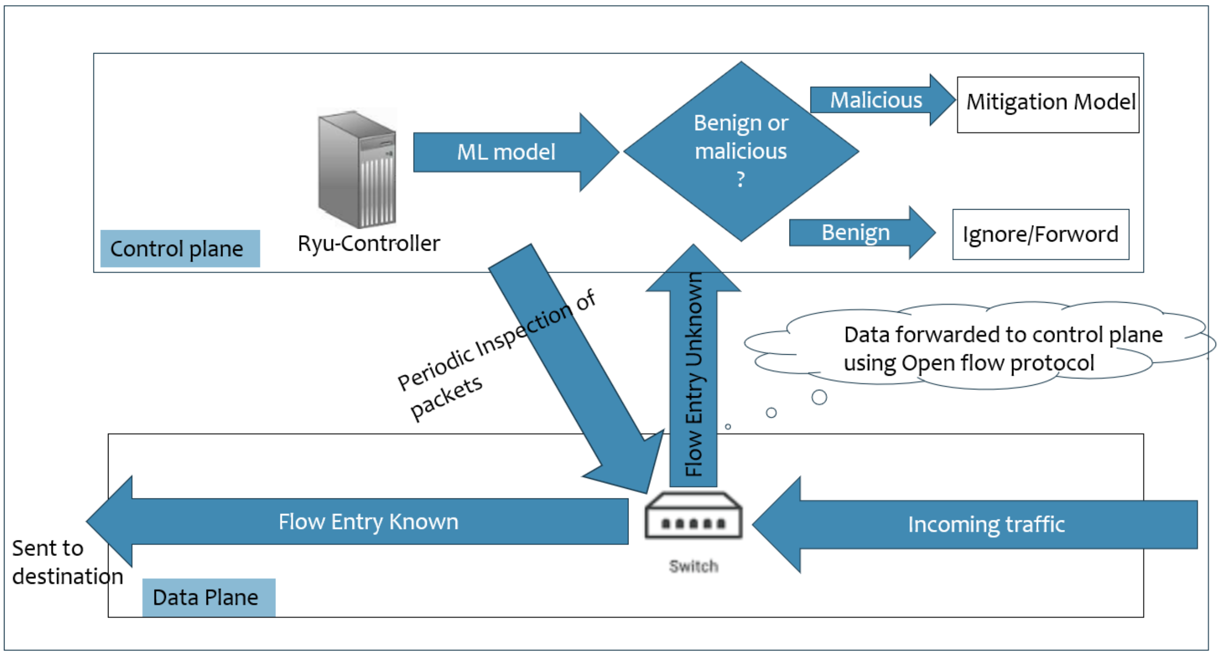


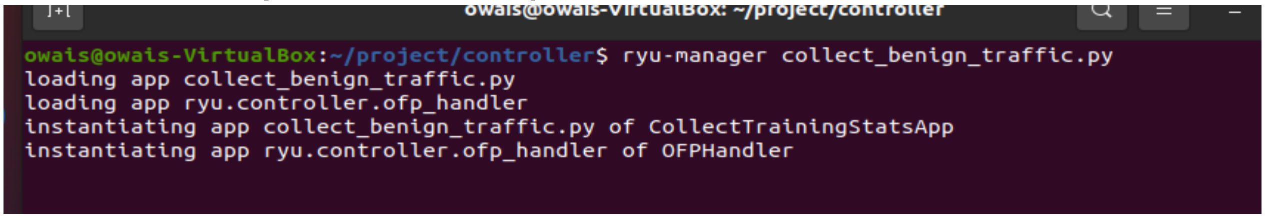
Figure 6: System Architecture Design

7. Implementation

The implementation of the DDoS detection and mitigation framework is meticulously structured to ensure seamless integration within a Software-Defined Networking (SDN) environment. This section delves into the specific technologies, tools, and methodologies employed across the core modules of the system.

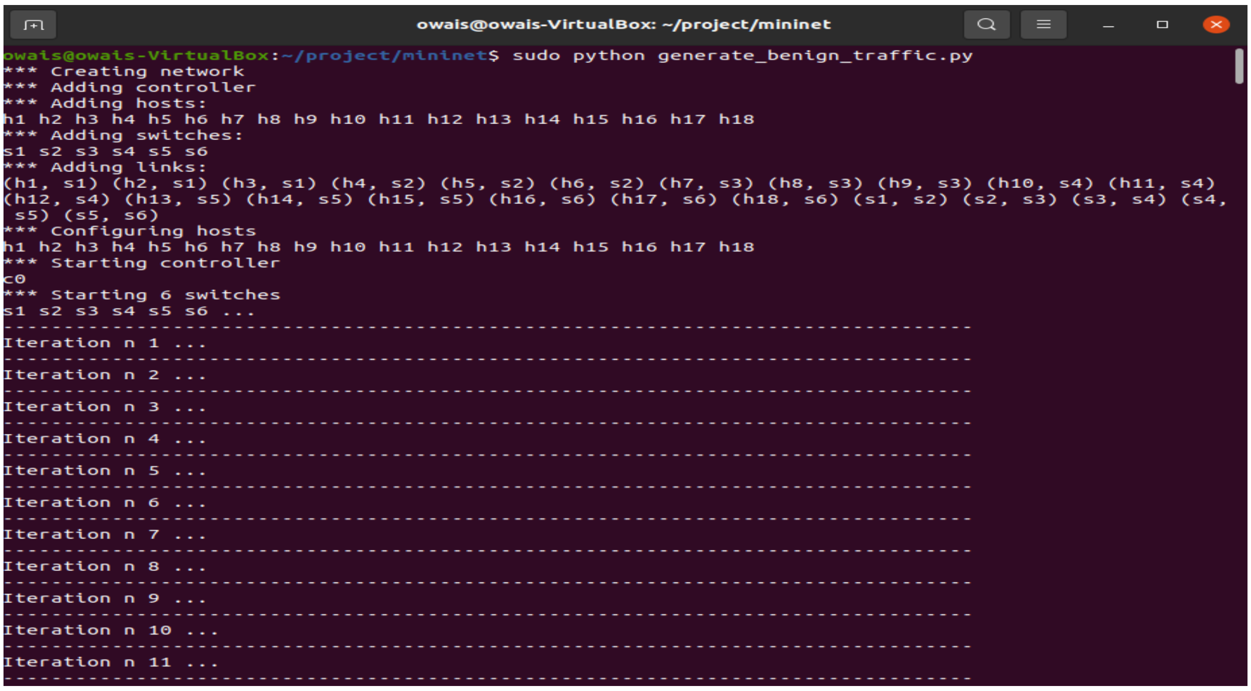
7.1. Flow Collector Module:

- **Technology Stack**: Python programming language, coupled with the Ryu SDN controller.
- **Functionality**: This module utilizes the Ryu controller to send OpenFlow messages to network switches, requesting the current state of their flow tables.
- **Implementation Details**: The flow collector module periodically sends a **FlowStatsRequest** to each switch. When the switches reply with a **FlowStatsReply**, this module parses the flow data, including metrics like packet counts and byte counts. This data is then forwarded to the Feature Extender Module.
- **Feature Implementation Snapshots**:



```
owais@owais-VirtualBox:~/project/controller$ ryu-manager collect_benign_traffic.py
loading app collect_benign_traffic.py
loading app ryu.controller.ofp_handler
instantiating app collect_benign_traffic.py of CollectTrainingStatsApp
instantiating app ryu.controller.ofp_handler of OFPHandler
```

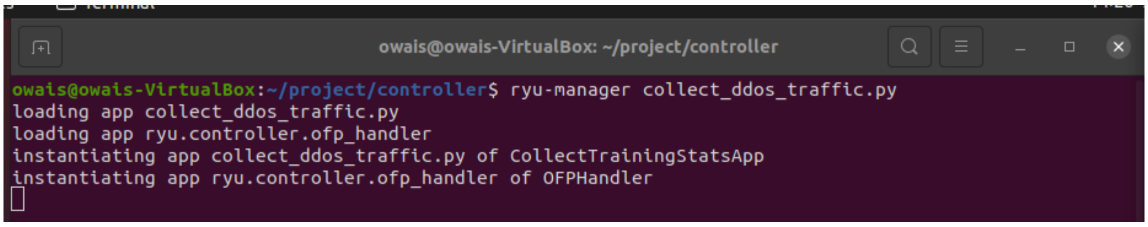
Figure 7: Collect Normal traffic.



