TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

LALITPUR ENGINEERING COLLEGE

PROJECT DEFENSE ON

"DDOS ATTACK DETECTION AND MITIGATION USING MACHINE LEARNING IN MULTICONTROLLER SDN ENVIRONMENT"

Presenter

ANSHUL RAWAL[LEC077BCT002]

KAPIL PARAJULI[LEC077BCT010]

SHIVAM GUPTA[LEC077BCT022]

SAIROJ PRASAI[LEC077BCT030]

Under The Supervision of Er. Binod Sapkota

December 27, 2024

OUTLINES

- Introduction
- Problem Statement
- Project Objective
- Scope of Project
- Potential Project Application
- Originality of Project
- Literature Review
- Feasibility Study
- Methodology
- Conclusion
- Discussion
- References

INTRODUCTION

- **Objective**: Develop a machine learning-based scheme to detect and mitigate DDoS attacks in SDN.
- **Problem**: Traditional detection techniques are ineffective in SDN environments against DDoS attacks that overload systems with excessive traffic.
- **Solution**: Leverage SDN's programmability and centralized control to address volume-based and application-based DDoS attacks, such as TCP SYN flood, Ping flood, and Slow HTTP attacks.
- **Evaluation**: Implement the solution on the RYU controller and evaluate performance using Mininet emulation.

PROBLEM STATEMENT

- Traditional security measures struggle to counter increasingly sophisticated DDoS threats.
- Modern DDoS attacks inflict severe operational, financial, and reputational damage on individuals, businesses, and governments.
- High data volume and diversity result in elevated false positive rates, straining detection algorithms and delaying responses.
- Rapid and effective mitigation techniques are crucial to safeguard network availability, confidentiality, and integrity.

PROJECT OBJECTIVE

- To create a balanced multi-controller SDN environment, and to train and deploy an Machine Learning model for DDoS attack detection and mitigation, and test its effectiveness in the control plane.

PROJECT SCOPE

- The system involves continuous monitoring and analysis of network traffic to identify and address potential security threats.
- It detects deviations from typical traffic patterns to highlight suspicious activities that may indicate DDoS attacks.
- Advanced technologies like machine learning and Software-Defined Networking (SDN) enhance the system’s detection capabilities.
- Ensures robust protection against emerging DDoS threats while maintaining network integrity and availability.

POTENTIAL PROJECT APPLICATION

- Corporate Networks
- Protection of Critical Infrastructure
- Cloud Security
- Government and Military Networks
- Financial Services

ORIGINALITY OF PROJECT

- Detection Algorithms
- Secure Monitoring.
- Behavioral Analysis.
- Integration with SDN.
- Anomaly Detection.
- Multicontroller.

LITERATURE REVIEW

- Banitalebi Dehkordi, Soltanaghaei, and Boroujeni explore the detection of Distributed Denial of Service (DDoS) attacks through the application of machine learning and statistical methods in Software-Defined Networking (SDN) in their work published in the Journal of Supercomputing in 2020.
- The paper titled "The Beacon OpenFlow controller" by D. Erickson, presented at HotSDN 2013, provides insights into the Beacon OpenFlow controller, a software component central to Software-Defined Networking (SDN).

LITERATURE REVIEW

- Fatih, Cengiz, and Enis examine the utilization of machine learning algorithms for flow-based anomaly detection systems in Software Defined Networks (SDNs) in their contribution to "Intelligent and Fuzzy Techniques: Smart and innovative Solutions" in 2021.
- The paper *“Toward Network-Based DDoS Detection in Software-Defined Networks”* (2018) explores DDoS detection challenges in SDNs, emphasizing the limitations of traditional methods. It highlights SDNs' unique features, like centralized control and programmable data planes, and their role in enhancing detection strategies.

FEASIBILITY STUDY[1]

1. **Technical Feasibility**
   - Utilization of advanced algorithms and machine learning for accurate DDoS detection.
   - Compatibility with existing network hardware and software infrastructure.
   - Use of GPU provided by google collab or kaggle to train the Machine Learning model.

FEASIBILITY STUDY[2]

2. **Operational Feasibility**
   - Integration with existing network infrastructure for seamless implementation.
   - Scalable architecture to accommodate evolving network demands and threats.
   - Compatibility with diverse network environments for widespread applicability.

FEASIBILITY STUDY[3]

3. Economic Feasibility

- Consideration of Initial cost to integrate DDoS Detection system in the network.
- Effective Resource Utilization as GPU from google collab and kaggle providing cost effectiveness
- Long-term cost savings through prevention of costly security incidents.

METHODOLOGY – THEORETICAL BACKGROUND

- A Distributed Denial of Service (DDoS) detection and mitigation system is essential in cybersecurity, aiming to detect and prevent efforts to overload a network, service, or application with excessive malicious traffic.
- Machine learning improves DDoS detection by analyzing historical data to spot abnormal traffic patterns, enabling identification of both known and new attack methods.
- It can also quickly analyze lots of network data to find problems and react fast to stop them.

METHODOLOGY – THEORETICAL BACKGROUND

- SDN (Software-Defined Networking) is a network architecture that decouples the control plane from the data plane, enabling programmable network management.
- It uses software to control and optimize how data flows throughout the network.
- SDN simplifies network adjustments and optimizations compared to traditional hardware-based approaches.
- The architecture allows efficient resource management and dynamic traffic handling through centralized control mechanisms.

METHODOLOGY – THEORETICAL BACKGROUND

Data Plane

- Responsible for forwarding and processing data packets between devices.
- Handles tasks like routing, switching, and enforcing network policies.

Application Plane

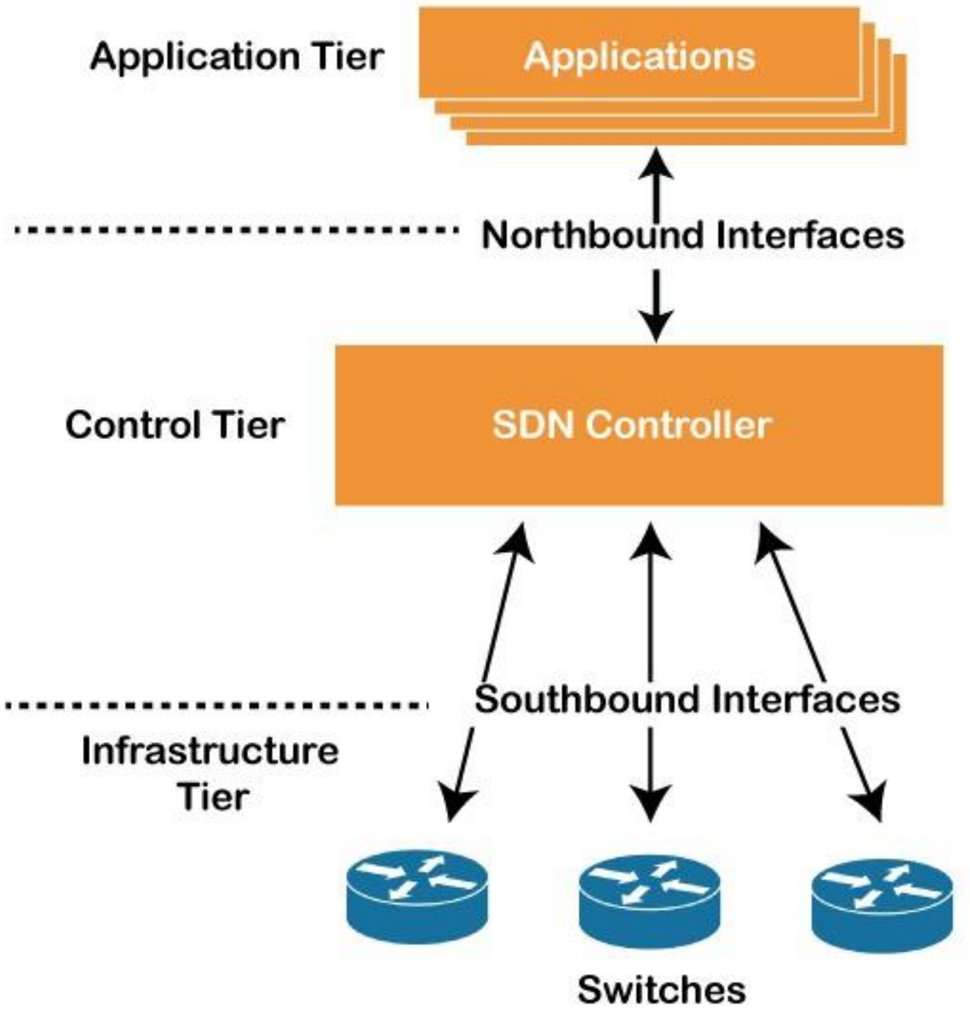
- Hosts software applications that define and implement high-level network services.
- They are connected and controlled by the Controller through a Rest API.

