DDoS Attack Detection and Mitigation

Pampati Deekshith Kumar
220010039

Soumyadeep Das
220010056

Subhash Chandra
220120024

Utkarsh Gupta
220020056

Vemula Chandrahaas Reddy
220010062

Abstract

Distributed Denial-of-Service (DDoS) attacks pose a significant threat to network availability by overwhelming systems with malicious traffic. Traditional mitigation techniques often fail to distinguish attack flows from legitimate ones, particularly when attackers mimic normal behavior. This project leverages Software-Defined Networking (SDN) and machine learning to develop a robust DDoS defense system. Using the Ryu SDN framework and Mininet, the system includes a network topology with 6 switches and 18 hosts, traffic creation for simulation, a flow collector for data aggregation, DDoS detection with a Random Forest model, and prevention through dynamic flow rules. This report details the design, implementation, results, and future directions of the system.

Contents

1 Introduction

As networked services become increasingly critical, DDoS attacks threaten their availability by flooding targets with traffic. SDN offers centralized control and programmability, but its controller is a vulnerable target for attackers. This project proposes a machine-learning-driven DDoS defense system, integrating Ryu and Mininet to simulate and mitigate attacks in real time. The system is structured into five key components: network topology, traffic creation, flow collection, DDoS detection, and prevention.

2 System Architecture

2.1 Network Topology

The network topology is simulated using Mininet with a custom configuration defined in `topology.py`. It consists of:

- 6 OpenFlow 1.3-enabled switches (s1 to s6).
- 18 hosts (h1 to h18), each with a unique MAC address and IP in the 10.0.0.0/24 subnet.
- Switch-to-switch and host-to-switch connections forming a linear topology (s1-s2-s3-s4-s5-s6).

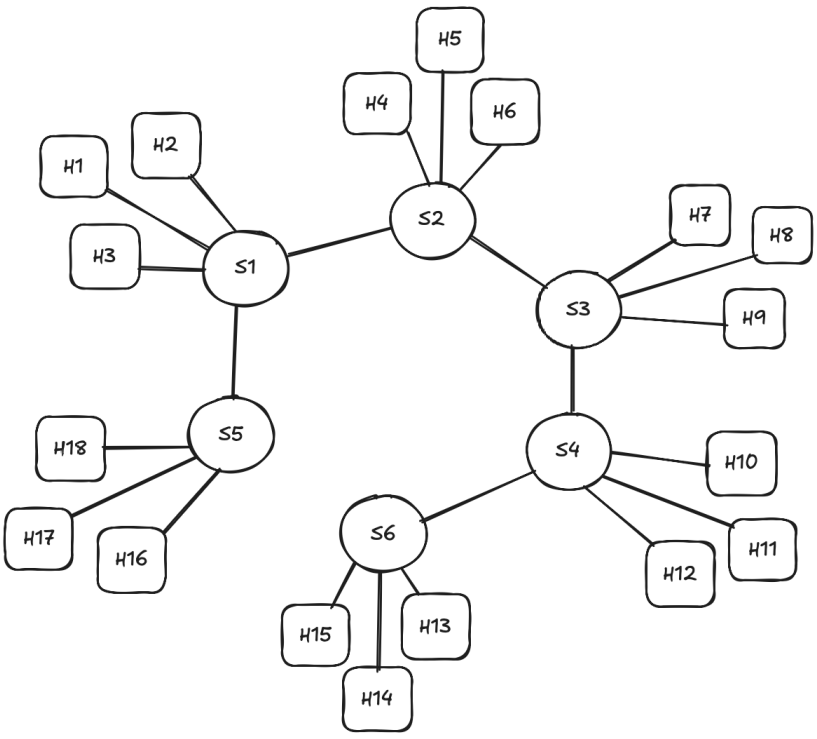

Figure 1: Network Topology

This topology mimics a small-scale enterprise network, allowing for controlled traffic generation and monitoring. The Ryu controller manages the switches, collecting flow statistics and enforcing mitigation rules. A visual representation of the topology can be imagined as a series of interconnected nodes with cables, as depicted in the provided image of networked devices.

2.2 Traffic Creation

Traffic is generated using Mininet scripts to simulate both benign and malicious scenarios:

- **Benign Traffic** (`generate_benign_traffic.py`): - Simulates 100 iterations of ICMP (ping), TCP (HTTP downloads, iperf), and UDP (iperf) traffic. - Hosts randomly initiate traffic to IPs in the 10.0.0.0/24 range, collected as legitimate data.
- **DDoS Traffic** (`generate_ddos_traffic.py`): - Generates a TCP-SYN flood using `hping3`, targeting host h1 with random source IPs. - Additional attack types (e.g., ICMP, UDP floods) are commented out but can be enabled for diversity.