owais@owais-VirtualBox: ~/project/mininet

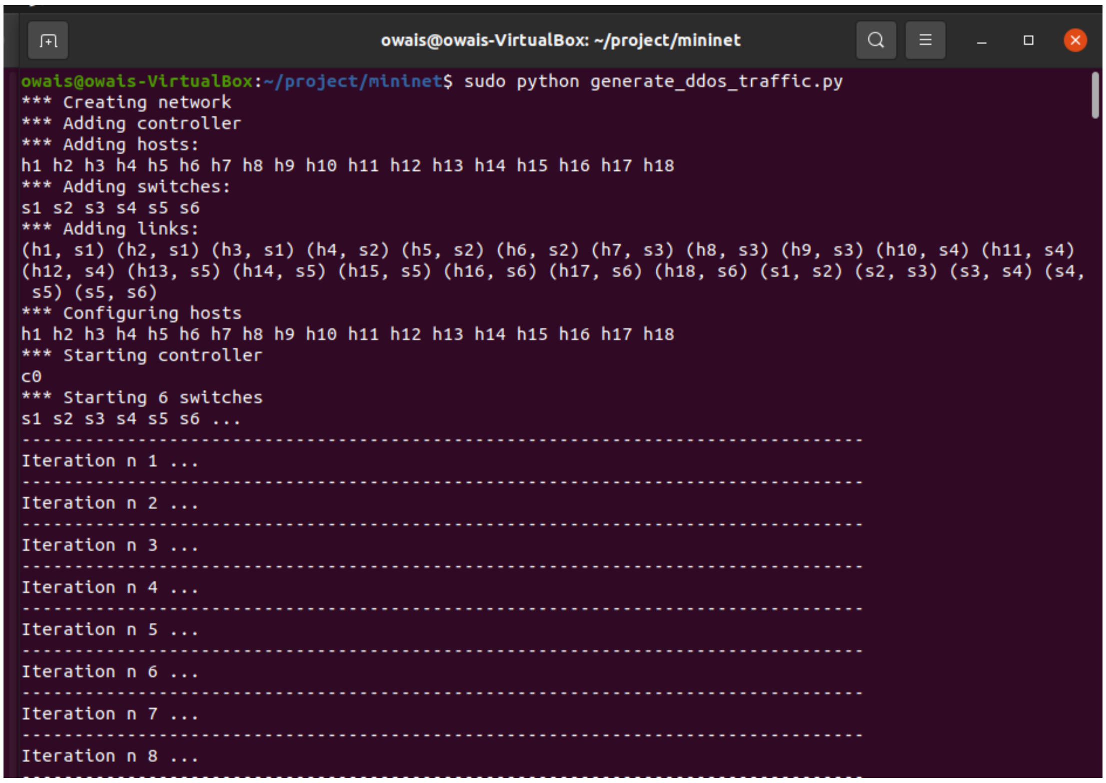
```
owais@owais-VirtualBox:~/project/mininet$ sudo python generate_benign_traffic.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3) (h10, s4) (h11, s4)
(h12, s4) (h13, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s6) (h18, s6) (s1, s2) (s2, s3) (s3, s4) (s4,
 s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
--------------------------------------------------------------------------------
Iteration n 1 ...
--------------------------------------------------------------------------------
Iteration n 2 ...
--------------------------------------------------------------------------------
Iteration n 3 ...
--------------------------------------------------------------------------------
Iteration n 4 ...
--------------------------------------------------------------------------------
Iteration n 5 ...
--------------------------------------------------------------------------------
Iteration n 6 ...
--------------------------------------------------------------------------------
Iteration n 7 ...
--------------------------------------------------------------------------------
Iteration n 8 ...
--------------------------------------------------------------------------------
Iteration n 9 ...
--------------------------------------------------------------------------------
Iteration n 10 ...
--------------------------------------------------------------------------------
Iteration n 11 ...
--------------------------------------------------------------------------------
```

Figure 8: Starting Normal traffic generation.



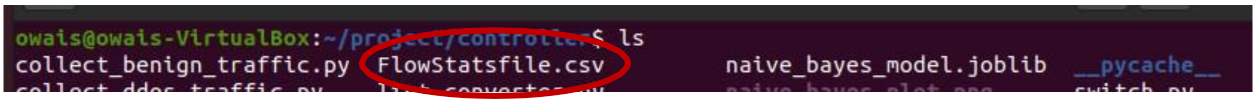
```
owais@owais-VirtualBox:~/project/controller$ ryu-manager collect_ddos_traffic.py
loading app collect_ddos_traffic.py
loading app ryu.controller.ofp_handler
instantiating app collect_ddos_traffic.py of CollectTrainingStatsApp
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figure 9: Starting DDOS traffic collection controller.



```
owais@owais-VirtualBox:~/project/mininet$ sudo python generate_ddos_traffic.py
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3) (h10, s4) (h11, s4)
(h12, s4) (h13, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s6) (h18, s6) (s1, s2) (s2, s3) (s3, s4) (s4,
 s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
c0
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
--------------------------------------------------------------------------------
Iteration n 1 ...
--------------------------------------------------------------------------------
Iteration n 2 ...
--------------------------------------------------------------------------------
Iteration n 3 ...
--------------------------------------------------------------------------------
Iteration n 4 ...
--------------------------------------------------------------------------------
Iteration n 5 ...
--------------------------------------------------------------------------------
Iteration n 6 ...
--------------------------------------------------------------------------------
Iteration n 7 ...
--------------------------------------------------------------------------------
Iteration n 8 ...
--------------------------------------------------------------------------------
```

Figure 10: Starting Controller



```
owais@owais-VirtualBox:~/project/controller$ ls
collect_benign_traffic.py  FlowStatsfile.csv          naive_bayes_model.joblib  __pycache__
```

Figure 11: FlowStatsfile.csv

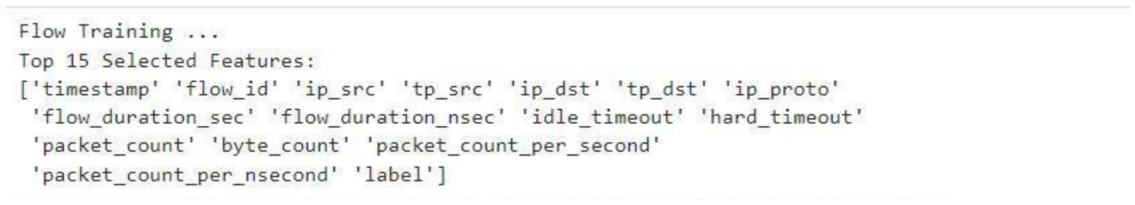
7.2. Feature Engineering and Model Training:

- **Technology Stack**: Python scripting for data manipulation, using libraries such as scikit-learn for implementing ML algorithms, feature selection and Pandas for handling large datasets efficiently.

- **Functionality**: Standardizes the data and splits it into training and testing sets to ensure the model is not biased toward the structure of the data it was trained on. Divides the dataset into features **(‘X_flow’)** and labels **(‘y_flow’)**, separating inputs from the target variable. Applies variance thresholding and ANOVA F-test to retain features with the highest statistical significance.

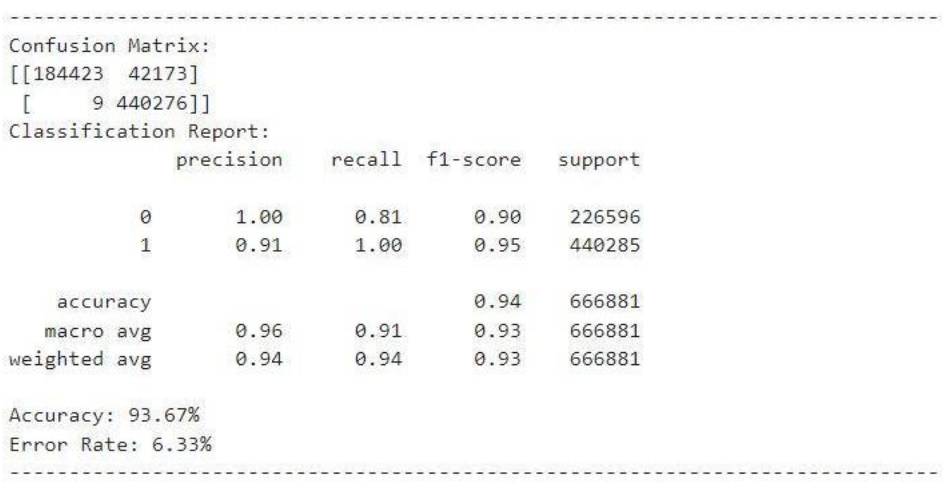
- **Implementation Details**: **Variance Thresholding** technique is applied to remove features with zero variance, which are non-contributive to model predictions. The **SelectKBest** method then refines this further by selecting the top 15 features based on **ANOVA F-test scores**, focusing on the most relevant attributes for the model. Data normalization through Standard Scaling ensures that the **Gaussian Naive Bayes** classifier performs optimally under the assumptions of a standard normal distribution. The model is trained on this scaled data and evaluated based on accuracy, precision, and recall metrics derived from the **confusion matrix**. Additionally, performance metrics such as **execution time, memory, CPU, and disk usage** are meticulously tracked to optimize and assess the efficiency of the training process.

- **Feature Implementation Snapshots:**



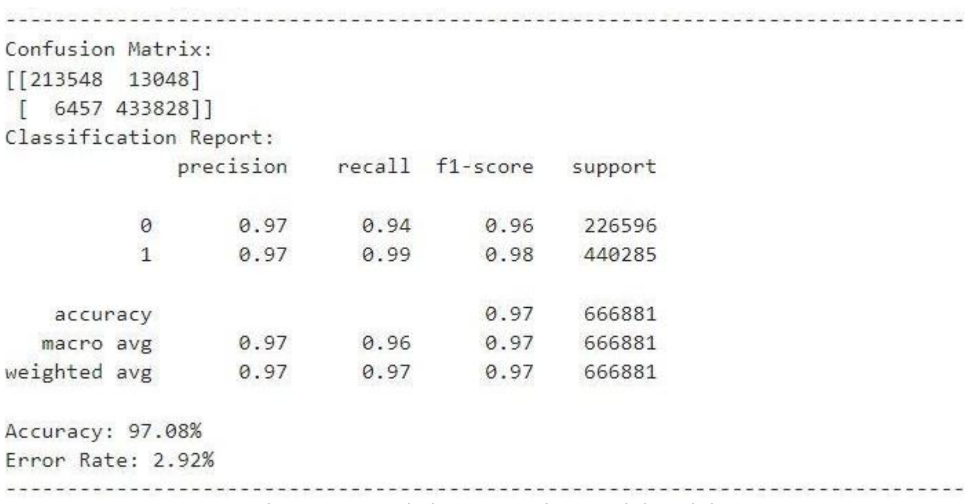
```
Flow Training ...
Top 15 Selected Features:
['timestamp' 'flow_id' 'ip_src' 'tp_src' 'ip_dst' 'tp_dst' 'ip_proto'
 'flow_duration_sec' 'flow_duration_nsec' 'idle_timeout' 'hard_timeout'
 'packet_count' 'byte_count' 'packet_count_per_second'
 'packet_count_per_nsecond' 'label']
```

Figure 12: Features Selected using SelectKBest.