METHODOLOGY – THEORETICAL BACKGROUND

Control Plane

- This layer is considered the brain of the entire system.
- Communicates with the data plane to enforce policies, manage routing, and optimize network performance.
- Manages and decides how data flows through the network.
- RYU controller is used.
- The detection and mitigation is integrated in this plane.

METHODOLOGY – THEORETICAL BACKGROUND



METHODOLOGY – SOURCES OF DATA

- The dataset was generated in a controlled environment using locally available resources.
- A Mininet network with multiple controllers and custom topologies was configured to simulate traffic.
- Both normal network activities and various DDoS attack scenarios were simulated to create the dataset.

METHODOLOGY – SOURCES OF DATA

Some of key attributes in Normal Traffic Dataset:

- **Source Host:** IP address of the host initiating the traffic (e.g., h1, h2, etc.).
- **Destination Host:** IP address of the receiving host (e.g., h1, h2, etc.).
- **Traffic Type:** Type of traffic being generated (e.g., Web Traffic, ICMP, TCP, UDP).
- **Protocol:** The protocol used for communication (e.g., HTTP, ICMP, TCP, UDP).
- **Source Port:** Port number on the source host used for communication.
- **Destination Port:** Port number on the destination host used for communication.

METHODOLOGY – SOURCES OF DATA

- **Packet Size:** Size of the packets being sent (in bytes).
- **Traffic Volume:** Total amount of data sent during the simulation (in bytes or packets).
- **Time Stamp:** Timestamp of when each traffic flow or packet was transmitted.
- **Request Type:** Type of request sent (e.g., HTTP GET/POST, ICMP Echo Request).
- **Download File:** File name or size if the traffic involves file download (e.g., index.html, test.zip).

METHODOLOGY – SOURCES OF DATA

DDoS Dataset Attributes:

- **Source Host (Attacker):** IP address of the host executing the attack.
- **Target Host (Victim):** IP address of the victim host under attack (e.g., h1).
- **Attack Type:** Type of DDoS attack being simulated (e.g., ICMP Flood, UDP Flood, TCP-SYN Flood).
- **Packet Type:** Type of packet used in the attack (e.g., ICMP Echo, UDP Datagram, SYN Packet).

METHODOLOGY – SOURCES OF DATA

- **Source Port:** Port number of the attacker host used in the attack.
- **Destination Port:** Port number on the victim host targeted by the attack.
- **Packet Size:** Size of the packets sent during the attack (in bytes).
- **Traffic Volume:** Total volume of malicious traffic sent during the attack (in bytes or packets).
- **Attack Duration:** Duration of the DDoS attack (in seconds or time window).

METHODOLOGY – SOURCES OF DATA

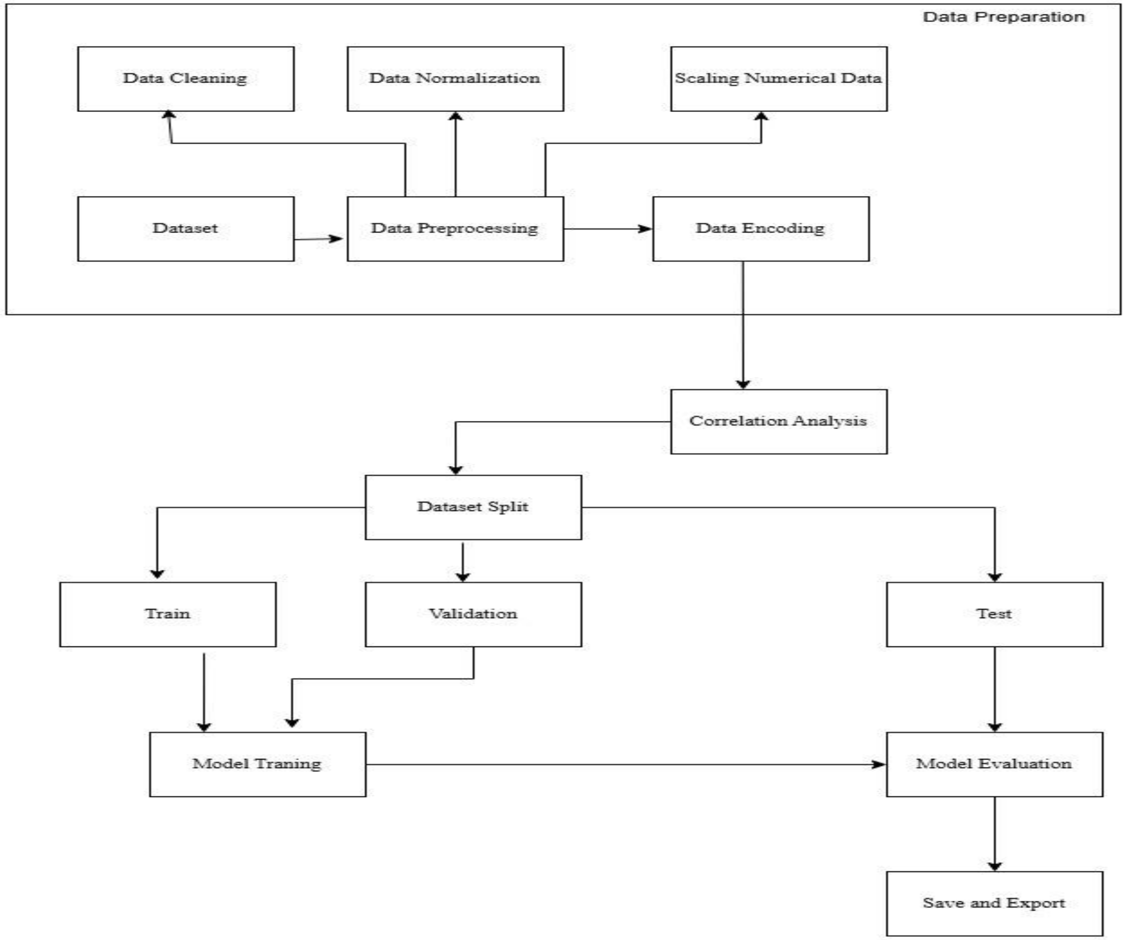
- **Frequency:** Rate of packet transmission (e.g., packets per second).
- **Flood Intensity:** Intensity or frequency of packets being sent in a flood (e.g., low, medium, high).
- **Time Stamp:** Timestamp for each packet sent during the DDoS attack.

METHODOLOGY – INSTRUMENTATION REQUIREMENT

- Hardware Tools:
  - Personal laptops – for development
  - GPU and CPU from Kaggle for model training

- Software Tools:
  - Python
  - RYU Controller
  - Mininet
  - Wireshark
  - Tensorflow

METHODOLOGY – MODEL TRAIN BLOCK DIAGRAM



METHODOLOGY – WORKING PRINCIPLE

Data Preprocessing

- **Data Cleaning:** Removed missing, NAN, and infinite values, along with redundant data that did not contribute to analysis, ensuring better data quality.
- **Data Transformation:** Normalized timestamps and standardized features by scaling to unit variance for consistent contribution in modeling.
- **Data Encoding:** Applied label encoding for binary categories and transformed IP addresses into numerical format for model compatibility.
- **Correlation and Feature Extraction:** Analysed correlations to remove highly correlated features, reducing redundancy and improving model performance.