Traffic data is captured by the flow collector, labeled as 0 (benign) or 1 (DDoS) in `FlowStatsfile.csv`, requiring a fix in `start_traffic_collection.py` to handle DDoS labeling dynamically. Our RF model is trained on all three TCP-SYN, ICMP, UDP traffic.

2.3 Flow Collector

The flow collector, implemented in `start_traffic_collection.py` and `test_mitigate.py`, aggregates flow statistics from OpenFlow switches:

- Requests flow stats every 10 seconds (training) or 5 seconds (runtime) using `OFPFlowStatsRequest`.
- Records the seven-tuple (source/destination IP, ports, protocol, ICMP code/type) and counters (packet/byte count, duration).
- Computes derived features: packet/byte counts per second and nanosecond.
- Writes data to `FlowStatsfile.csv` (training) and `PredictFlowStatsfile.csv` (runtime).

This module ensures a continuous data feed for training and real-time analysis, forming the foundation for DDoS detection.

2.4 Detect DDoS

DDoS detection is handled by the Random Forest model in `test_mitigate.py`:

- **Training**: Trains on `FlowStatsfile.csv` with a 60/20/20 train/validation/test split, using 100 estimators.
- **Prediction**: Analyzes `PredictFlowStatsfile.csv` every 5 seconds, classifying flows as legitimate (0) or DDoS (1).
- **Threshold**: Sets a mitigation flag if the legitimate traffic ratio falls below 80%.

Challenges include dataset imbalance (more benign than DDoS samples) and static thresholding, which may be addressed with adaptive techniques.

2.5 Preventing DDoS Attacks

Mitigation logic is implemented in `test_mitigate.py`. The key mechanisms are:

- When the `mitigation` flag is enabled, the `block_port` method installs a high-priority OpenFlow rule to drop traffic from the offending port.
- Each mitigation rule installs a high-priority drop for the offending host's traffic and carries a `10s hard timeout`, automatically blocking the sender for the duration and then expiring to restore normal connectivity.

This approach ensures dynamic and reversible mitigation.

3 System Setup

Click here to view the Screen Recording

Project Setup and Installation

Make the installer script executable and run it:

```
chmod +x install.sh
./install.sh
```

Activating Virtual Environment

Activate the Python 3.7 virtual environment:

```
source ryu37-env/bin/activate
```

Installing Required Packages

Install all required Python packages:

```
pip3 install -r requirements.txt
```

Running the Code

In one terminal, start the Ryu controller:

```
ryu-manager Codes/controller/test_mitigate.py
```

Open a new terminal and run the traffic recorder:

```
python3 recorder.py
```

Open another terminal and launch the Mininet test script with elevated privileges:

```
sudo python3 Codes/mininet/generate_ddos_trafic.py
```

4 Results

Our system successfully identified both benign and malicious network traffic patterns. During normal operation, we observed balanced packet distributions, standard TCP connection sequences, and diverse communication patterns across the network, as shown in Figure 2.

When DDoS attacks were simulated, the system detected abnormal patterns including high packet rates, TCP-SYN flood signatures, and disproportionate traffic to target hosts. Figure 3 demonstrates these malicious traffic signatures.

The mitigation response was effective, with the system promptly identifying attacking ports and implementing blocking rules with 10-second timeouts, as evidenced in Figures 4 and 5. This approach ensured protection while preventing permanent disruption to legitimate users.

Our Random Forest classifier performed well across TCP-SYN, ICMP, and UDP attack types, maintaining high accuracy and low false positives, as demonstrated by the confusion matrix in Figure 6.