```
------------------------------------------------------------------------------
Confusion Matrix:
[[184423  42173]
 [     9 440276]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.81      0.90    226596
           1       0.91      1.00      0.95    440285

    accuracy                           0.94    666881
   macro avg       0.96      0.91      0.93    666881
weighted avg       0.94      0.94      0.93    666881

Accuracy: 93.67%
Error Rate: 6.33%
------------------------------------------------------------------------------
```

Figure 13: Naive Bayes model training.

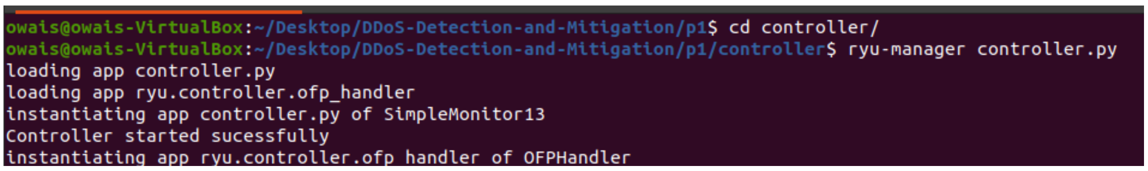


```
--------------------------------------------------------------------------------
Confusion Matrix:
[[213548  13048]
 [  6457 433828]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.94      0.96    226596
           1       0.97      0.99      0.98    440285

    accuracy                           0.97    666881
   macro avg       0.97      0.96      0.97    666881
weighted avg       0.97      0.97      0.97    666881

Accuracy: 97.08%
Error Rate: 2.92%
--------------------------------------------------------------------------------
```

Figure 14: Logistic Regression Model Training.

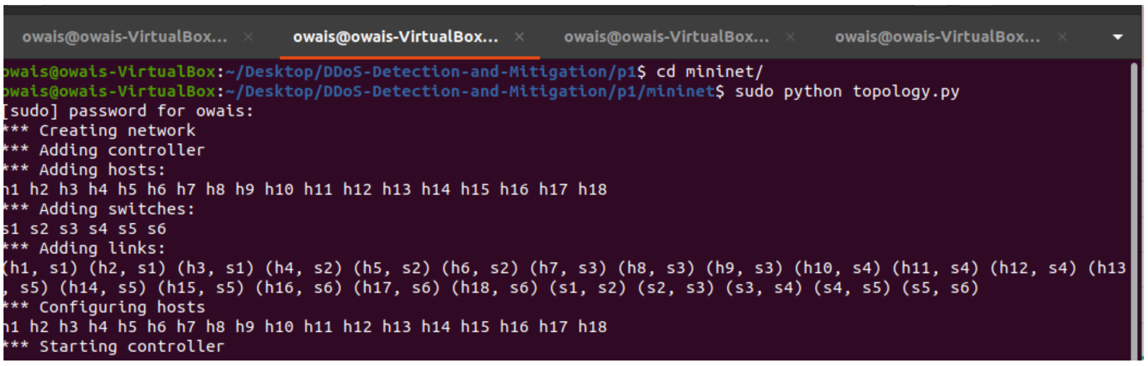
7.3. Anomaly Detection Module:

- **Technology Stack**: Scikit-learn for implementing machine learning models.
- **Functionality**: Utilizes trained machine learning models to classify traffic flows based on the features extended by the previous module.
- **Implementation Details**: This module trains the Naïve Bayes classifier using a dataset labeled with instances of normal and DDoS traffic. The training process involves feature selection and model validation to optimize performance and accuracy. After training, the model continuously receives new traffic data, classifying it in real-time to detect potential DDoS attacks.
- **Feature Implementation Snapshots:**



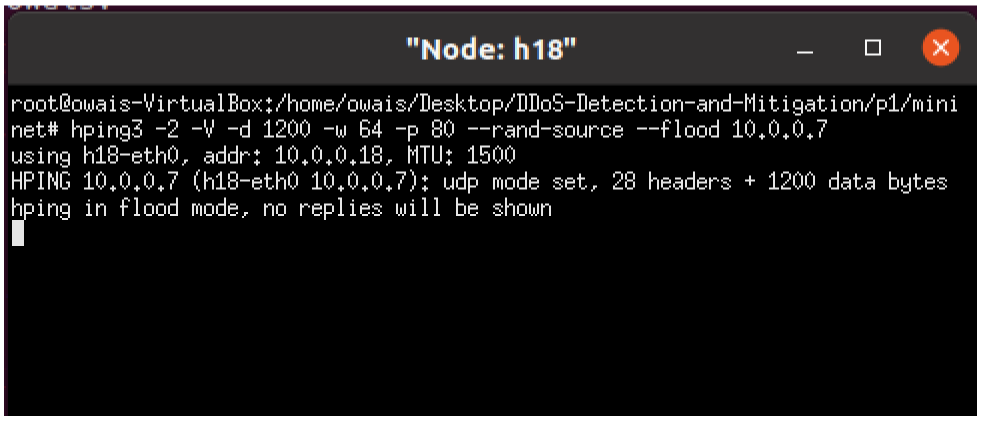
```
owais@owais-VirtualBox:~/Desktop/DDoS-Detection-and-Mitigation/p1$ cd controller/
owais@owais-VirtualBox:~/Desktop/DDoS-Detection-and-Mitigation/p1/controller$ ryu-manager controller.py
loading app controller.py
loading app ryu.controller.ofp_handler
instantiating app controller.py of SimpleMonitor13
Controller started sucessfully
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figure 15: Starting Controller


owais@owais-VirtualBox...
owais@owais-VirtualBox...
owais@owais-VirtualBox...
owais@owais-VirtualBox...