METHODOLOGY – WORKING PRINCIPLE

- XGBoost is an advanced machine learning algorithm for classification and regression, based on gradient tree boosting techniques.
- XGBoost uses a regularized objective function to balance model accuracy and complexity, preventing overfitting. The function is:

$$L(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), \quad \text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

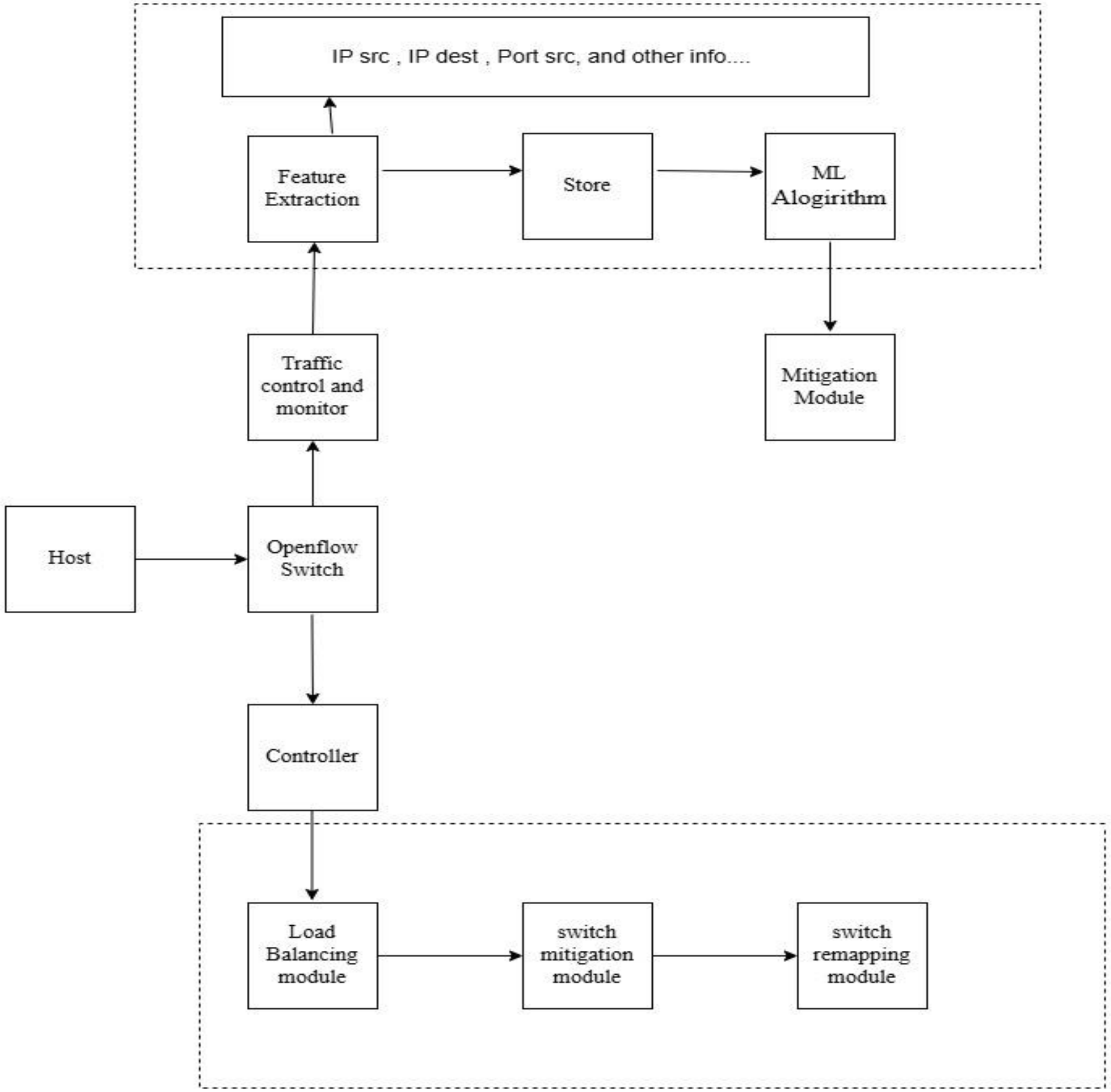
- XGBoost adds decision trees sequentially to minimize the objective function, with each tree improving predictions based on previous residual errors. The update is

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

METHODOLOGY – WORKING PRINCIPLE

- XGBoost includes features like shrinkage (learning rate adjustment) and column subsampling to prevent overfitting and improve computational efficiency.
- XGBoost leverages parallel and distributed computing to handle large-scale datasets efficiently, making it ideal for complex machine learning problems.
- XGBoost efficiently handles sparse data by using techniques like missing value imputation and sparse matrix representation, ensuring high performance.

SYSTEM BLOCK DIAGRAM



IMPLEMENTATION

Network Topology

- A custom SDN topology was developed with six switches (s1 to s6) and ten hosts.
- Hosts were assigned IP addresses in the 10.0.0.x/24 range for proper network segmentation.
- The topology featured a linear chain design with direct links between adjacent switches.
- This setup mimics real-world hierarchical networks with vertical traffic flow.
- Three remote controllers (c1, c2, and c3) were implemented, operating on ports 6653, 6654, and 6655.
- Controller assignments ensured load balancing and efficient traffic distribution:

  Controller 1 managed Switches 1 and 2.

  Controller 2 managed Switches 3 and 4.

  Controller 3 managed Switches 5 and 6.

IMPLEMENTATION

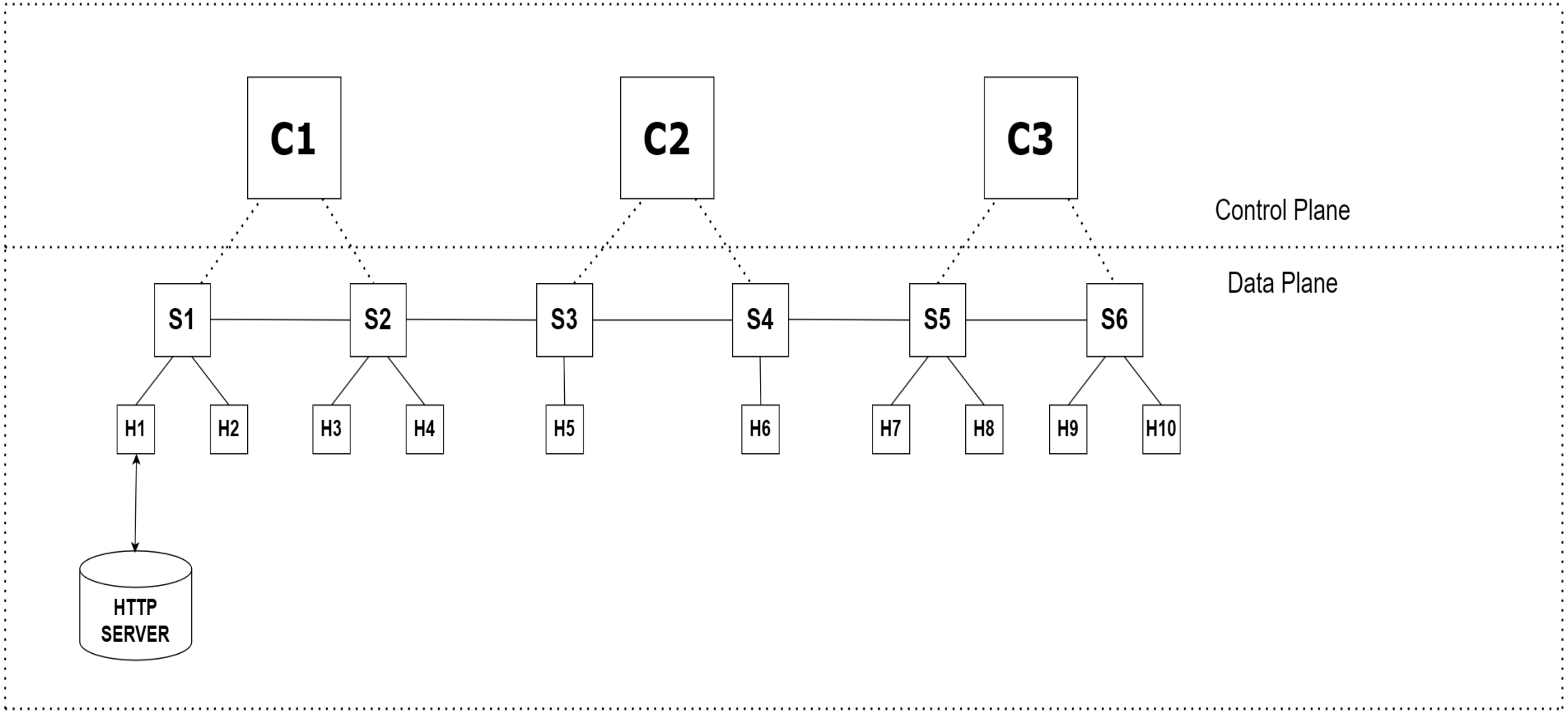


Fig : : Network Topology

IMPLEMENTATION

Traffic Control Module

- The Flow Collector initiates the process by requesting traffic data from the controller.
- The controller uses the OpenFlow protocol to send flow-stats requests to each switch, asking for flow statistics.
- Each switch responds by sending a flow-stats reply message to the controller, which contains all flow entries from its flow tables.
- The controller collects the flow-stats reply messages from all switches and compiles the traffic data for the Flow Collector.