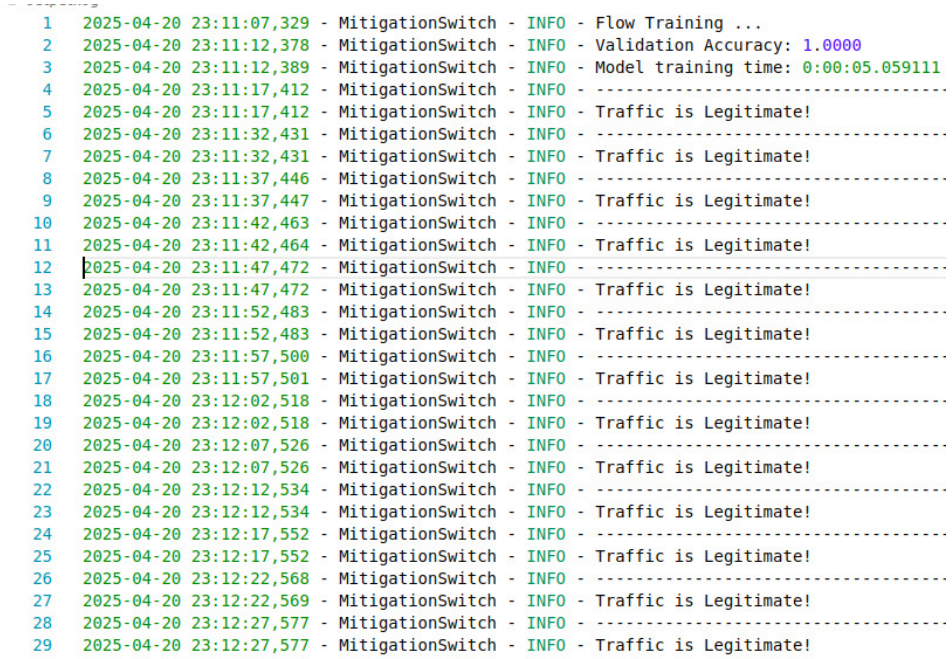

Figure 2: Benign Traffic

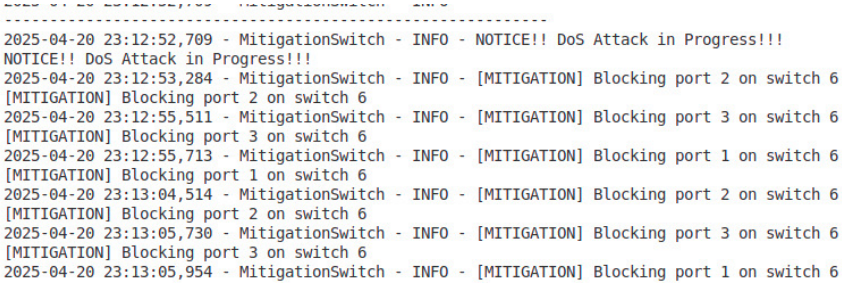

Figure 3: Malicious Traffic

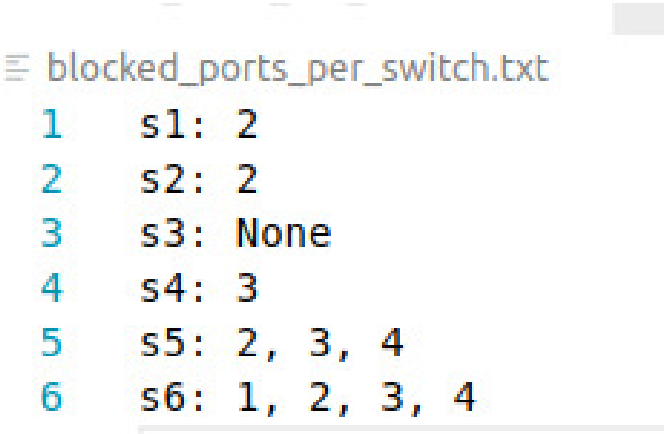

Figure 4: The Blocked Ports

```
shenlong@shenlong-Dell-G15-5530:~/Desktop/Final_Code$ python3 ports.py
s1: 2
s2: 2
s4: 3
s5: 2, 3, 4
s6: 1, 2, 3, 4
```

Figure 5: Verification of Blocked Ports

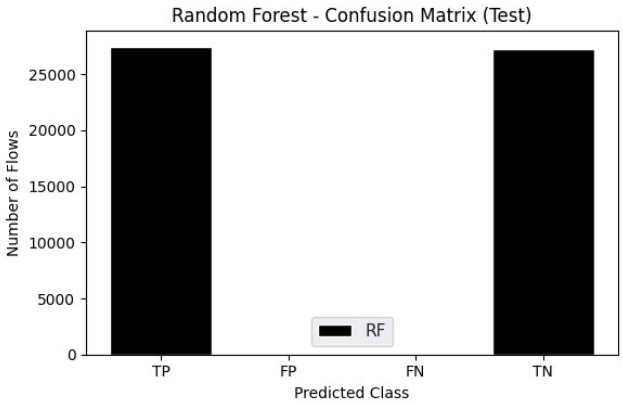

Figure 6: Confusion Matrix

5 Conclusion

This project demonstrates a machine-learning-driven DDoS defense for SDN, effectively integrating network topology, traffic simulation, flow collection, detection, and prevention. The system leverages Ryu and Mininet to achieve near real-time mitigation, with a Random Forest model identifying anomalies and OpenFlow rules blocking attacks. Future work can explore advanced ML models, dynamic thresholds, and multi-controller setups for broader applicability.

6 Future Work

- Integrate deep learning (e.g., LSTM) for time-series analysis.
- Develop adaptive thresholds based on traffic patterns.
- Support multi-controller architectures for scalability.
- Analyze encrypted traffic using flow behavior.
- Create a real-time dashboard (e.g., Grafana) for monitoring.