



**Prifysgol Cymru**  
Y Drindod Dewi Sant  
**University of Wales**  
Trinity Saint David

**University of Wales Trinity Saint David, London**

## **Masters Project Report**

**Adaptive AI-Based Real-Time auto DDoS and zero-day Detection  
and  
Mitigation for SDN Network**

**Shafee Ul Mahmud Chowdhury**  
Student ID: 2436374

**Project submitted as part of the requirements for the award  
of  
MSc Cyber Security and Digital Forensics**

Supervisor: Renuga Jayakumar

September 2026

## Declaration and Consent

### Declaration

I, Shafee Ul Mahmud Chowdhury, hereby declare that this Master's Project is my own original work and that I have undertaken the research and preparation of this report independently. All sources, ideas, data, and materials consulted or incorporated in this project have been appropriately acknowledged and referenced. I further confirm that this report complies with the prescribed word limit and has not previously been submitted, in whole or in part, for the award of any other degree or academic qualification.

**Signature:** \_\_\_\_\_ Shafee Ul Mahmud Chowdhury \_\_\_\_\_

**Date:** \_\_\_\_\_ 25<sup>th</sup> September 2026 \_\_\_\_\_

### Form of Consent

I, Shafee Ul Mahmud Chowdhury, give my consent for this Master's Project, submitted in fulfilment of the requirements for the award of MSc in Cybersecurity and Digital Forensic, to be made available for inter-library loan or photocopying, where applicable, in accordance with the relevant copyright regulations. I also authorize the University to make the title and abstract of this project available to external organizations for appropriate academic or educational purposes.

**Signature:** \_\_\_\_\_ Shafee Ul Mahmud Chowdhury \_\_\_\_\_

**Date:** \_\_\_\_\_ 25<sup>th</sup> September 2026 \_\_\_\_\_

## Abstract

Distributed Denial-of-Service (DDoS) attacks remain a significant challenge for modern networks because they can generate large volumes of malicious traffic that disrupt service availability and degrade network performance. Traditional network security approaches may struggle to detect evolving attack patterns and provide timely mitigation. This research investigates a Software-Defined Networking (SDN)-based approach for detecting and mitigating DDoS attacks by integrating Machine Learning (ML) with programmable network control.

The proposed system was implemented in a controlled SDN environment using Mininet and Open vSwitch (OVS), with OS-Ken serving as the SDN controller through the OpenFlow 1.3 protocol. The INSDN dataset was used as the principal reference dataset for developing the detection model. A Random Forest classifier was trained using network-flow features to distinguish between benign and DDoS traffic. The feature set was expanded to 20 traffic and behavioral features, including packet and byte rates, flow duration, packet counts, source and destination statistics, and protocol-related characteristics. The trained model achieved an accuracy of 91.96%, with precision of 93.37%, recall of 93.77%, and an F1-score of 93.57% during model evaluation.

To improve the robustness of the detection process, the proposed framework combines Random Forest classification with behavioral analysis. In addition, an Isolation Forest-based novelty detection mechanism was incorporated to identify traffic patterns that differ from established benign behavior, providing an additional mechanism for detecting potentially previously unseen attacks. When malicious traffic is identified with sufficient confidence, the OS-Ken controller dynamically generates OpenFlow mitigation rules and installs them in the OVS flow table to block the identified malicious traffic. The mitigation mechanism also monitors flow activity and removes or releases mitigation rules when malicious activity is no longer observed.

Experimental evaluation was conducted under normal, attack, and post-mitigation conditions. The results demonstrate that the proposed SDN-based framework can identify malicious traffic, dynamically enforce mitigation through the network infrastructure, and support restoration of normal service conditions after an attack. The integration of ML-based detection with programmable SDN control provides a flexible approach to automated DDoS defense. Overall, the research demonstrates the practical feasibility of combining supervised classification, behavioral analysis, novelty detection, and OpenFlow-based mitigation within an SDN environment.

**Keywords:** Software-Defined Networking, DDoS Attack, Machine Learning, Random Forest, OS-Ken, OpenFlow, Mininet, Open vSwitch, Isolation Forest, Network Security.