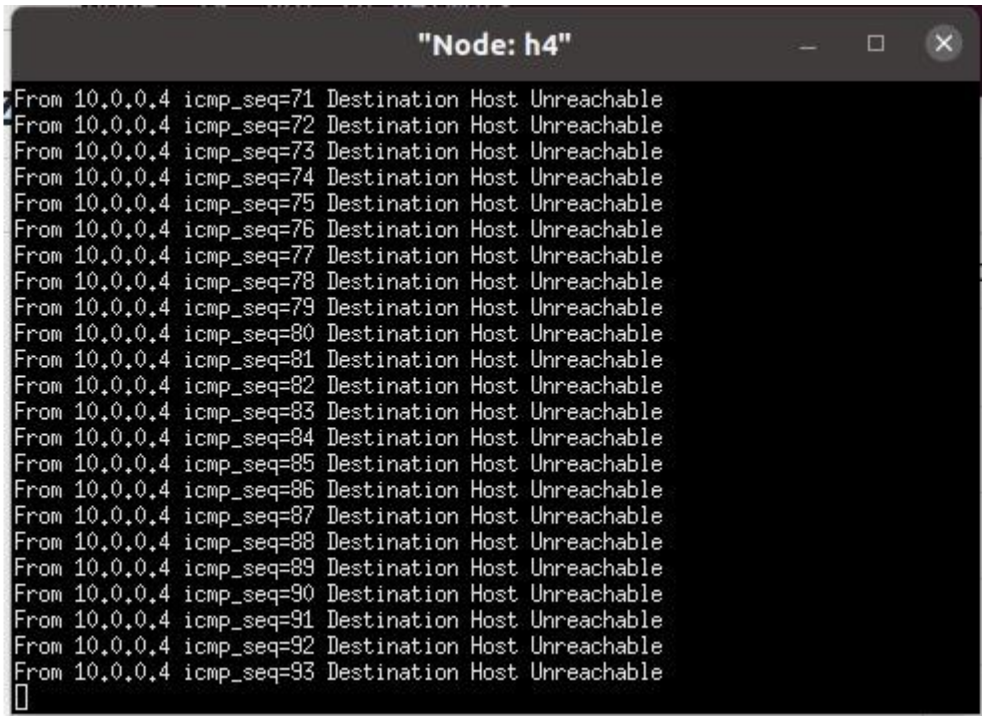
```
owais@owais-VirtualBox:~/Desktop/DDoS-Detection-and-Mitigation/p1$ cd mininet/
owais@owais-VirtualBox:~/Desktop/DDoS-Detection-and-Mitigation/p1/mininet$ sudo python topology.py
[sudo] password for owais:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s2) (h5, s2) (h6, s2) (h7, s3) (h8, s3) (h9, s3) (h10, s4) (h11, s4) (h12, s4) (h13
, s5) (h14, s5) (h15, s5) (h16, s6) (h17, s6) (h18, s6) (s1, s2) (s2, s3) (s3, s4) (s4, s5) (s5, s6)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
*** Starting controller
```

Figure 16: Starting Mininet



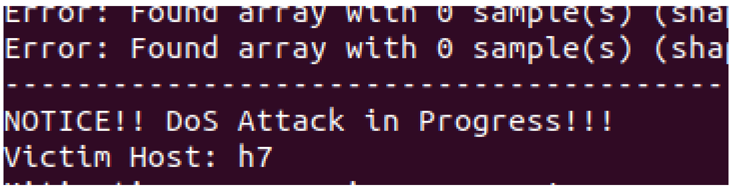
```
root@owais-VirtualBox:/home/owais/Desktop/DDoS-Detection-and-Mitigation/p1/mini
net# hping3 -2 -V -d 1200 -w 64 -p 80 --rand-source --flood 10.0.0.7
using h18-eth0, addr: 10.0.0.18, MTU: 1500
HPING 10.0.0.7 (h18-eth0 10.0.0.7): udp mode set, 28 headers + 1200 data bytes
hping in flood mode, no replies will be shown
```

Figure 17: Flooding host 7



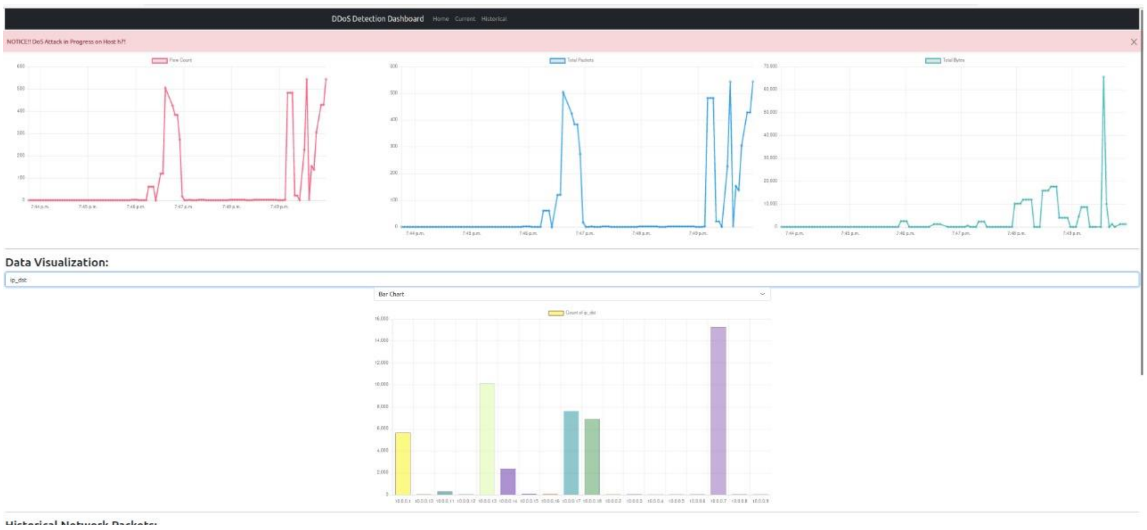
```
From 10.0.0.4 icmp_seq=71 Destination Host Unreachable
From 10.0.0.4 icmp_seq=72 Destination Host Unreachable
From 10.0.0.4 icmp_seq=73 Destination Host Unreachable
From 10.0.0.4 icmp_seq=74 Destination Host Unreachable
From 10.0.0.4 icmp_seq=75 Destination Host Unreachable
From 10.0.0.4 icmp_seq=76 Destination Host Unreachable
From 10.0.0.4 icmp_seq=77 Destination Host Unreachable
From 10.0.0.4 icmp_seq=78 Destination Host Unreachable
From 10.0.0.4 icmp_seq=79 Destination Host Unreachable
From 10.0.0.4 icmp_seq=80 Destination Host Unreachable
From 10.0.0.4 icmp_seq=81 Destination Host Unreachable
From 10.0.0.4 icmp_seq=82 Destination Host Unreachable
From 10.0.0.4 icmp_seq=83 Destination Host Unreachable
From 10.0.0.4 icmp_seq=84 Destination Host Unreachable
From 10.0.0.4 icmp_seq=85 Destination Host Unreachable
From 10.0.0.4 icmp_seq=86 Destination Host Unreachable
From 10.0.0.4 icmp_seq=87 Destination Host Unreachable
From 10.0.0.4 icmp_seq=88 Destination Host Unreachable
From 10.0.0.4 icmp_seq=89 Destination Host Unreachable
From 10.0.0.4 icmp_seq=90 Destination Host Unreachable
From 10.0.0.4 icmp_seq=91 Destination Host Unreachable
From 10.0.0.4 icmp_seq=92 Destination Host Unreachable
From 10.0.0.4 icmp_seq=93 Destination Host Unreachable
```

Figure 18: Host 7 not reachable.



```
Error: Found array with 0 sample(s) (sha
Error: Found array with 0 sample(s) (sha
----------------------------------------
NOTICE!! DoS Attack in Progress!!!
Victim Host: h7
```

Figure 19: Dos Attack detected.

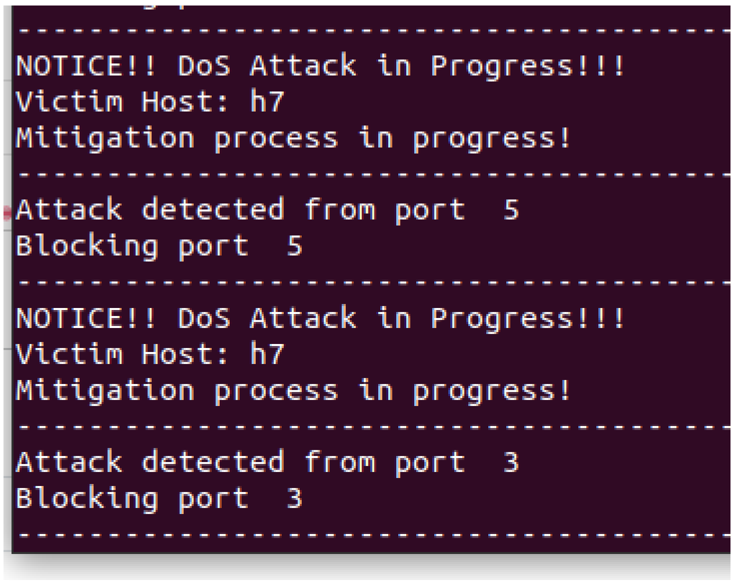


NOTICE!! DoS Attack in Progress on Host h7!

