Real-Time Intrusion Detection in SDN Environments Using Random Forest and Ryu Controller: A Neptune Attack Mitigation Framework

Kogira Sai teja[1], Mahaprabhu L[2], Manthan S Kumar[3], Nagalakshmi R[4], Rekha Jayaram[5]
[1,2,3,4,5]*Department of ISE, Dayananda Sagar College of Engineering, Bengaluru, India*

***Abstract*—Software Defined Networking (SDN) separates the control and data planes to enable centralized, programmable network management, but this centralization also makes the controller vulnerable to attacks such as DDoS and SYN flooding. Traditional rule-based detection methods struggle to handle such dynamic threats in real time. This work proposes a machine learning–driven security framework using a customized Ryu controller with Mininet and MiniNAM. The controller monitors traffic, extracts features, classifies flows as normal or malicious, and dynamically installs OpenFlow rules to block attacks at the switch level. Implemented in Python with Ryu, Open switch, and Mininet, the system is tested under varied traffic scenarios using ping, hping3, and iperf. Results show improved adaptability and accuracy over static rule-based approaches, demonstrating the effectiveness of integrating machine learning with SDN for modern network security.**

***Index Terms*— Software-Defined Networking (SDN); Intrusion Detection System (IDS); Ryu Controller; Mininet; OpenFlow Protocol; Random Forest**

I. INTRODUCTION

Software Defined Networking (SDN) is a modern networking approach that separates the control plane from the data plane, allowing networks to be managed in a more flexible and programmable way. Instead of relying on fixed hardware-based configurations, SDN enables administrators to control network behavior through software using centralized controllers. This makes it well suited for environments such as cloud platforms, data centers, and large enterprise networks. Tools like OpenFlow, Mininet, and the Ryu controller are commonly used to build and experiment with such programmable networks.

However, this flexibility also introduces new security concerns. Because SDN depends on a centralized controller, it can become a target for attacks such as Distributed Denial of Service (DDoS), SYN flooding, and abnormal traffic bursts. A large volume of malicious requests can overwhelm the controller and affect the entire network. Traditional security mechanisms, which rely on fixed rules or known attack signatures, often fail to detect new or evolving threats and cannot adapt quickly to changing network conditions.

To overcome these limitations, this project presents a machine learning-based security framework for SDN. The system uses a customized Ryu controller integrated with Mininet and MiniNAM to monitor network activity in real time. It extracts key flow-based features from traffic and uses a trained machine learning model to classify whether the traffic is normal or malicious. Based on this classification, the controller dynamically installs OpenFlow rules to either allow or block traffic. This approach enables faster detection, better adaptability, and more effective protection against modern network attacks in SDN environments.

II. BACKGROUND & LITERATURE SURVEY

Software Defined Networking (SDN) is a modern networking paradigm that separates the control plane from the data plane, allowing network administrators to manage and configure network behavior through centralized software controllers rather than relying on distributed hardware devices. This architecture improves flexibility, programmability, scalability, and resource management in comparison with traditional networking approaches. SDN commonly uses the OpenFlow protocol to enable communication between

the controller and forwarding devices such as switches. Software Defined Networking has become increasingly important in cloud computing, data centers, enterprise networks, and large-scale communication systems because it allows centralized monitoring and dynamic traffic control [1].

Although SDN provides many advantages, its centralized architecture also introduces significant security challenges. Because the controller manages the entire network, attacks such as Denial of Service (DoS), Distributed Denial of Service (DDoS), SYN flood attacks, and abnormal traffic bursts can overload the controller and disrupt network operations. Traditional intrusion detection systems are usually signature-based or rule-based, making them less effective against new and unknown attack patterns. Researchers have therefore focused on integrating machine learning techniques into SDN to improve attack detection accuracy and reduce false positives. [2]

Machine learning-based intrusion detection systems can analyze traffic flows, extract relevant features, and classify traffic as normal or malicious in real time. Several studies have shown that algorithms such as Random Forest, Decision Tree, Support Vector Machine, and Deep Neural Networks can significantly improve intrusion detection performance in SDN environments. The use of datasets such as NSL-KDD and CICIDS2017 further supports the development of intelligent controllers capable of responding dynamically to evolving threats. These approaches provide better adaptability, scalability, and automated response compared to conventional rule-based methods. [3]

Karthika et al. proposed an SDN-based DDoS detection framework using OpenFlow port statistics and machine learning to identify TCP SYN flood attacks. The system was implemented using Mininet and SDN controllers, and it demonstrated that flow statistics can effectively distinguish between normal and malicious traffic. However, the approach focused mainly on SYN flood attacks and did not address real-time mitigation or broader attack categories [4].