IMPLEMENTATION

Detection Module

- The Attack Detection Module analyzes the statistical information from the flow tables using a trained Machine Learning (ML) model to detect network anomalies.
- If the traffic flow is deemed normal, the packets continue to enter the system without interruption.
- If the traffic flow is predicted as dangerous, the Attack Detection Module forwards the information about that flow to the Mitigation Module for further action.

IMPLEMENTATION

Mitigation Module

- When hazardous traffic is identified by the detection module, the Mitigation Module requests the controller to block traffic flows from the corrupted switch and port.
- For example, if switch-id-1 and port 1 are receiving hazardous traffic, the Mitigation Module instructs the controller to remove these traffic flows from port 1.
- At the same time, the Mitigation Module sends continuous alerts to the administrator for ongoing system monitoring.
- Once the attack flow ends, the system will automatically re-establish the connection to port 1 to resume normal operation.

IMPLEMENTATION

Verification and Validation

- Accuracy, Precision, and Recall are metrics used to evaluate the performance of a classification model.
- Accuracy is the ratio of correctly predicted instances to the total instances.
- Precision is the ratio of correctly predicted positive instances to the total predicted positive instances.
- Recall is the ratio of correctly predicted positive instances to the total actual positives

RESULTS

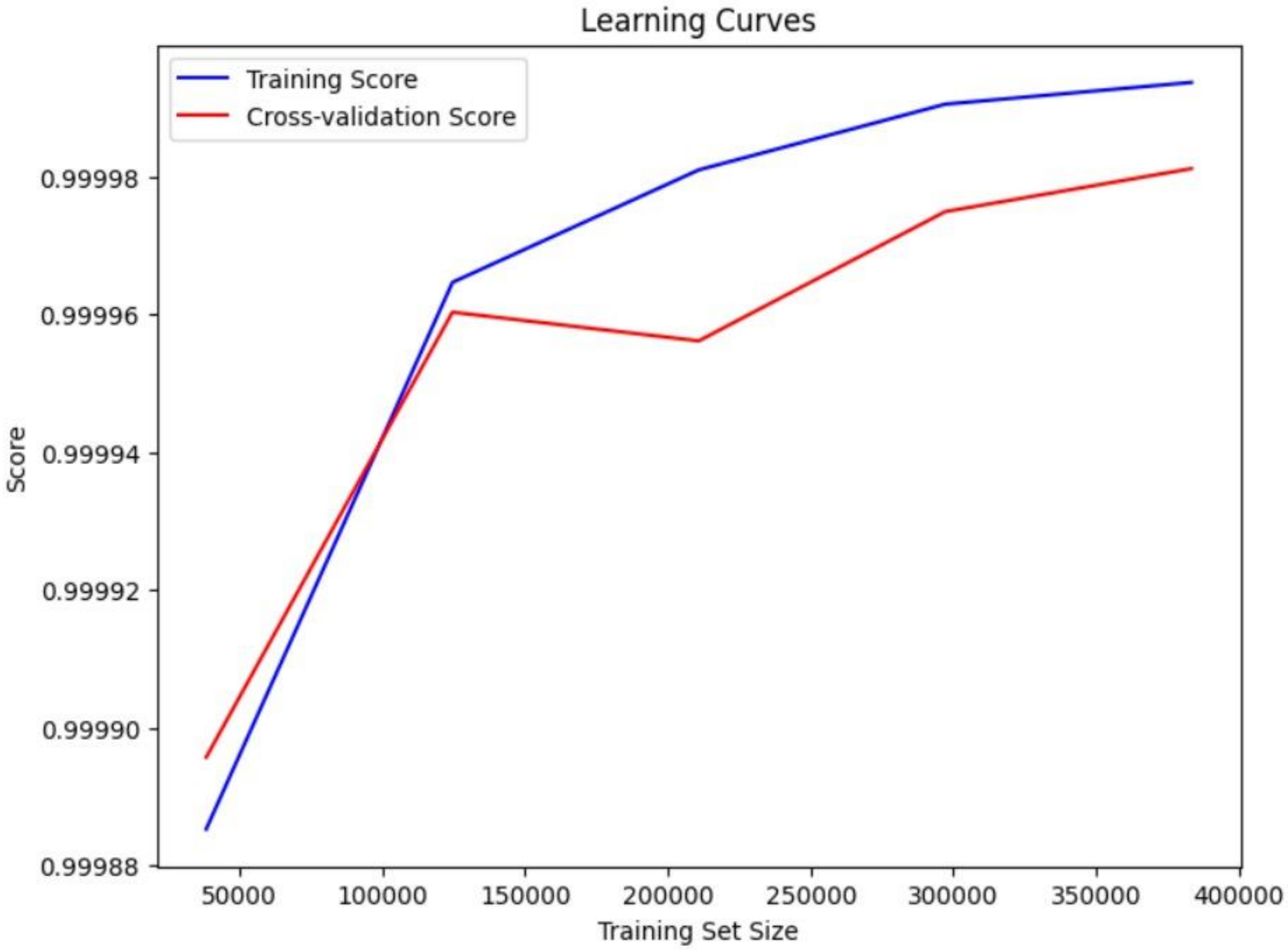


Fig: : Learning Curve

RESULTS

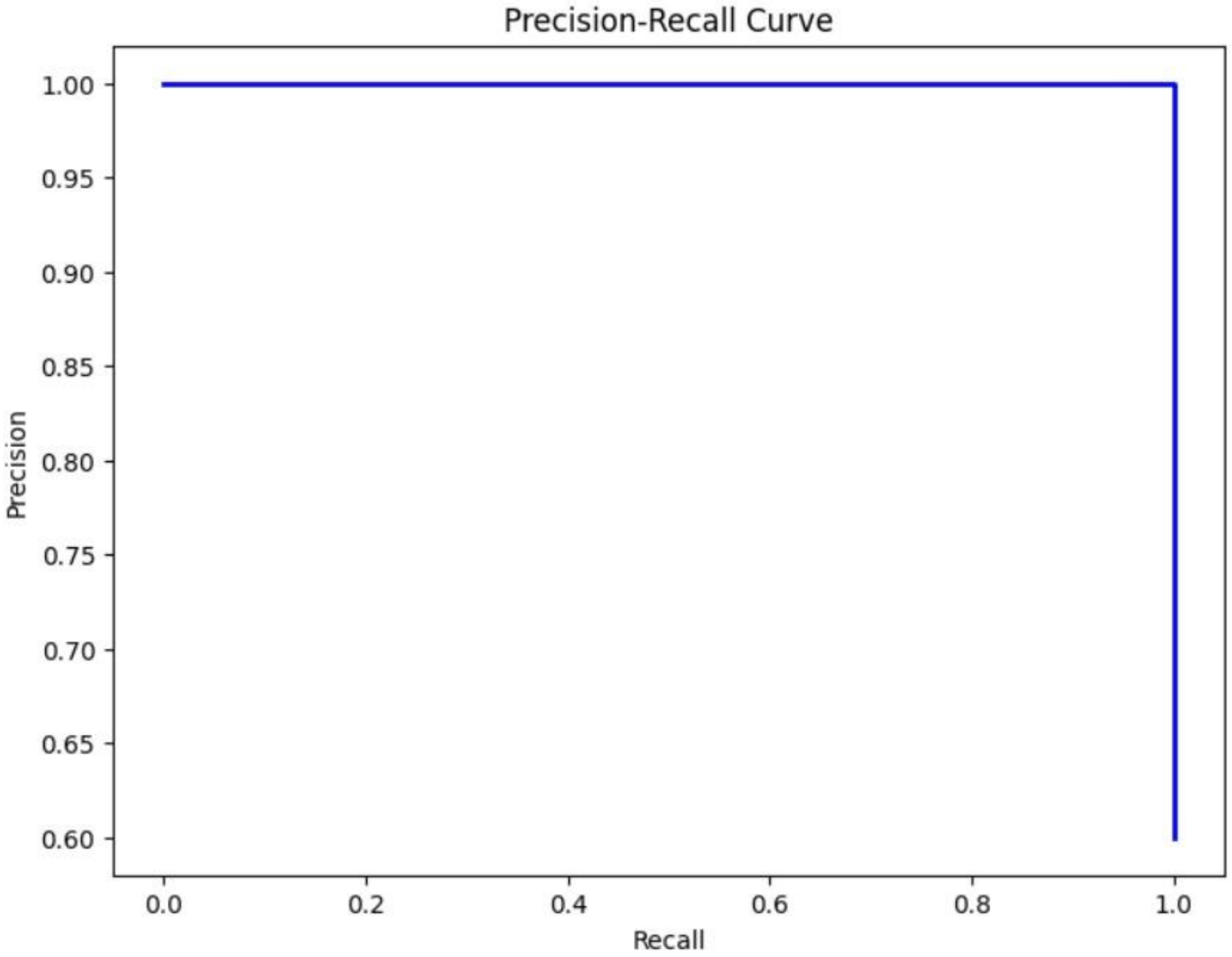


Fig: recall Scores v/s precision

RESULTS

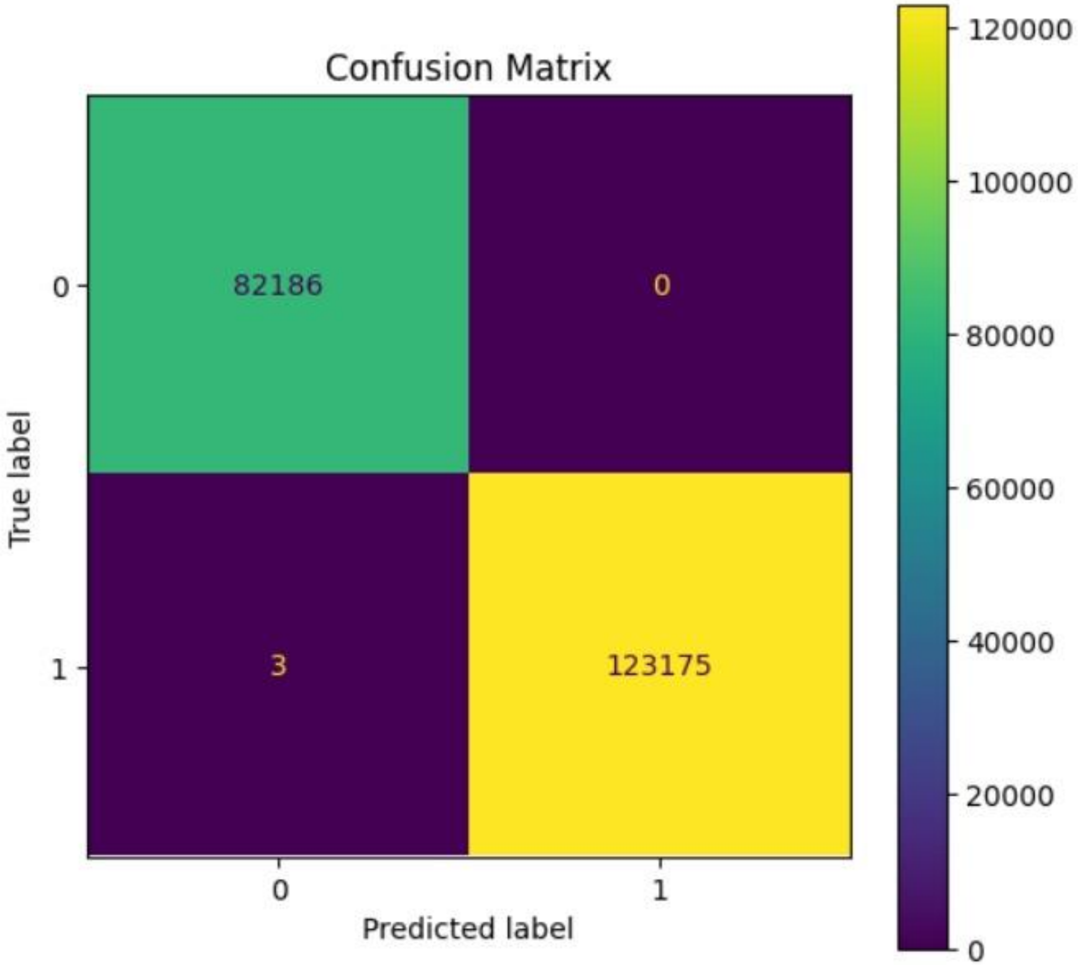
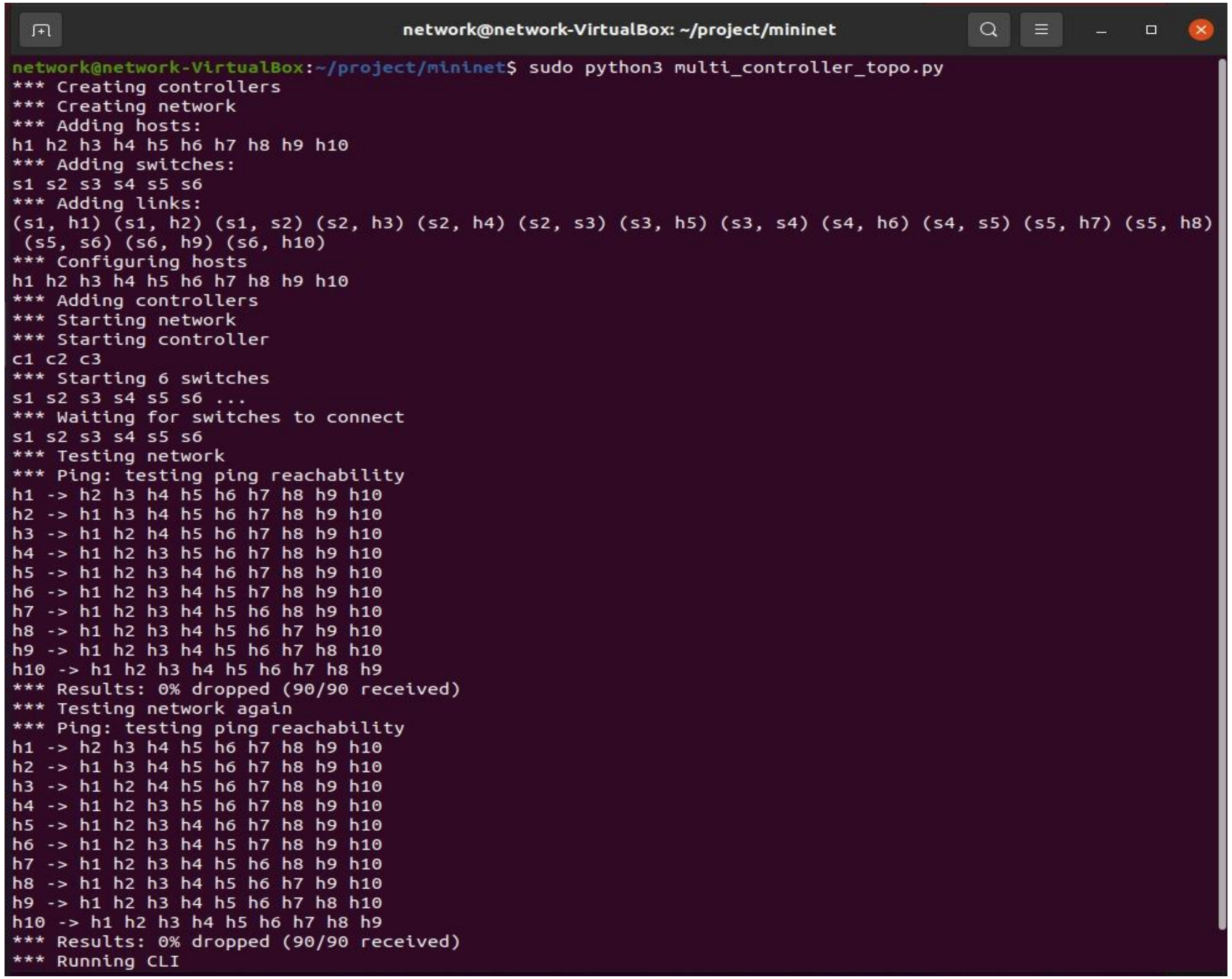