Figure 20: Dos Attack detected Alert

7.4. Mitigation Module

- **Technology Stack**: Python with integration into the Ryu controller for executing network commands.
- **Functionality**: Responds to detected threats by implementing predefined mitigation strategies.
- **Implementation Details**: In the development of our DDoS mitigation strategy, one of the key techniques involved the detection and isolation of the specific incoming port through which the DDoS attack was being perpetrated. By accurately identifying the compromised port, our system was able to implement a targeted timeout on this port, effectively halting the DDoS traffic without impacting other unaffected traffic channels. This selective approach not only minimized downtime for legitimate network users but also enhanced the overall security posture by swiftly neutralizing the threat at its source. The advantage of this method lies in its precision and efficiency. By focusing directly on the source of the attack, the system avoids the broader network disruptions typically associated with more general DDoS mitigation techniques, such as total bandwidth throttling or blanket IP blocking. Additionally, this targeted timeout approach conserves network resources, maintains optimal network performance for legitimate users, and significantly reduces the window of vulnerability during an attack, thereby enhancing both the responsiveness and resilience of the network infrastructure.



```
----------------------------------------
NOTICE!! DoS Attack in Progress!!!
Victim Host: h7
Mitigation process in progress!
----------------------------------------
Attack detected from port  5
Blocking port  5
----------------------------------------
NOTICE!! DoS Attack in Progress!!!
Victim Host: h7
Mitigation process in progress!
----------------------------------------
Attack detected from port  3
Blocking port  3
----------------------------------------
```

Figure 21: Mitigating by port blocking.

8. System Integration and Testing

8.1. Integration: All modules are integrated within a single application framework running atop the Ryu controller, facilitating direct communication between components and synchronous operations.

8.2. Testing Environment: Mininet is used to create a virtual network that mimics the deployment environment, enabling comprehensive testing of the system under controlled conditions. This setup allows for the simulation of both benign and DDoS traffic to test the system’s responsiveness and effectiveness.

Performance Metrics: The system’s performance is evaluated based on detection accuracy, mitigation effectiveness, and resource efficiency (CPU, memory usage). Automated scripts monitor these metrics during test scenarios to ensure the system meets predefined performance standards.

9. User Interface

- **Technology Stack**: React.js for the frontend, interfacing with a Flask-based API for backend communications.
- **Functionality**: Provides network administrators with a real-time dashboard displaying traffic statistics, alerts, and controls for adjusting detection and mitigation settings.
- **Implementation Details**: The front-end communicates with the backend via RESTful APIs, fetching updated data and sending user commands. The dashboard visualizes key metrics and allows administrators to configure thresholds and rules dynamically.

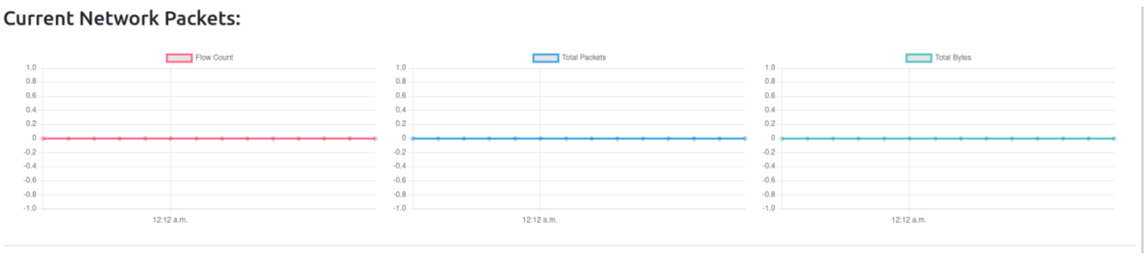

Current Network Packets:

Figure 22: No packets

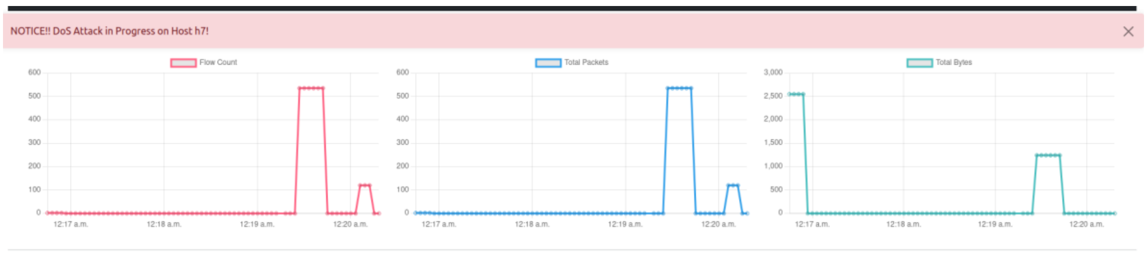

NOTICE!! DoS Attack in Progress on Host h7!