Fatehi and Montazerolghaem presented a deep learning-based DDoS detection method for SDN environments using a Multi-Layer Perceptron (MLP) model integrated with the Ryu controller. Their approach achieved better accuracy than traditional machine learning techniques and enabled real-time attack detection in simulated SDN networks. However, the study relied heavily on dataset-specific behavior and focused mainly on DDoS attacks rather than multiple traffic anomalies [5].

Satheesh et al. developed a flow-based anomaly intrusion detection system using machine learning and SDN. Their work introduced a priority-based traffic management model that continuously monitored network behavior and used Random Forest classification to detect abnormal traffic. The proposed method improved bandwidth utilization and anomaly detection efficiency. However, the system mainly emphasized QoS and bandwidth allocation, with limited focus on controller-based real-time mitigation [6].

Manso et al. proposed an SDN-based intrusion detection system for early detection and mitigation of DDoS attacks. Their approach integrated an IDS with an SDN controller to automatically detect malicious traffic and apply mitigation rules dynamically. The system showed promising results in protecting normal traffic and reducing the impact of attacks. However, the study primarily focused on DDoS attacks and required additional IDS integration, increasing system complexity [7].

Jayaram and Rangarajan [14] and [17] proposed a rule-based approach for identifying failure-inducing combinations in combinatorial testing. Their method systematically analyzes test outcomes to isolate parameter value combinations responsible for failures, demonstrating improved fault localization accuracy. This technique is relevant to SDN testing, where combinatorial interactions among controller parameters, traffic features, and protocol settings can produce unexpected security vulnerabilities.

Jayaram and Krishnan [15] examined SDN controller design from a combinatorial testing perspective, demonstrating that pairwise and higher-strength test suites can effectively cover critical interaction scenarios among SDN parameters. Their work established a systematic basis for designing test cases that expose faults arising from complex parameter interactions in SDN controllers, providing a foundation for evaluating controller security and reliability.

"Christa and Jayaram [16] proposed an automated approach for predicting problematic system parameters using machine learning techniques, presented at IEEE CONECCT 2025. Their work demonstrated that ML models can effectively identify parameter

configurations likely to degrade system performance or cause failures, offering a proactive detection strategy applicable to networked and SDN environments."

III. PROPOSED METHOD

The proposed system is developed using a combination of Software Defined Networking (SDN) components and machine learning techniques to enable intelligent traffic analysis and real-time intrusion detection. It consists of a custom Ryu controller integrated with Mininet, MiniNAM, and Open vSwitch. MiniNAM is used to design and visualize the network topology, while Mininet provides a virtual environment for emulating hosts, switches, and network links. Open vSwitch operates as the forwarding element in the data plane and communicates with the controller through the OpenFlow protocol.

The Ryu controller plays a central role in the system by continuously monitoring network traffic and collecting flow-level information. Relevant features such as packet count, traffic rate, protocol type, source and destination IP addresses, and flow duration are extracted from the incoming traffic. These features are then supplied to a trained machine learning model, which analyzes traffic patterns and classifies them as either normal or malicious.

Based on the classification outcome, the controller dynamically installs appropriate flow rules in the switch. Normal traffic is forwarded without interruption, while malicious traffic is blocked by applying drop rules. This dynamic and adaptive approach improves detection accuracy, minimizes controller overhead, and enhances the overall security of the SDN environment by enabling real-time threat mitigation.

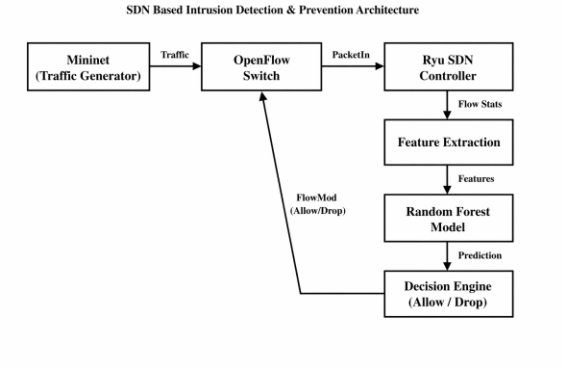

Fig. 1. System Architecture

The use case diagram illustrates the interaction between the administrator and the proposed SDN-based intrusion detection and prevention system. It represents the major functionalities supported by the system and shows how the administrator can monitor and manage network security activities within the SDN environment.