## Table of contents

|                                                                                           |    |
|-------------------------------------------------------------------------------------------|----|
| Declaration and Consent .....                                                             | 2  |
| Abstract .....                                                                            | 3  |
| Table of contents .....                                                                   | 4  |
| List of Figures .....                                                                     | 7  |
| List of Tables .....                                                                      | 8  |
| 1. INTRODUCTION .....                                                                     | 9  |
| 1.1 Background .....                                                                      | 9  |
| 1.2 Software-Defined Networking and Its Advantages and vulnerability for Network Security | 9  |
| 1.3 Artificial intelligence and Machine learning .....                                    | 10 |
| 1.4 Research Motivation and Problem Context .....                                         | 10 |
| 1.5 Research Aim and Objectives .....                                                     | 10 |
| 2. LITERATURE REVIEW .....                                                                | 12 |
| 2.1 Overview .....                                                                        | 12 |
| 2.2 Network Vulnerabilities and Various Attacks .....                                     | 12 |
| 2.3 Why SDN Was Selected over Traditional Networking, .....                               | 13 |
| 2.4 DDoS Attacks and the SDN Security Problem .....                                       | 14 |
| 2.5 Machine Learning-Based DDoS Detection .....                                           | 14 |
| 2.6 Deep Learning Approaches .....                                                        | 15 |
| 2.7 DDoS and SDN Datasets .....                                                           | 16 |
| 2.8 InSDN Dataset .....                                                                   | 17 |
| 2.9 Comparative Dataset Analysis, .....                                                   | 17 |
| 2.10 Generalization and Previously Unseen DDoS Behaviors .....                            | 18 |
| 2.11 Real-Time DDoS Detection .....                                                       | 19 |
| 2.12 Adaptive Detection .....                                                             | 20 |
| 2.13 SDN-Based Automated Mitigation .....                                                 | 20 |
| 2.14 Integrated ML for Detection and Mitigation .....                                     | 21 |
| 2.15 Existing Research and the Proposed Research .....                                    | 21 |
| 2.16 Critical Comparison of Representative Literature .....                               | 22 |
| 2.17 Evaluation Metrics: From Classification to System Performance .....                  | 23 |

|                                                                                 |    |
|---------------------------------------------------------------------------------|----|
| 2.18 Methodological Limitations Identified in Existing Research .....           | 24 |
| 2.19 SDN Environment and Selection of OS-Ken, OpenFlow, OVS and Mininet .....   | 24 |
| 2.20 Selection of Random Forest .....                                           | 26 |
| 2.21 Python and Supporting Libraries .....                                      | 27 |
| 2.22 Why Machine Learning Rather Than Deep Learning .....                       | 27 |
| 2.23 System Architecture and Network Topology .....                             | 28 |
| 2.24 Research Gap .....                                                         | 28 |
| 2.25 Evaluation Matrix - Accuracy, F1-Score and Latency in DDoS Detection ..... | 30 |
| 2.26 Summary .....                                                              | 30 |
| 3. METHODOLOGY .....                                                            | 32 |
| 3.1 Research Philosophy .....                                                   | 32 |
| 3.2 Data Collection .....                                                       | 32 |
| 3.3 Research Methodology .....                                                  | 32 |
| 3.4 How the Research Will Be Conducted .....                                    | 32 |
| 4. DESIGN AND IMPLEMENTATION .....                                              | 34 |
| 4.1 Overview .....                                                              | 34 |
| 4.2 SDN Network Topology Design .....                                           | 34 |
| 4.3 Web-Service Implementation .....                                            | 36 |
| 4.4 OS-Ken Controller Implementation .....                                      | 36 |
| 4.5 Flow Monitoring and Feature Extraction .....                                | 37 |
| 4.6 InSDN Dataset Processing .....                                              | 38 |
| 4.7 Random Forest Model Implementation .....                                    | 38 |
| 4.8 Python-Based Detection Component .....                                      | 39 |
| 4.9 Detection Decision and Controller Integration .....                         | 40 |
| 4.10 SDN-Based Mitigation Implementation .....                                  | 42 |
| 4.11 End-to-End Implementation Workflow .....                                   | 42 |
| 4.12 Previously Unseen Attack Implementation .....                              | 43 |
| 4.13 Implementation Verification .....                                          | 43 |
| 4.14 Chapter Summary .....                                                      | 46 |
| 5. EVALUATION .....                                                             | 48 |
| 5.1 Overview .....                                                              | 48 |

|                                                         |    |
|---------------------------------------------------------|----|
| 5.2 Experimental Scenarios .....                        | 48 |
| 5.3 DDoS Detection and Classification Performance ..... | 49 |
| 5.4 Previously Unseen DDoS Behavior .....               | 52 |
| 5.5 Detection Latency .....                             | 53 |
| 5.6 Mitigation Response Time .....                      | 54 |
| 5.7 Network Performance .....                           | 55 |
| 5.8 End-to-End Evaluation .....                         | 57 |
| 5.9 Evaluation Summary .....                            | 57 |
| 5.10 Comparison with existing research .....            | 58 |
| 5.11 Limitations .....                                  | 59 |
| 6. CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK ..... | 60 |
| 6.1 Conclusion .....                                    | 60 |
| 6.2 Recommendations for Future Work .....               | 61 |
| 7. REFLECTION .....                                     | 62 |
| 8. REFERENCES .....                                     | 64 |