Figure 23: Incoming Packets

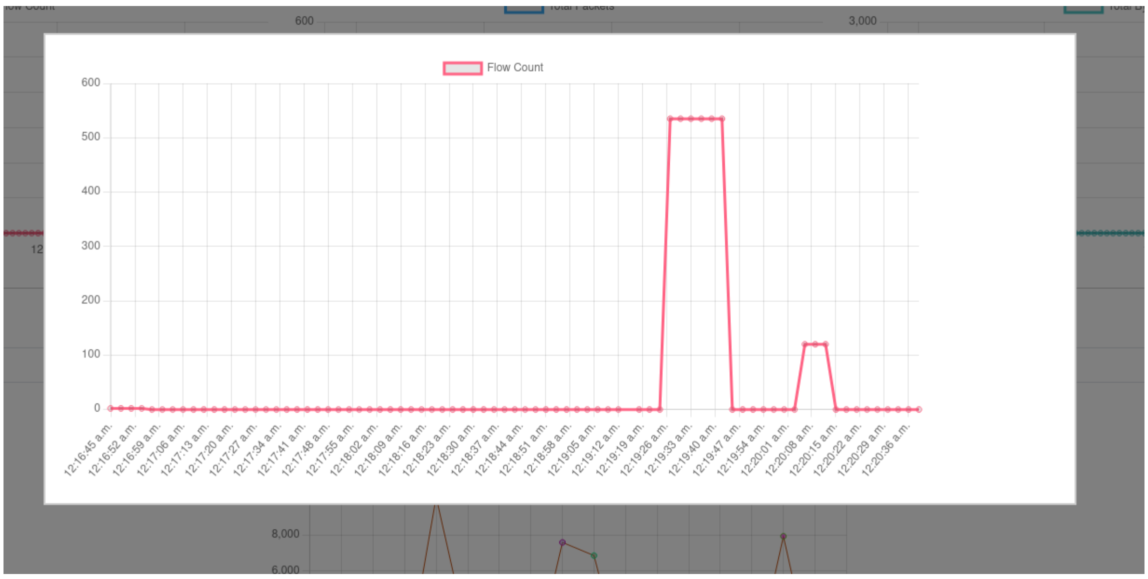


Figure 24: Flow Count

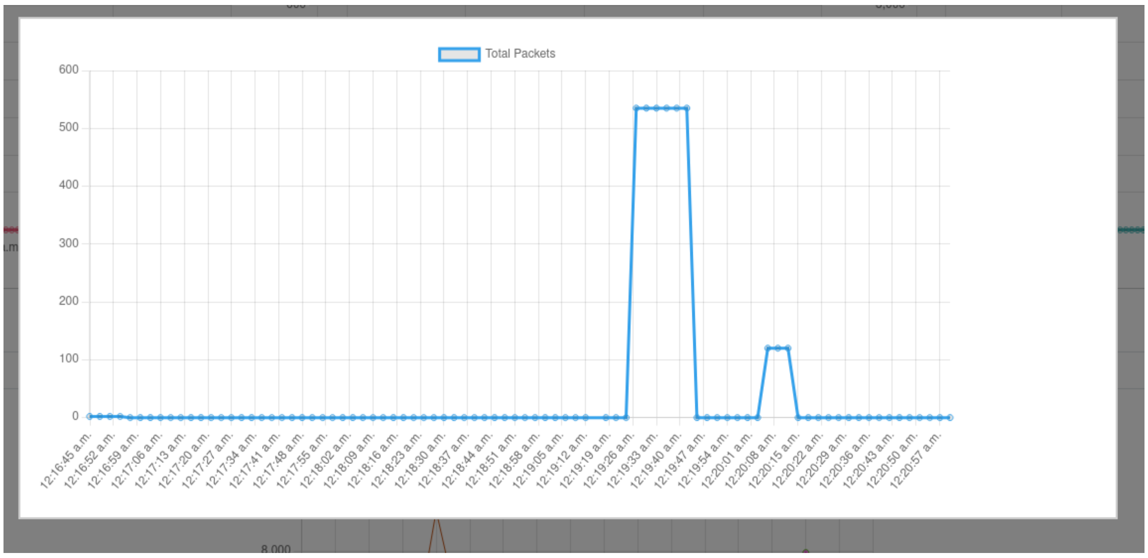


Figure 25: Total Packets

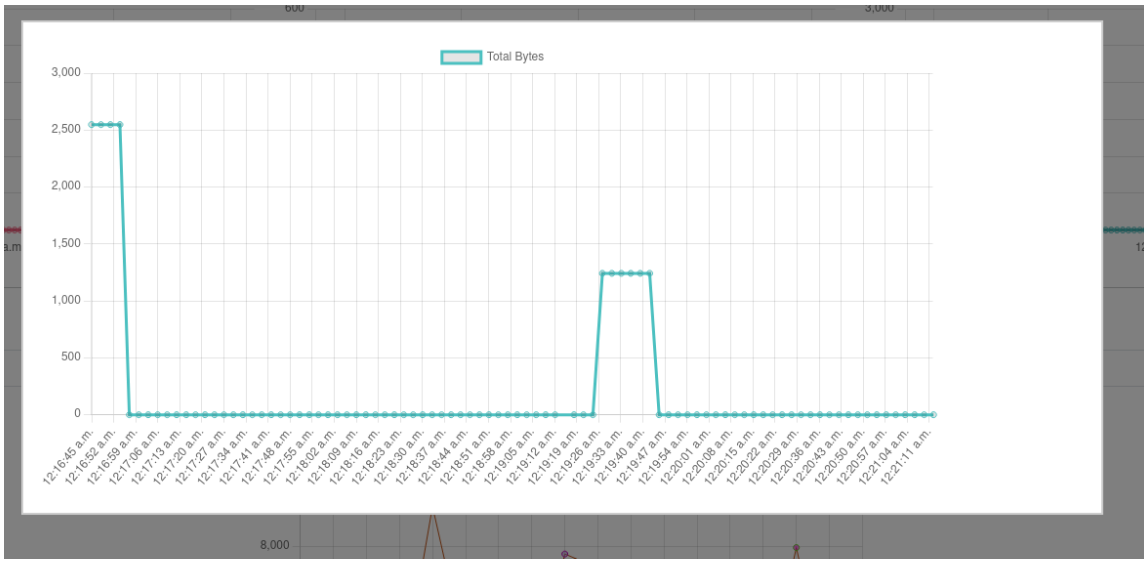
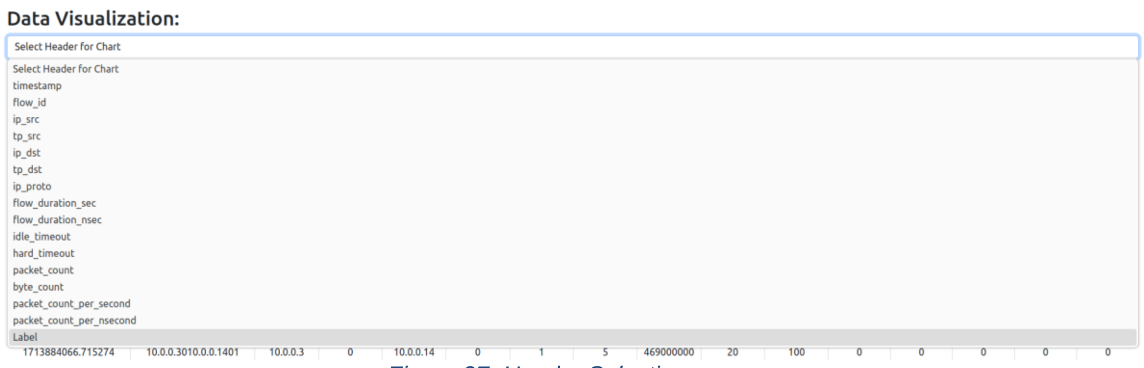


Figure 26: Total Bytes



Data Visualization:

Select Header for Chart

Select Header for Chart
timestamp
flow_id
ip_src
tp_src
ip_dst
tp_dst
ip_proto
flow_duration_sec
flow_duration_nsec
idle_timeout
hard_timeout
packet_count
byte_count
packet_count_per_second
packet_count_per_nsecond
Label

```
1713884066.715274   10.0.0.3010.0.0.1401   10.0.0.3   0   10.0.0.14   0   1   5   469000000   20   100   0   0   0   0   0
```

Figure 27: Header Selection

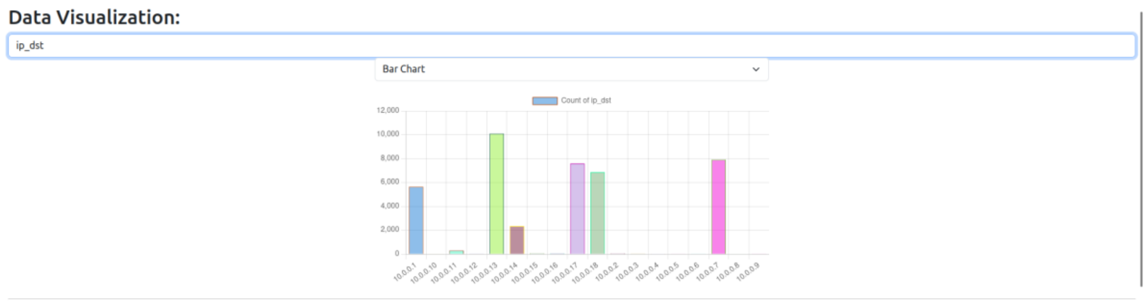


Figure 28: Bar Chart

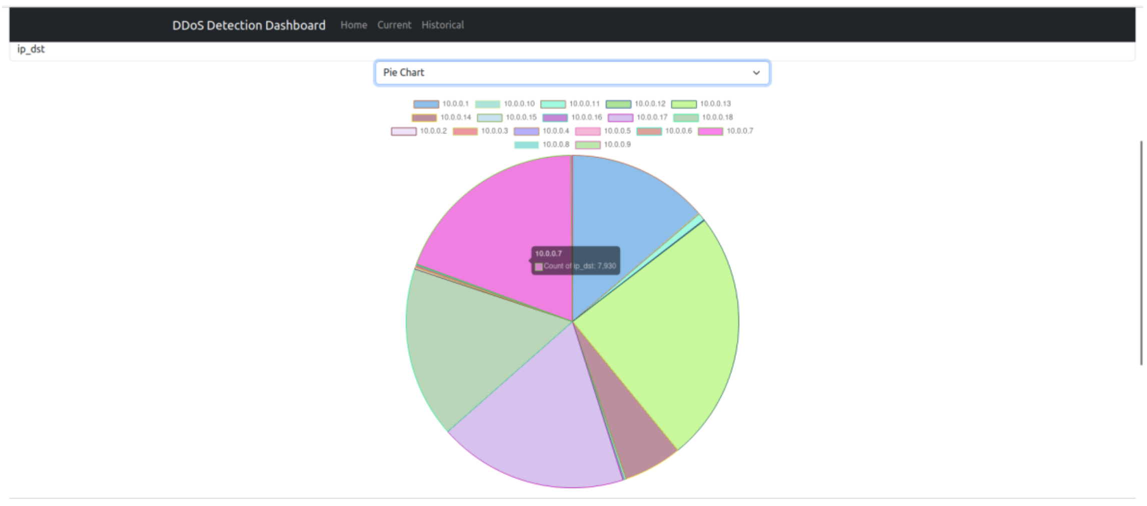


Figure 29: Pie Chart

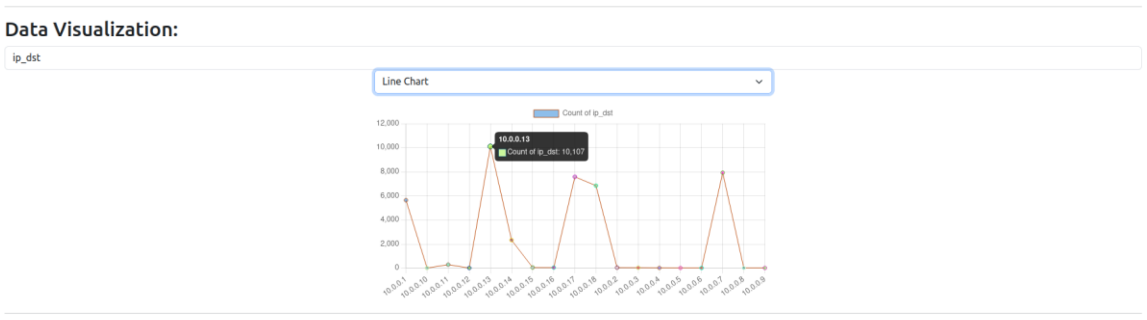


Figure 30: Line Chart

10. Challenges and Solutions

- **Data Handling**: Managing large volumes of traffic data efficiently was achieved through optimized data structures and processing algorithms.