The administrator is responsible for monitoring network traffic, generating normal and attack traffic for testing, viewing logs and alerts, and observing the behavior of the network during intrusion detection and mitigation. The system continuously captures packets from the Open vSwitch and analyzes the traffic flowing through the network.

Important traffic features such as protocol information, packet count, traffic rate, and source and destination details are extracted and forwarded to the Random Forest machine learning model for classification. The model then determines whether the traffic is normal or malicious based on learned traffic patterns.

If malicious traffic such as a Neptune (TCP SYN flood) attack is detected, the Ryu controller automatically installs suitable flow rules, including DROP rules, to block the attacker and protect the network from further harmful activity. The system also generates logs and notifications, enabling the administrator to monitor network events and analyze detected attacks effectively.

Overall, the use case diagram demonstrates the automated workflow of traffic monitoring, attack detection, traffic classification, alert generation, and real-time mitigation within the proposed SDN security framework.

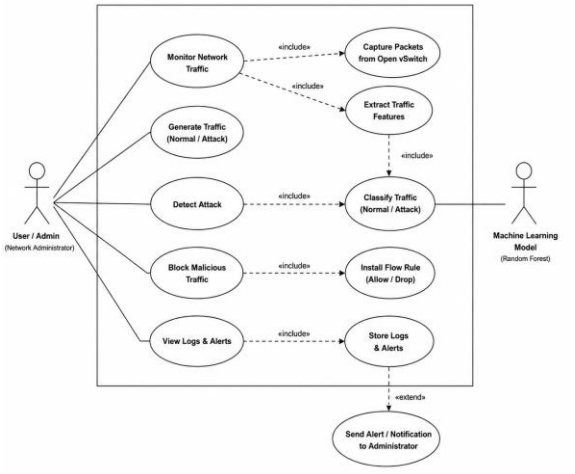

Fig 2. Use Case Diagram

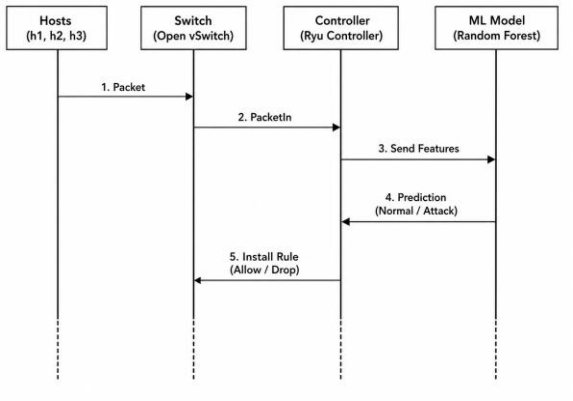

Fig 3. Sequence Diagram

The interaction between the main components of the system is shown in Fig. 1. When a host sends a packet, it is first received by the switch. If the switch does not have a matching rule for that packet, it forwards the request to the controller. The controller then analyzes the packet by extracting important features and passing them to the machine learning model. Based on the model's prediction, the traffic is classified as either normal or malicious. The controller then takes the appropriate action by installing a rule in the switch. Normal traffic is allowed to pass, while malicious traffic is blocked. This process enables the system to respond quickly and handle threats in real time.

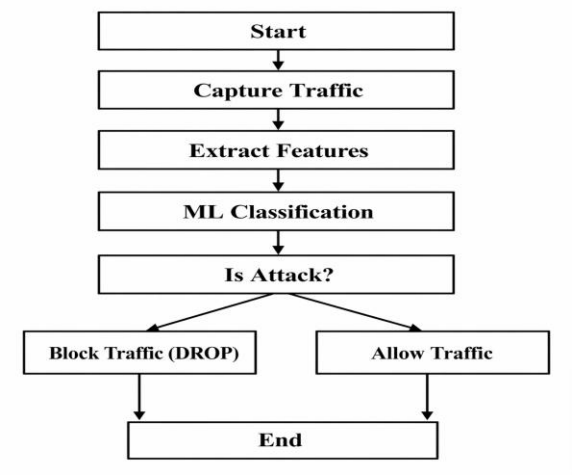

Fig 4. Activity Diagram

The overall working of the system is illustrated in Fig. 2. The process starts with capturing network traffic from the SDN environment. The system then extracts relevant features from the traffic and provides them to the machine learning model for analysis. Based on the prediction, the system checks whether the traffic is normal or an attack. If it is identified as malicious, the system blocks the traffic by applying a drop rule. Otherwise, the traffic is allowed to pass normally. This cycle continues for all incoming traffic, allowing the system to monitor the network continuously and respond automatically to potential threats.