## List of Figures

|                                                                                                       |    |
|-------------------------------------------------------------------------------------------------------|----|
| Figure 1 Sequential detection and mitigation pipeline.....                                            | 34 |
| Figure 2 Mininet/OVS topology and IP addressing used in the experimental environment.....             | 35 |
| Figure 3 OS-Ken controller interaction with the OVS data plane and OpenFlow control path.....         | 37 |
| Figure 4 Flow-statistics collection and feature-extraction pipeline.....                              | 37 |
| Figure 5 InSDN data-preparation and model-training pipeline.....                                      | 38 |
| Figure 6 Random Forest training, validation, and model-loading workflow.....                          | 39 |
| Figure 7 Machine-learning detection decision process.....                                             | 40 |
| Figure 8 Integration Process.....                                                                     | 41 |
| Figure 9 ML detection-to-OS-Ken controller integration.....                                           | 41 |
| Figure 10 OpenFlow mitigation process from malicious-flow detection to OVS flow-rule enforcement..... | 42 |
| Figure 11 End-to-end implementation flowchart.....                                                    | 43 |
| Figure 12 Normal traffic flow.....                                                                    | 44 |
| Figure 13 Generating Attack traffic from kali Linux.....                                              | 45 |
| Figure 14 Attack detected.....                                                                        | 45 |
| Figure 15 Drop rule installed.....                                                                    | 46 |
| Figure 16 Normal traffic and successful web-service access.....                                       | 48 |
| Figure 17 Controlled DDoS traffic generation.....                                                     | 49 |
| Figure 18 Random Forest DDoS detection output.....                                                    | 49 |
| Figure 19 Random Forest classification performance for known DDoS traffic.....                        | 50 |
| Figure 20 Random Forest confusion matrix.....                                                         | 51 |
| Figure 21 Accuracy, precision, recall and F1 score.....                                               | 51 |
| Figure 22 Generating Unseen Zero-day attack.....                                                      | 52 |
| Figure 24 Detection output for previously unseen DDoS behaviour.....                                  | 53 |
| Figure 25 Detection latency across experimental runs.....                                             | 54 |

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| Figure 26 OS-Ken detection and mitigation log.....                                     | 55 |
| Figure 27 OVS flow table before mitigation.....                                        | 55 |
| Figure 28 OVS flow table after mitigation.....                                         | 55 |
| Figure 29 Network performance under normal, attack and post-mitigation conditions..... | 56 |
| Figure 30 End-to-end detection-to-mitigation process and experimental evidence.        | 57 |

## List of Tables

|                                                   |    |
|---------------------------------------------------|----|
| Table 1 Dataset Comparison.....                   | 18 |
| Table 2 Gap Comparison.....                       | 22 |
| Table 3 Critical Comparison.....                  | 23 |
| Table 4 Comparison between existing research..... | 58 |

# 1. INTRODUCTION

## 1.1 Background

The rapid growth of digital technologies has made computer networks essential for cloud computing, IoT, 5G, edge computing, and enterprise services. As dependence on interconnected systems increases, network security and resilience have become critical requirements. Modern networks face various threats, including malware, brute-force attacks, phishing, intrusion, and denial-of-service attacks.[1] Among these, DDoS attacks are particularly significant because they can overwhelm network resources with excessive or abnormal traffic, directly affecting service availability and legitimate users. Their increasing scale and diversity also make timely detection and effective mitigation challenging. [1][2]

Traditional networks rely on distributed control, where individual routers, switches, and security devices manage their own control and forwarding functions. This can limit network-wide visibility, programmability, and rapid response to changing security conditions. [2][3]

Software-Defined Networking (SDN) is used in this project because of its centralized control, network-wide visibility, and programmable flow management to support dynamic security monitoring and automated mitigation. However, the centralized SDN controller can act as a single point of failure, potentially affecting network availability if compromised or unavailable. [2] [3]

SDN can support DDoS defense by analyzing flow-level traffic and dynamically applying mitigation actions such as blocking or restricting suspicious traffic. However, centralization can expose the controller and control plane to attacks, while abnormal flows may increase controller workload and affect legitimate traffic. Therefore, effective DDoS protection requires accurate detection and timely automated mitigation.[4] [5]