Fig: Confusion Matrix

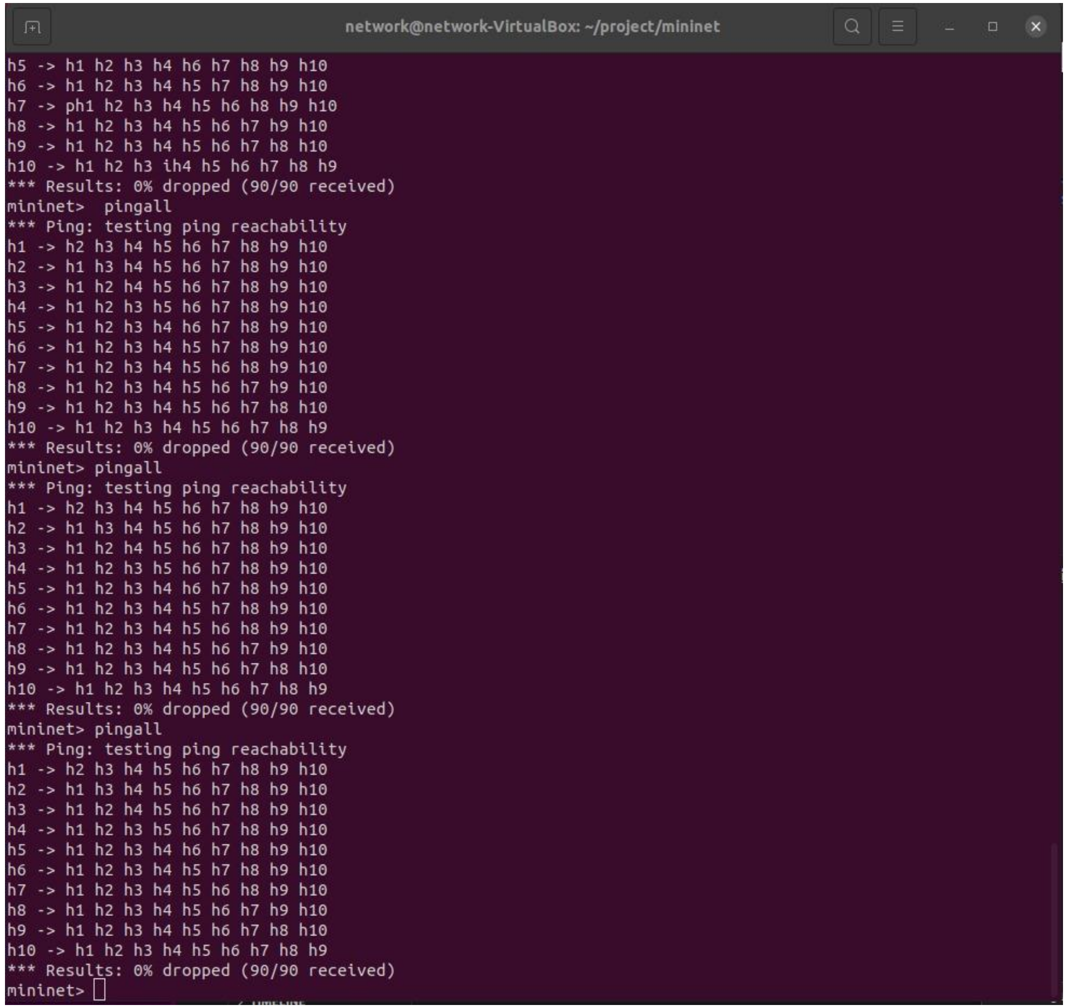
RESULTS



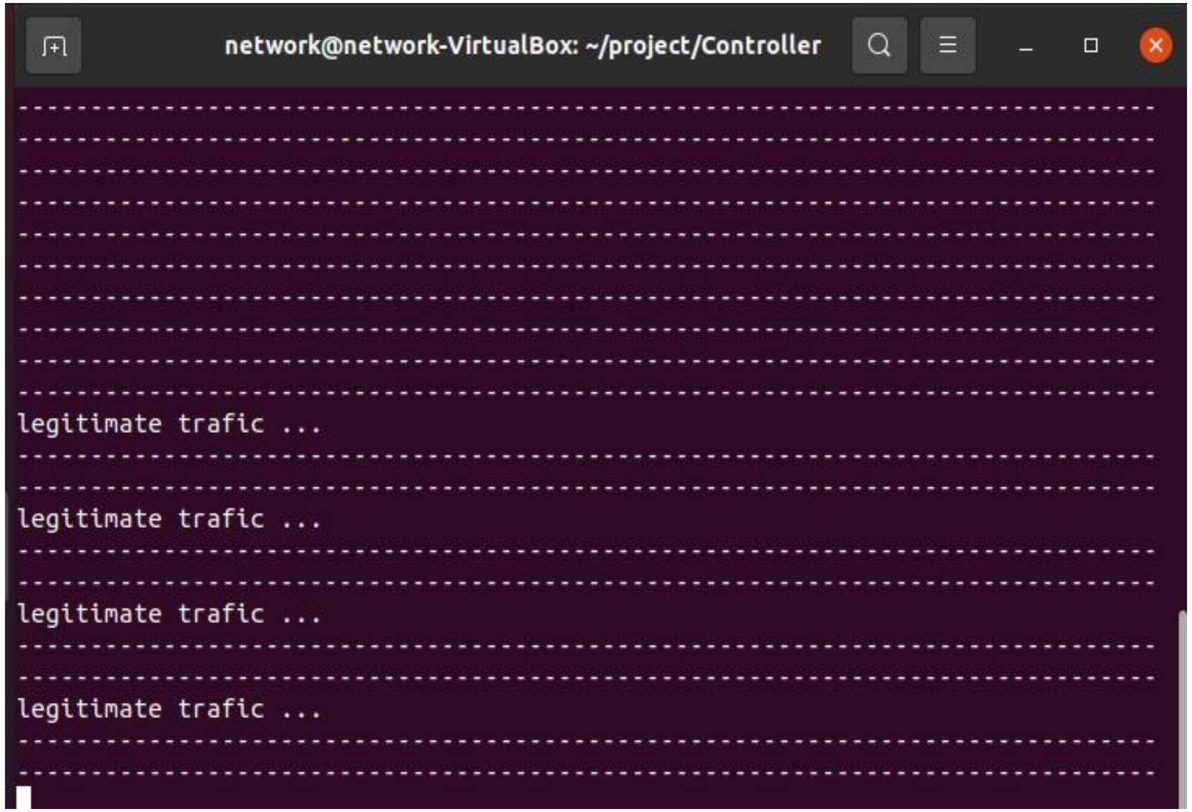
```
network@network-VirtualBox:~/project/mininet$ sudo python3 multi_controller_topo.py
*** Creating controllers
*** Creating network
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1 s2 s3 s4 s5 s6
*** Adding links:
(s1, h1) (s1, h2) (s1, s2) (s2, h3) (s2, h4) (s2, s3) (s3, h5) (s3, s4) (s4, h6) (s4, s5) (s5, h7) (s5, h8)
 (s5, s6) (s6, h9) (s6, h10)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding controllers
*** Starting network
*** Starting controller
c1 c2 c3
*** Starting 6 switches
s1 s2 s3 s4 s5 s6 ...
*** Waiting for switches to connect
s1 s2 s3 s4 s5 s6
*** Testing network
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
*** Testing network again
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
*** Running CLI
```