A Components

1. Network Topology Design using MiniNAM

MiniNAM is used as a graphical interface to design the SDN topology. It enables users to create hosts, switches, and links visually, without the need for complex scripting. The resulting topology represents a virtual network environment that can be easily modified and used to test different communication scenarios, making experimentation more efficient.

2. Network Emulation using Mininet

Once the topology is defined, Mininet is used to emulate the network. It creates virtual hosts, Open vSwitch (OVS) instances, and virtual links. Each host behaves like a real machine with its own IP address and network stack. This setup provides a realistic and controlled environment for generating and analyzing network traffic without requiring physical infrastructure.

3. Data Plane Operation using Open vSwitch

Open vSwitch operates in the data plane and is responsible for forwarding packets based on flow rules. When a packet arrives without a matching rule, the switch forwards it to the controller for further processing. After the controller installs the required rules, the switch can handle subsequent packets independently, improving efficiency and reducing controller workload.

4. Control Plane using Neptune ML Ryu Controller

The Ryu controller serves as the central control unit of the network. It communicates with Open vSwitch through the OpenFlow protocol and receives detailed flow-level information, including packet headers and addressing details. The controller analyzes this information, makes decisions, and installs appropriate flow rules in the switches.

5. Machine Learning-Based Traffic Analysis

A machine learning model is integrated into the controller to analyze traffic behavior. It uses features such as packet rate, flow duration, byte count, protocol

type, and TCP flags to identify patterns. A Random Forest classifier is trained to distinguish between normal and malicious traffic, enabling accurate detection of attacks such as DDoS and SYN floods in real time.

6. Intelligent Decision and Network Security Enforcement
Based on the model's prediction, the controller dynamically applies the appropriate action. Normal traffic is allowed by installing forwarding rules, while malicious traffic is blocked by applying DROP rules in the switch. This adaptive mechanism enables real-time threat mitigation while maintaining overall network performance.

B Working Flow
1. Network Topology Design
The process begins with designing the SDN topology using MiniNAM. The network components such as hosts, switches, controllers, and links are created through a graphical interface. This approach simplifies the creation of network structures and allows easy modification without the need for manual scripting.

2. Mininet Emulation
After defining the topology, Mininet is used to emulate the network. It creates virtual hosts, Open vSwitch instances, and links between them. Each host behaves like a real system with its own IP address and network stack, enabling realistic traffic generation in a controlled environment.

3. Ryu Controller Initialization
The Ryu controller is started before the network begins operation. It listens for connections from switches on the OpenFlow port. Once the Open vSwitch starts, it establishes communication with the controller, enabling centralized network management.

4. Traffic Generation
Once the network is active, traffic is generated between hosts. Normal traffic includes ICMP pings and bandwidth tests using iperf, while attack traffic is generated using tools like hping3 to simulate SYN flood attacks and abnormal traffic bursts. These scenarios are used to evaluate system performance.

5. PacketIn Event Trigger
When a switch receives a packet that does not match any existing rule, it forwards the packet to the controller using a PacketIn message. This message contains important details such as source and destination addresses, protocol type, and packet size, allowing the controller to analyze the traffic.

6. Feature Extraction
The controller extracts relevant features from the received traffic. These may include source and destination IP addresses, packet count, byte count, flow duration, protocol type, traffic rate, and TCP flags. These features are used as input for the machine learning model.

7. Machine Learning Classification
The extracted features are passed to the trained machine learning model, which analyzes the traffic pattern and classifies it as normal or malicious. For instance, a SYN flood attack can be identified based on high packet rates, repeated SYN flags, and unusual flow behavior.