- **Model Accuracy**: Multiple models were tested, and Naïve Bayes was selected due to its balance between detection accuracy and training time.

11. Testing and Evaluation

The testing and evaluation phase of the DDoS detection framework is crucial for validating the system's effectiveness in detecting and mitigating malicious network traffic. This phase involves:

Testing:

1. **Dataset Creation**:
    - **Traffic Generation**: Using Mininet, both benign and DDoS traffic scenarios were generated. Normal traffic included basic ping commands and HTTP requests, while DDoS traffic involved attack vectors like ICMP floods, SYN floods, UDP floods, and Smurf attacks. The hping3 tool was used to simulate these attacks.
    - **Traffic Collection**: The Ryu controller collected traffic data, storing it in a structured format for further analysis. Flow statistics were captured to provide features such as source/destination IPs, port numbers, protocol types, and packet/byte counts.
2. **Feature Engineering and Model Training**:
    - **Feature Extraction**: Features were extracted using custom Python scripts to compute additional metrics like packet count per second, flow duration, and average packet size. These features were crucial in distinguishing normal traffic from attacks.
    - **Model Training**: Naïve Bayes and Logistic Regression models were trained using the scikit-learn library. The dataset was split into training and test sets to validate the models' performance.
3. **Model Testing and Real-Time Analysis**:
    - **Offline Testing**: Models were first tested on the test dataset to assess accuracy, precision, recall, and F1 score. This ensured that models could effectively distinguish between benign and malicious traffic.
    - **Real-Time Testing**: The selected Naïve Bayes model was deployed on the Ryu controller to classify traffic in real-time. Generated traffic was then analyzed to determine if the model could accurately identify DDoS attacks.
4. **Mitigation Testing**:

- **Rule Implementation**: Custom mitigation rules were implemented to block or isolate malicious traffic based on model predictions.
- **Effectiveness Testing**: Various DDoS attack scenarios were simulated to evaluate how quickly and accurately the system could respond and mitigate attacks.

Evaluation:

1. **Accuracy Metrics**:
    - **Detection Accuracy**: The ability of the model to correctly identify benign and malicious traffic was measured. The Naïve Bayes model achieved an accuracy of 93.72%.
    - **False Positives/Negatives**: The rate of incorrectly classified traffic was measured to evaluate the model's reliability.
2. **Performance Metrics**:
    - **Execution Time**: The time taken to train and deploy the model was measured. Naïve Bayes had a training time of about 20 seconds, whereas Logistic Regression took 3-5 minutes.
    - **Resource Usage**: The framework’s impact on CPU, memory, and disk usage was monitored to ensure it remained lightweight and suitable for deployment in resource-constrained environments.
3. **Mitigation Effectiveness**:
    - **Response Time**: The time taken for the system to detect an attack and implement mitigation actions was measured.
    - **Traffic Impact**: The system's impact on legitimate traffic during attack scenarios was assessed to ensure that mitigation did not disrupt normal network operations.

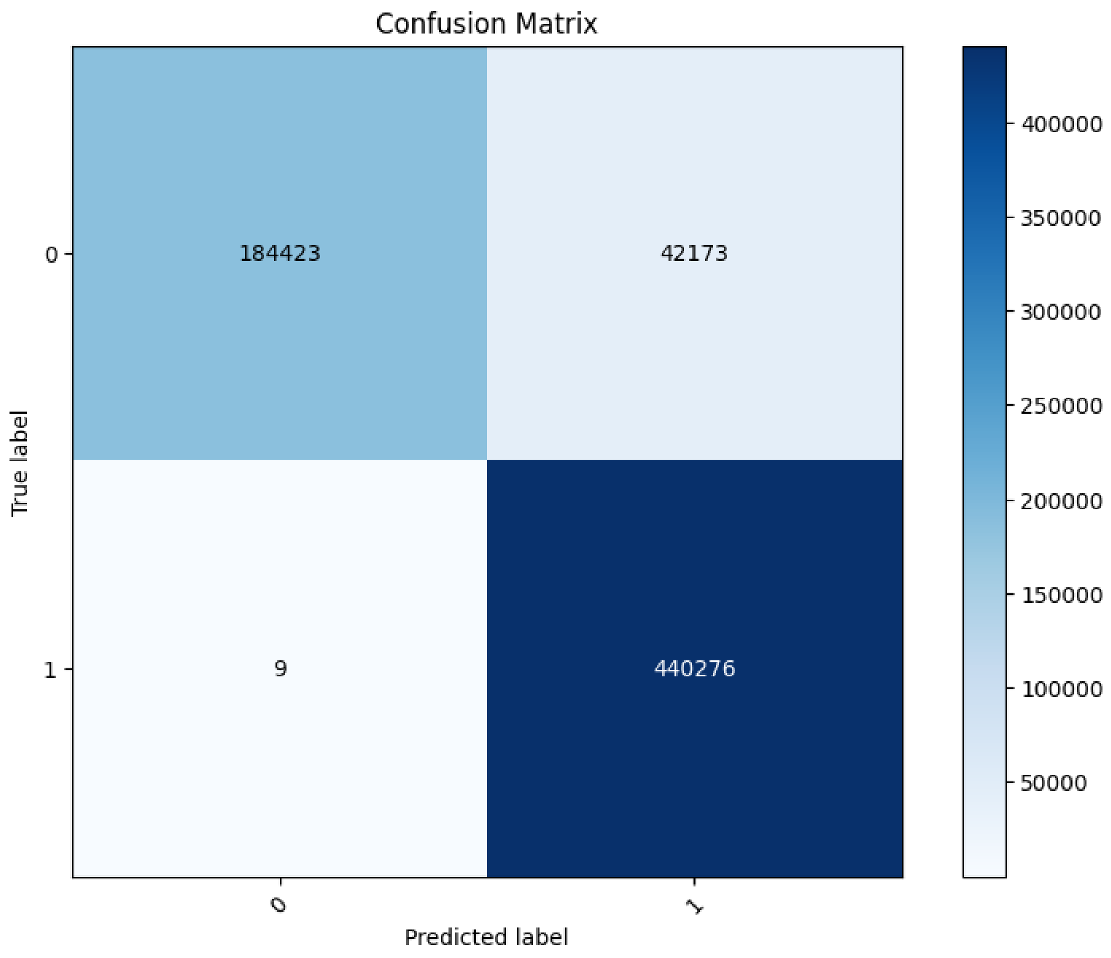


Figure 31: Confusion Matrix

Table 2: Summary of Model Performance

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Time to Detect**	**Peak Memory Usage (k-best)**	**Peak Memory Usage (w/o k-best)**
Naive Bayes	93.67%	0.91	0.99	0.95	8.37 sec	933.18 MB	1541.69 MB
Logistic Regression	97.08%	0.99	0.97	0.98	5.08 sec	812.23 MB	1340.10 MB

12. Conclusion

The designed DDoS detection and mitigation framework for Software-Defined Networking (SDN) environments successfully balances detection accuracy, response time, and resource efficiency. The comprehensive framework integrates several innovative elements to enhance network security in real-time:

1. **Framework Capabilities**: The framework leverages modular components for traffic collection, feature engineering, anomaly detection, and mitigation. This modularity

ensures adaptability to various network configurations and allows for future enhancements. The flow collector efficiently gathers traffic data, while the feature extender augments this data for better classification. The anomaly detection module uses machine learning to identify abnormal traffic patterns, and the mitigation module swiftly responds to mitigate attacks.

2. **Machine Learning Effectiveness**: The Naïve Bayes machine learning model provided effective DDoS detection with a 93.72% accuracy rate, balancing between accuracy and efficiency. The training time of under 20 seconds indicates that this approach can be rapidly deployed and updated, which is crucial for evolving threats.
3. **Mitigation Strategies**: The framework's mitigation module provides actionable responses to detected DDoS attacks, isolating and blocking malicious traffic in real-time. The flexibility in defining mitigation rules allows network administrators to customize responses based on the specific network environment.
4. **Performance and Resource Efficiency**: The framework demonstrated efficient usage of computational resources, ensuring that its deployment does not hinder normal network operations. The testing and evaluation phase confirmed that the framework could operate in a real-world network without significant latency or resource overhead.
5. **Comprehensive Testing**: The framework was tested in a simulated SDN environment using Mininet and the Ryu controller, ensuring realistic traffic conditions. The testing demonstrated the framework's ability to detect and mitigate a variety of DDoS attacks, providing practical validation of its effectiveness.