Fig: Network Topology In Mininet

RESULTS



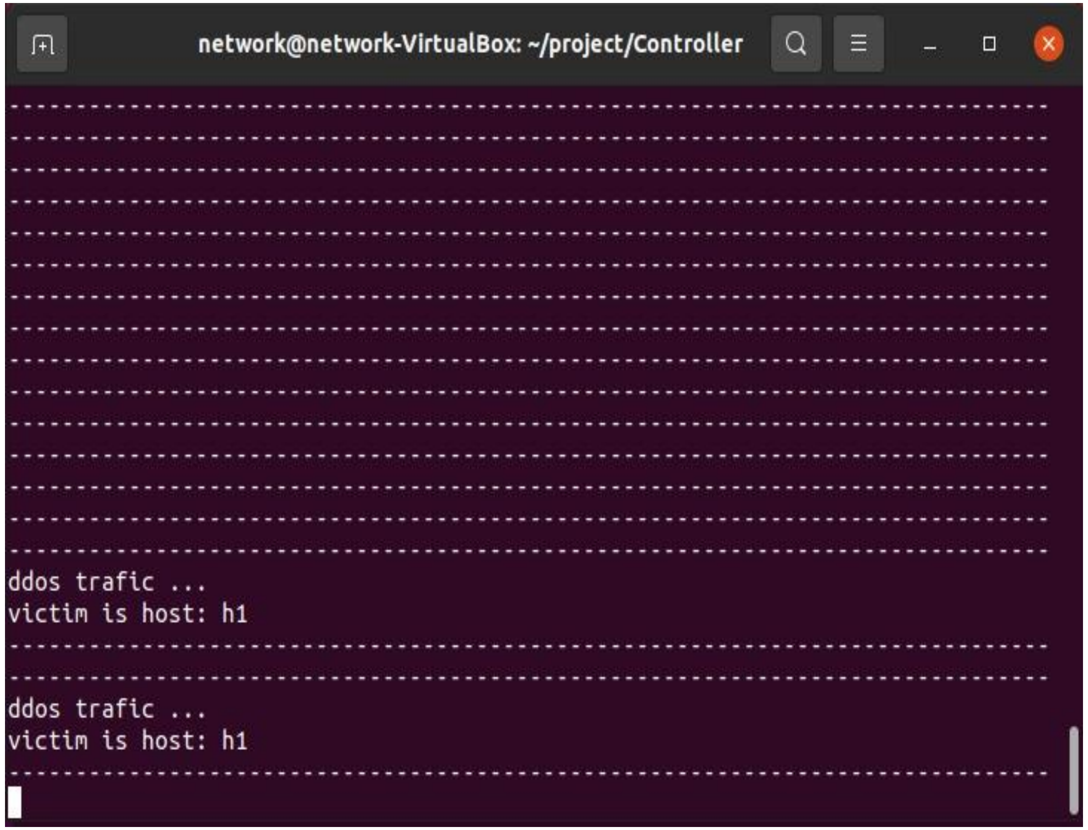
```
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> ph1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 ih4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet>  pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8 h9 h10
h2 -> h1 h3 h4 h5 h6 h7 h8 h9 h10
h3 -> h1 h2 h4 h5 h6 h7 h8 h9 h10
h4 -> h1 h2 h3 h5 h6 h7 h8 h9 h10
h5 -> h1 h2 h3 h4 h6 h7 h8 h9 h10
h6 -> h1 h2 h3 h4 h5 h7 h8 h9 h10
h7 -> h1 h2 h3 h4 h5 h6 h8 h9 h10
h8 -> h1 h2 h3 h4 h5 h6 h7 h9 h10
h9 -> h1 h2 h3 h4 h5 h6 h7 h8 h10
h10 -> h1 h2 h3 h4 h5 h6 h7 h8 h9
*** Results: 0% dropped (90/90 received)
mininet> 
```



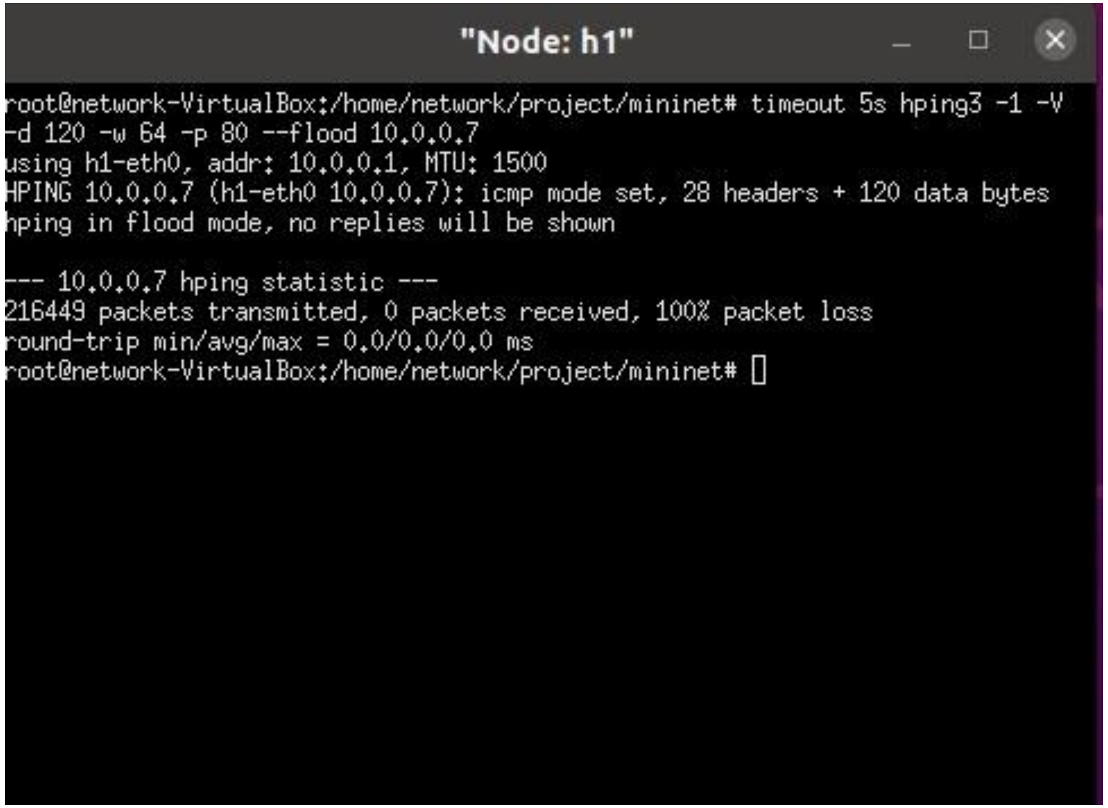
```
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
legitimate trafic ...
------------------------------------------------------------------------------
------------------------------------------------------------------------------
legitimate trafic ...
------------------------------------------------------------------------------
------------------------------------------------------------------------------
legitimate trafic ...
------------------------------------------------------------------------------
------------------------------------------------------------------------------
legitimate trafic ...
------------------------------------------------------------------------------
------------------------------------------------------------------------------
```

Fig: Crediting Traffic as Legitimate

RESULTS



```
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ddos trafic ...
victim is host: h1
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
ddos trafic ...
victim is host: h1
--------------------------------------------------------------------------------
```

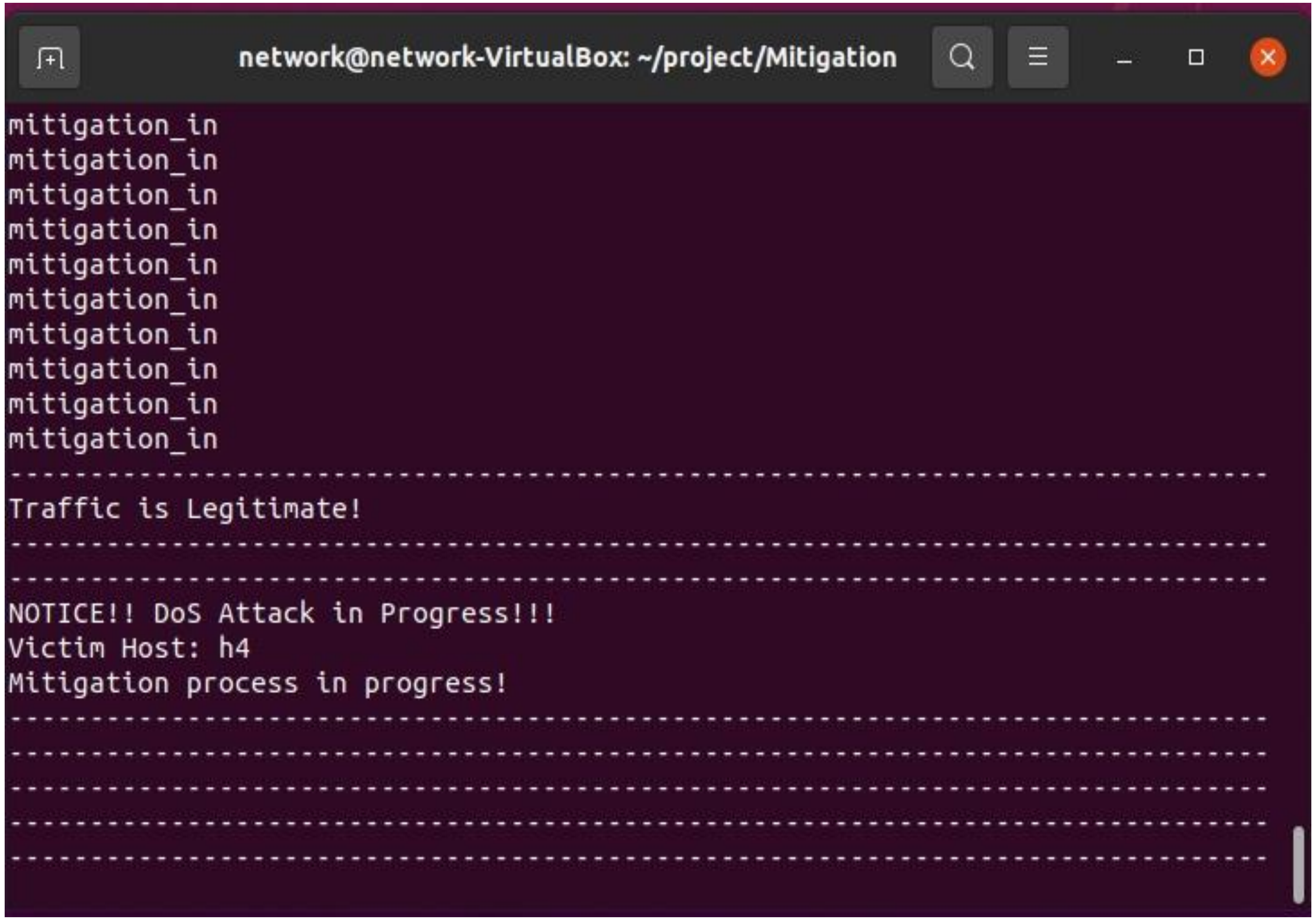


```
root@network-VirtualBox:/home/network/project/mininet# timeout 5s hping3 -1 -V
-d 120 -w 64 -p 80 --flood 10.0.0.7
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.7 (h1-eth0 10.0.0.7): icmp mode set, 28 headers + 120 data bytes
hping in flood mode, no replies will be shown

--- 10.0.0.7 hping statistic ---
216449 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
root@network-VirtualBox:/home/network/project/mininet# 
```