8. Flow Rule Installation
Based on the classification result, the controller installs appropriate OpenFlow rules in the switch. Normal traffic is allowed through forwarding rules, while malicious traffic is blocked by applying DROP rules or restricting the source.

9. Real-Time Response
After rule installation, subsequent packets are handled directly by the switch without involving the controller, which reduces processing overhead and improves response time. If new traffic patterns emerge, the switch again communicates with the controller for further decisions.

10. Monitoring and Evaluation
The overall system performance is continuously monitored using tools such as ovs-ofctl, flow statistics, ping outputs, and controller logs. Metrics like detection accuracy, false positives, packet loss, and response time are used to evaluate the effectiveness of the proposed approach.

C. Dataset

Dataset Name	NSL-KDD Dataset
Source	University of New Brunswick
Type of Data	Network intrusion detection data containing normal and attack traffic records, including DoS attacks such as Neptune, Probe, U2R, and R2L
Data Format	Structured tabular dataset with numerical and categorical features
Number of Features	41 input features + 1 class label
Example Features	Duration, Protocol Type, Source Bytes, Destination Bytes, Count, Srv Count, Serror Rate
Target Classes	Normal traffic and attack traffic
Attack Type Used	Neptune Attack
Purpose	Training and testing ML models for intrusion detection in SDN environments

D Algorithm

Random Forest is selected as the classification algorithm in this work due to its strong performance and reliability. Compared to a single decision tree, it provides higher accuracy and reduces the risk of overfitting. It is capable of handling both numerical and categorical features effectively, which makes it well suited for network intrusion detection datasets.

In addition, Random Forest helps identify important features that influence the classification process, such as source bytes, connection count, serror rate, and srv_serror_rate. The algorithm works by constructing multiple decision trees using different subsets of the dataset and randomly selected features. Each tree independently predicts whether the traffic is normal or malicious, and the final decision is obtained through majority voting.

This ensemble approach improves the overall robustness and stability of the model, making it effective for detecting attacks such as Neptune (TCP SYN flood) in SDN environments.

IV. IMPLEMENTATION

The proposed intrusion detection system is implemented by combining Software Defined Networking (SDN) with machine learning techniques to identify Neptune (TCP SYN flood) attacks in real time. Python is used as the primary programming language for data preprocessing, model training, and controller development. The SDN environment is created using Mininet, while network control and monitoring are handled by the Ryu controller.

The NSL-KDD dataset is used to train and evaluate the machine learning model. From this dataset, relevant traffic features such as source bytes, destination bytes, connection count, service count, serror rate, and srv_serror_rate are selected for classification. A Random Forest model is trained to differentiate between normal traffic and attack traffic.

During testing, normal traffic is generated using ping and iperf, while attack traffic is simulated using hping3 to produce SYN flood behavior. The Ryu controller continuously monitors network traffic, extracts the required features, and passes them to the trained model for prediction. Based on the output, the system either allows the traffic or blocks it by installing a DROP rule in the switch. Alerts are also generated when malicious activity is detected, enabling real-time monitoring and response.

Tools and Technologies Used:

Python: Python serves as the core programming language for implementing the system. It is used for data preprocessing, feature extraction, machine learning model development, and integration of the model with the SDN controller for real-time prediction

Ryu Controller: The Ryu controller acts as the central control unit of the SDN environment. It manages communication between network devices and collects traffic statistics such as byte counts, connection rates, and error rates. Based on the analysis, it enforces security decisions by installing appropriate flow rules.

Mininet: Mininet is used to create a virtual SDN network consisting of hosts, switches, and links. It provides a realistic and controlled environment for testing both normal and malicious traffic scenarios without requiring physical hardware.

Random Forest: Random Forest is used as the classification algorithm due to its high accuracy and robustness. It works by combining multiple decision trees and producing the final output through majority voting, making it effective for distinguishing between normal and attack traffic.

Scikit-learn: Scikit-learn is used to train and evaluate the machine learning model. It provides tools for data splitting, model evaluation, and performance metrics such as accuracy and confusion matrix.

Pandas: Pandas is used for handling and preprocessing the dataset. It helps in loading data, selecting relevant features, handling missing values, and converting categorical attributes into numerical form.

NSL-KDD Dataset: The NSL-KDD dataset is used for training and testing the model. It contains labeled records of both normal and attack traffic, including Neptune attacks, along with multiple network-related features.

hping3: hping3 is used to generate attack traffic in the network. It creates a large number of TCP SYN packets to simulate Neptune attack behavior, allowing the system to be tested under realistic attack conditions.