13. Future Work

While the framework demonstrates potential in mitigating DDoS attacks within SDN environments, several enhancements could further bolster its effectiveness:

- **Advanced Machine** Learning Models: Investigating more sophisticated machine learning and deep learning algorithms may improve the precision and reliability of attack detection.
- **Feature Expansion:** Adding a broader set of network metrics can enrich the analysis of traffic patterns, thereby boosting the framework’s capability to identify anomalies.
- **Real-World Deployment:** Deploying the framework in a live production environment will enable performance evaluation under authentic traffic conditions, offering valuable insights into its operational efficacy.

Final Thoughts

This framework marks a substantial advancement in network security for SDN contexts, delivering a potent means for detecting and counteracting DDoS threats. Its modular architecture, streamlined design, and integration of machine learning technologies render it an adaptable and powerful tool for network defense. With ongoing enhancements, this framework is poised to set a new benchmark in protecting SDN-based networks from DDoS dangers.

14. References

[1] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J, Shenker S, Turner J. OpenFlow: enabling innovation in campus networks. Proceedings of ACM SIGCOMM Communication Review 2008; 38(2): 69–74.

[2] Varghese, J. E., & Muniyal, B. (2021). An efficient ids framework for ddos attacks in SDN environment. IEEE Access, 9, 69680–69699. https://doi.org/10.1109/access.2021.3078065

[3] Q. Hu, S.-Y. Yu, and M. R. Asghar, 'Analysing performance issues of open-source intrusion detection systems in high-speed networks,' J. Inf. Secur. Appl., vol. 51, Apr. 2020, Art. no. 102426

[4] Q. Hu, M. R. Asghar, and N. Brownlee, 'Evaluating network intrusion detection systems for high-speed networks,' in Proc. 27th Int. Telecommun. Netw. Appl. Conf. (ITNAC), Nov. 2017, pp. 1–6

[5] S. Campbell and J. Lee, ''Intrusion detection at 100G,' in Proc. State Pract. Rep. (SC), 2011, pp.1–9.

[6] J. Yang, L. Jiang, X. Bai, H. Peng, and Q. Dai, ''A high-performance roundrobin regular expression matching architecture based on FPGA,'' in Proc. IEEE Symp. Comput. Commun. (ISCC), Jun. 2018, pp. 1-7.

[7] L. Schaelicke and J. C. Freeland, ''Characterizing sources and remedies for packet loss in network intrusion detection systems,' in Proc. IEEE Int. Workload Characterization Symp., Oct. 2005, pp. 188–196.

[8] T. H. Ptacek and T. N. Newsham, ''Insertion, evasion, and denial of service: Eluding network intrusion detection,' Tech. Rep. ADA391565, 1998.

[9] M. Arafune, B. Goswami, M. Kulkarni, N. Venkatachalam, and S. Asadollahi, "Lightweight Anti DDoS Security Tool: Edge Level Filtering in SDN using P4," 2023 Fifth International Conference on Electrical, Computer and Communication Technologies (ICECCT), Erode, India, 2023, pp. 1-7, doi: 10.1109/ICECCT56650.2023.10179747.

[10] DPDK Testpmd. Accessed: Nov. 7, 2018. [Online]. Available: https://github.com/ceph/dpdk/blob/master/app/test-pmd/testpmd.c

[11] M. J. Shayegan and A. Damghanian, "A Review of Methods to Prevent DDOS Attacks Using NFV and SDN," 2023 9th International Conference on Web Research (ICWR), Tehran, Iran, Islamic Republic of, 2023, pp. 346-355, doi: 10.1109/ICWR57742.2023.10139112.

[12] Kishiyama, Brian and Guerrero, Jesus and Alsmadi, Izzat, Security Policies Automation in Software Defined Networking (March 11, 2023). Available at SSRN: https://ssrn.com/abstract=4384690 or http://dx.doi.org/10.2139/ssrn.4384690

[13] F. Khashab, J. Moubarak, A. Feghali and C. Bassil, "DDoS Attack Detection and Mitigation in SDN using Machine Learning," 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), Tokyo, Japan, 2021, pp. 395-401, doi: 10.1109/NetSoft51509.2021.9492558.

[14] Jagdeep Singh, Sunny Behal, Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions, Computer Science Review, Volume 37, 2020, 100279, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2020.100279.

[15] S. Kautish, R. A and A. Vidyarthi, "SDMTA: Attack Detection and Mitigation Mechanism for DDoS Vulnerabilities in Hybrid Cloud Environment," in IEEE Transactions on Industrial Informatics, vol. 18, no. 9, pp. 6455-6463, Sept. 2022, doi:

10.1109/TII.2022.3146290.

[16] Gadze, James Dzisi, Akua Acheampomaa Bamfo-Asante, Justice Owusu Agyemang, Henry Nunoo-Mensah, and Kwasi Adu-Boahen Opare. "An investigation into the application of deep learning in the detection and mitigation of DDOS attack on SDN controllers." Technologies 9, no. 1 (2021):

[17] Al-Duwairi, B., Al-Kahla, W., AlRefai, M.A., Abedalqader, Y., Rawash, A. and Fahmawi, R., 2020. SIEM-based detection and mitigation of IoT-botnet DDoS attacks. International Journal of Electrical and Computer Engineering, 10(2), p.2182.

[18] M. Tayyab, B. Belaton and M. Anbar, "ICMPv6-Based DoS and DDoS Attacks Detection Using Machine Learning Techniques, Open Challenges, and Blockchain Applicability: A Review," in IEEE Access, vol. 8, pp. 170529-170547, 2020, doi: 10.1109/ACCESS.2020.3022963.

[19] A. B. Dehkordi, M. Soltanaghaei and F. Z. Boroujeni, "The DDoS attacks detection through machine learning and statistical methods in SDN," The Journal of Supercomputing, vol. 77, no. 3, pp. 2383–2415, 2021.

[20] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang et al., "A new framework for DDoS attack detection and defense in SDN environment," IEEE Access, vol. 8, pp. 161908–161919, 2020.

[21] J. Ye, X. Cheng, J. Zhu, L. Feng and L. Song, "A DDoS attack detection method based on SVM in software defined network," Security and Communication Networks, vol. 2018, no. 4, pp. 11–23, 2018.

[22] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy et al., "An evolutionary SVM model for DDOS attack detection in software defined networks," IEEE Access, vol. 8, pp. 132502–132513, 2020.

[23] J. A. P. Díaz, I. A. Valdovinos, K. K. R. Choo and D. Zhu, "A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning," IEEE Access, vol. 8, pp.155859–155872, 2020.

[24] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," Computer Networks, vol. 62, no. 2, pp. 122–136, 2014.

[25] L. Schehlmann and H. Baier, "COFFEE: A concept based on openflow to filter and erase events of botnet activity at high-speed nodes, information. 2013-informatik angepasst an mensch," Organization und Umwelt, vol. 220, pp. 2225–2239, 2013.

[26] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in software defined networks using machine learning techniques," in National Software Engineering Conf., Rawalpindi, Pakistan, pp. 55–60,2014.

[27] R. Braga, E. Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/openFlow," in IEEE Local Computer Network Conf., Denver, USA, pp. 408–415, 2010.

[28] 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.

[29] W. Yassin, N. I. Udzir, Z. Muda and M. N. Sulaiman, "Anomaly-based intrusion detection through k-means clustering and naives bayes classification," in 4th Int. Conf. Computer Informatics, Sarawak, Malaysia, vol. 49, pp. 298–303, 2013. CMC, 2022, vol.71, no.1 789

[30] A. Saied, R. E. Overill and T. Radzik, "Detection of known and unknown DDoS

attacks using artificial neural networks," Neurocomputing, vol. 172, no. 7, pp. 385–393, 2016.

[31] M. Wang, Y. Lu and J. Qin, "A dynamic MLP-based DDoS attack detection method using feature selection and feedback," Computer Security, vol. 88, no. 7, pp. 101645–101658, 2020.

[32] M. P. Singh and A. Bhandari, "New flow-based DDoS attacks in SDN: Taxonomy, rationales, and research challenges," Computer Communications, vol. 154, no. 2, pp. 509–527, 2020.

[33] X. D. Zang, J. Gong and X. Y. Hu, "An adaptive profile-based approach for detecting anomalous traffic in backbone," IEEE Access, vol. 7, pp. 56920–56934, 2019.

[34] Dataset: https://www.kaggle.com/datasets/devendra416/ddos-datasets.