Fig: DDoS Traffic

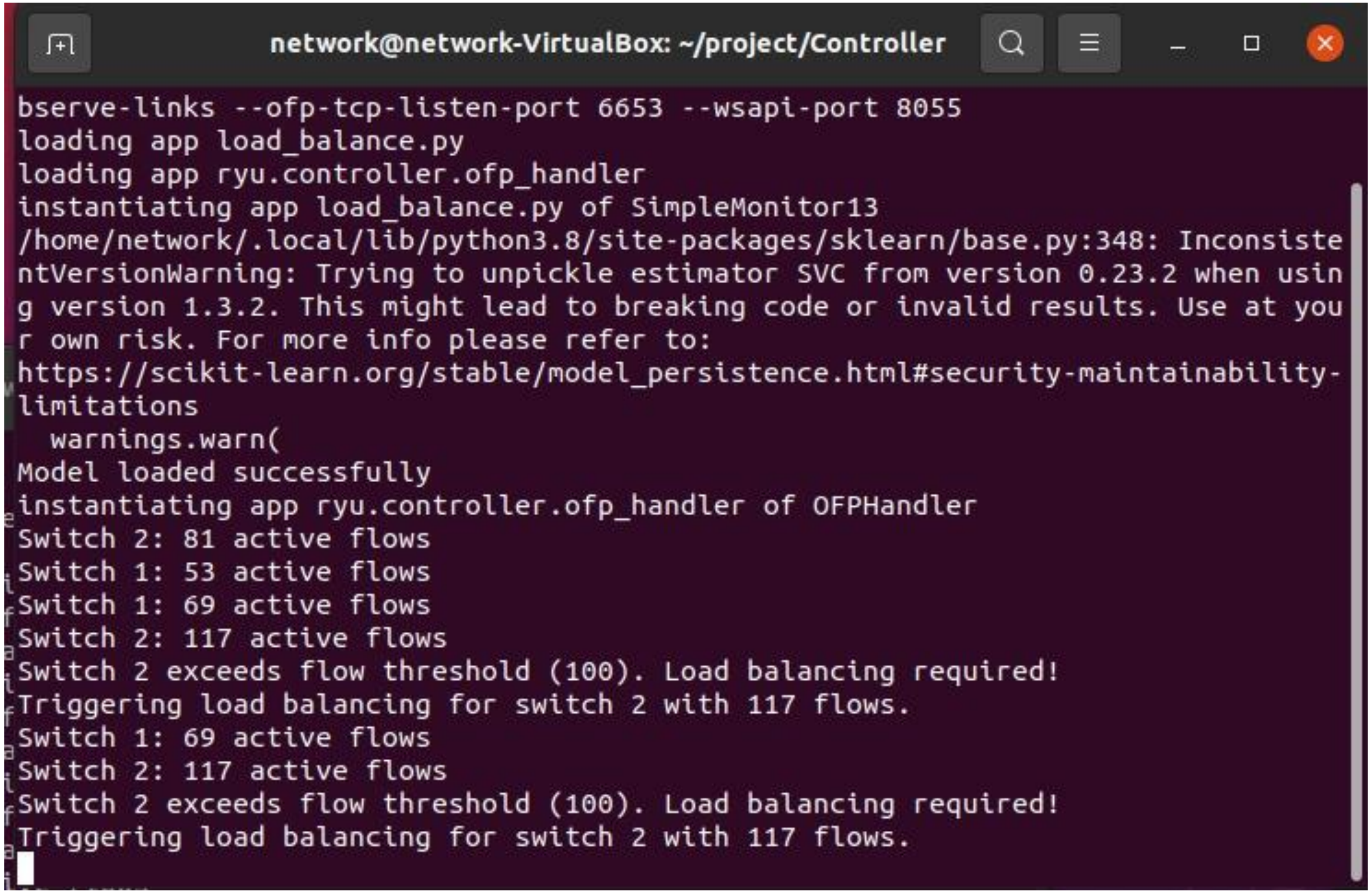
RESULTS



```
mitigation_in
mitigation_in
mitigation_in
mitigation_in
mitigation_in
mitigation_in
mitigation_in
mitigation_in
mitigation_in
mitigation_in
------------------------------------------------------------------------------
Traffic is Legitimate!
------------------------------------------------------------------------------
------------------------------------------------------------------------------
NOTICE!! DoS Attack in Progress!!!
Victim Host: h4
Mitigation process in progress!
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
------------------------------------------------------------------------------
```

Fig: Mitigation

RESULTS



```
bserve-links --ofp-tcp-listen-port 6653 --wsapi-port 8055
loading app load_balance.py
loading app ryu.controller.ofp_handler
instantiating app load_balance.py of SimpleMonitor13
/home/network/.local/lib/python3.8/site-packages/sklearn/base.py:348: Inconsiste
ntVersionWarning: Trying to unpickle estimator SVC from version 0.23.2 when usin
g version 1.3.2. This might lead to breaking code or invalid results. Use at you
r own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-
limitations
  warnings.warn(
Model loaded successfully
instantiating app ryu.controller.ofp_handler of OFPHandler
Switch 2: 81 active flows
Switch 1: 53 active flows
Switch 1: 69 active flows
Switch 2: 117 active flows
Switch 2 exceeds flow threshold (100). Load balancing required!
Triggering load balancing for switch 2 with 117 flows.
Switch 1: 69 active flows
Switch 2: 117 active flows
Switch 2 exceeds flow threshold (100). Load balancing required!
Triggering load balancing for switch 2 with 117 flows.
```

Fig: Load Balancing

DISCUSSION AND ANALYSIS

- This project proposes a DDoS attack detection and mitigation system using machine learning to identify and counteract threats in SDN.
- The system monitors network traffic, detects deviations from normal patterns, and provides real-time alerts and logs.
- The implementation uses a RYU controller to manage SDN modules for monitoring, traffic control, and threat detection, ensuring efficient and secure network operations.
- The detection module uses an ML model to predict potential threats, while the mitigation module blocks hazardous traffic and sends alerts for system monitoring.
- The modular design allows easy integration and testing of components, offering a robust defense against DDoS attacks in SDN environments.

REMAINING TASKS

- Model training of LGBM and Comparision.
- Integrating Load Balancing sub system into main system

REFERENCES

- A. Banitalebi Dehkordi, M. R. Soltanaghaei, and F. Z. Boroujeni. The ddos attacks detection through machine learning and statistical methods in sdn. J. Supercomputer., 77(3):2383–2415, 2020
- D. Erickson. The beacon openflow controller. In HotSDN 2013- Proc. 2013 ACM SIGCOMMWork. Hot Top. Softw. Defin. Netw., pages 13–18, 2013
- A. M. Fatih, G. Cengiz, and K. Enis.(2021) Usage of machine learning algorithms for flowbased anomaly detection system in software defined networks.
- F. Hu, Q. Hao, and K. Bao. A lightweight ddos detection and mitigation system in software-defined networks. Future Generation Computer Systems, 79:205–214, 2018.

THANK YOU