V. RESULTS AND DISCUSSION

Accuracy: 99.6451541361721%

Classification Report:

Class	Precision	Recall	F1-Score	Support
0 (Normal)	1.00	1.00	1.00	3178
1 (Attack)	1.00	0.99	0.99	1331
Accuracy	-	-	1.00	4509
Macro Avg	1.00	1.00	1.00	4509
Weighted Avg	1.00	1.00	1.00	4509

Confusion Matrix:

	Predicted Normal	Predicted Attack
Actual Normal	3173	5
Actual Attack	11	1320

The proposed system achieved an overall accuracy of 99.64%, indicating its strong ability to distinguish between normal and malicious network traffic. This high accuracy reflects the effectiveness of integrating machine learning with the SDN framework.

The classification results show that both normal and attack classes have precision values close to 1.00, which means the model produces very few false positives. The recall value for attack traffic is 0.99, suggesting that the system successfully detects most malicious activities, with only a small number of attacks remaining undetected.

The confusion matrix further confirms this performance. Out of all normal instances, 3173 were correctly classified, with only 5 misclassified as attacks. Similarly, 1320 attack instances were correctly identified, while only 11 were missed.

These observations indicate that the model achieves a good balance between accuracy and reliability. The low number of misclassifications highlights the robustness of the Random Forest algorithm in identifying intrusion patterns effectively.

Overall, the system demonstrates high detection capability, a low false alarm rate, and efficient real-time response, making it a suitable solution for securing SDN-based network environments.

VI. ADVANTAGES

1. The NSL-KDD dataset is effectively utilized for training and testing the model, providing a balanced set of normal and attack traffic, including Neptune attacks.
2. Proper data preprocessing is carried out to improve model performance. Irrelevant attributes are removed, categorical features are converted into numerical values, and inconsistencies in the data are handled.
3. Relevant traffic features such as duration, protocol type, source and destination bytes, connection count, and error rates are carefully selected, as they play a key role in identifying attack patterns.
4. The dataset is divided into training and testing sets, ensuring that the model is evaluated on unseen data for reliable performance assessment.
5. A Random Forest classifier is used, which provides high accuracy and reduces overfitting by combining multiple decision trees.
6. The model is evaluated using standard performance metrics such as accuracy, precision, recall, and confusion matrix, ensuring reliable validation of results.
7. The trained model is integrated with the Ryu controller, enabling real-time traffic analysis and prediction within the SDN environment.
8. A virtual network topology is created using Mininet, allowing realistic simulation of network behavior without the need for physical infrastructure.
9. Both normal and attack traffic are generated within the network, enabling comprehensive testing of the system under different conditions.

10. The Ryu controller continuously monitors traffic and collects relevant statistics required for intrusion detection.
11. Extracted traffic features are processed by the machine learning model to accurately classify traffic as normal or malicious.
12. The system provides real-time response by generating alerts and dynamically blocking malicious traffic using flow rules.
13. Overall, the system ensures high detection accuracy, low false alarm rate, and efficient handling of network traffic in SDN environments.

VII. CONCLUSION AND FUTURE SCOPE

The proposed system is designed to detect Neptune (TCP SYN flood) attacks in an SDN environment using machine learning techniques. It is implemented using Python, Mininet, and the Ryu controller, with a Random Forest model trained on the NSL-KDD dataset to classify network traffic as either normal or malicious.

The model effectively identifies key traffic features such as source bytes, connection count, serror rate, and srv_serror_rate, which are critical for detecting attack patterns. By integrating the trained model into the controller, the system is able to perform real-time traffic analysis and make decisions dynamically. Experimental results show that the system can accurately distinguish between normal and attack traffic.

Overall, the proposed approach provides an intelligent and automated mechanism for detecting and mitigating network attacks in SDN environments. It enhances network security by identifying malicious traffic at an early stage and can automatically block attackers when necessary. Additionally, the use of open-source tools makes the system cost-effective and suitable for further research and practical deployment.

Limitations:

The proposed system has a few limitations that may affect its performance in real-world scenarios. First, it is specifically designed to detect Neptune (TCP SYN flood) attacks and may not perform equally well for other types of network attacks. The effectiveness of the model also depends on the quality of the NSL-KDD dataset used for training, which may not fully capture modern network behavior or recent attack patterns. In some cases, normal traffic may exhibit characteristics similar to malicious activity, which can lead to false positives. Additionally, the system has been evaluated only in a simulated environment using Mininet, and its performance may differ when deployed in real-world networks.

The feature extraction process within the Ryu controller can introduce additional overhead, especially under high traffic conditions. Moreover, the system may struggle to detect more advanced or stealthy attacks that are not well represented in the training data. Finally, the current implementation focuses mainly on network-layer attacks such as SYN floods and does not address application-layer attacks or encrypted traffic.

Future Scope:

The proposed system can be enhanced in several ways to improve its effectiveness and applicability. One possible extension is to expand the system's capability to detect a wider range of network attacks beyond Neptune (TCP SYN flood). Incorporating advanced machine learning and deep learning techniques could further improve detection accuracy and help reduce false positives.

Training the model with more recent and diverse datasets would allow it to better reflect current network traffic patterns and emerging attack strategies. In addition, evaluating the system in larger and more complex SDN environments with multiple switches and hosts would help assess its scalability and performance under realistic conditions.

The system can also be improved by integrating real-time visualization dashboards to monitor network activity and detection results more effectively. The controller can be further enhanced to apply different mitigation strategies depending on the type of attack detected, making the response more adaptive.

Future developments may include extending the system to cloud and IoT environments, enabling broader security coverage. Detecting encrypted malicious traffic and application-layer attacks is another important direction. Finally, combining Random Forest with other machine learning models in a hybrid approach could improve overall robustness and performance.

REFERENCES

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[2] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. IEEE Local Computer Network Conference*, Oct. 2010, pp. 408–415.

[3] M. S. Elsayed, N. A. Le-Khac, S. Dev, and A. D. Jurcut, "Machine-learning techniques for detecting attacks in SDN," in *Proc. 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, Oct. 2019, pp. 277–281.

[4] Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. International Conference on Information Systems Security and Privacy (ICISSP)*, vol. 1, 2018, pp. 108–116.

[5] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[6] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[7] R. Ma, Q. Wang, X. Bu, and X. Chen, "Real-time detection of DDoS attacks based on random forest in SDN," *Applied Sciences*, vol. 13, no. 13, p. 7872, 2023.

[8] P. Manso, J. Moura, and C. Serrão, "SDN-based intrusion detection system for early detection and mitigation of DDoS attacks," *Information*, vol. 10, no. 3, p. 106, 2019.

[9] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in SDN," in *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2018.

[10] N. Ashodia and K. Makadiya, "Detection of DDoS attacks in SDN using machine learning," in *Proc. 2022 International Conference on Electronics and Renewable Systems (ICEARS)*, Mar. 2022, pp. 1322–1327.

[11] M. S. Ataa, E. E. Sanad, and R. A. El-Khoribi, "Intrusion detection in software defined network using deep learning approaches," *Scientific Reports*, vol. 14, no. 1, p. 29159, 2024.

[12] Adamou Djergou, Y. Maleh, and S. Mounir, "Machine learning techniques for intrusion detection in SDN: A survey," in *Proc. International Conference on Information, Communication & Cybersecurity*, Cham, Switzerland: Springer, Nov. 2021, pp. 460–473.

[13] D. Agrawal, C. Agrawal, and H. Yadav, "A machine learning based intrusion detection framework using KDDCUP 99 dataset," vol. 4, no. 11, 2020.

[14] R. Jayaram and K. Rangarajan, "A rule-based approach for identifying failure inducing combinations in combinatorial testing," *Journal of Information Systems Engineering and Management*, vol. 10, no. 5s, 2025.

[15] R. Jayaram and R. Krishnan, "Combinatorial test perspective on test design of SDN controllers," in *Proc. Second International Conference on Sustainable Expert Systems (ICSES 2021)*, Singapore: Springer Nature Singapore, 2022.

[16] S. Christa and R. Jayaram, "Automatic prediction of problematic system parameters using machine learning techniques," in *Proc. 2025 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, IEEE, 2025.

[17] M. Chakravarthy, V. Keerthana, A. Garg, R. Jayaram, and K. Rangarajan, "Identification of failure inducing combinations using greedy heuristic algorithm," *Journal of Information Systems Engineering and Management*, vol. 10, no. 30s, 2025.