## 1.2 Software-Defined Networking and Its Advantages and vulnerability for Network Security

Software-Defined Networking (SDN) separates network control logic from forwarding infrastructure and provides centralized control through an SDN controller. OpenFlow enables the controller to dynamically add, modify, and remove flow rules on switches.[2] These capabilities provide network-wide visibility and programmability, supporting automated security monitoring and response. In this project, SDN is used for DDoS detection and mitigation, allowing suspicious traffic to be analyzed and malicious flows to be blocked dynamically.[4] However, centralization also creates security challenges, as excessive requests can saturate the controller and make it a potential single point of failure. [3], [4], [5]

### 1.3 Artificial intelligence and Machine learning

Artificial Intelligence (AI) refers to the ability of computer systems to perform tasks that normally require human intelligence, such as learning from data, recognizing patterns, making decisions, and adapting to changing conditions.[6][5] In network security, AI can analyze large volumes of traffic data to identify unusual or potentially malicious behavior. Machine Learning (ML) is a subset of AI that enables systems to learn patterns from data and make predictions without requiring every decision to be explicitly programmed. [5] In this project, ML is effective because it can analyze multiple network-flow features to identify DDoS traffic more intelligently than relying only on fixed rules or thresholds. Combined with SDN, ML-based detection can support automated and dynamic mitigation of malicious flows.[5][6]

### 1.4 Research Motivation and Problem Context

Existing research shows that SDN provides centralized visibility and programmable control for automated DDoS defense, while ML and deep learning can identify complex traffic patterns. However, many studies focus on offline datasets and classification accuracy, with limited evaluation of complete real-time detection and mitigation.[4] Unseen DDoS behavior, detection latency, mitigation response time, and network performance are not consistently evaluated together. [2]This research therefore investigates an Adaptive AI-Based Real-Time DDoS Detection and Mitigation framework in an SDN environment, combining ML-based traffic analysis with programmable SDN control to evaluate these factors under live conditions. [2] [4] [6][7]

### 1.5 Research Aim and Objectives

**Aim:** The aim of this research is to develop and evaluate an adaptive AI-based real-time DDoS detection and mitigation framework in an SDN environment, with a particular focus on previously unseen DDoS behavior and timely network-level response.

**Objectives:** The research objectives are to:

1. Evaluate the framework against known and previously unseen DDoS behavior
2. To design and implement an SDN-based testbed for generating, monitoring, and analyzing normal and DDoS traffic under controlled conditions.
3. Develop an AI/ML-based mechanism for detecting DDoS traffic using network flow features.
4. Integrate the ML detection mechanism with SDN-based automated mitigation.
5. Evaluate detection performance, detection latency, mitigation response time, and network/service performance.

To design and implement an SDN-based testbed using Mininet as the network emulation platform, with an SDN controller managing the virtual network through OpenFlow. The testbed will provide a controlled environment for generating, monitoring and analyzing normal and DDoS traffic, while enabling the integration and evaluation of AI/ML-based detection and automated mitigation mechanisms.

The Introduction chapter presents the research background, aim, objectives, and research gap. The Literature Review chapter examines SDN, DDoS, AI/ML, and existing detection approaches. The Methodology chapter explains the research methods. System Design and Implementation describes the proposed framework. Results and Evaluation presents findings, while Discussion and Conclusion covers interpretations, limitations and future work.

## 2. LITERATURE REVIEW

### 2.1 Overview

This chapter critically reviews research related to Distributed Denial-of-Service (DDoS) detection and mitigation in Software-Defined Networking (SDN), with particular emphasis on machine learning (ML), deep learning (DL), adaptive detection, previously unseen attack behavior, real-time response and automated SDN mitigation.[3] The purpose is not simply to list previous studies, but to establish how the research problem has developed, what approaches have been validated, what assumptions are commonly made, and which limitations remain relevant to the proposed study. Recent systematic reviews show that SDN-DDoS research has developed from statistical and rule-based techniques towards ML, DL, ensemble and increasingly adaptive approaches, while Mininet-based experimental environments and datasets such as CICIDS2017, CSE-CIC-IDS2018 and InSDN remain common research resources. [4] [6]

The review is organized as a progression. It first explains why the limitations of conventional networks provide motivation for programmable SDN. It then examines DDoS threats against SDN, traditional detection approaches, and the development of ML/DL-based detection. The review subsequently examines datasets, generalization, previously unseen attack behavior, real-time detection, and adaptive mechanisms. Finally, it considers SDN-based mitigation and integrated detection-to-mitigation frameworks. A critical comparison is then used to position the proposed research. This is important because the proposed contribution is not simply another ML classifier: it evaluates an adaptive SDN defense pipeline in which traffic is observed, analyzed, classified and translated into a mitigation action under controlled live network conditions.

### 2.2 Network Vulnerabilities and Various Attacks

Modern computer networks are exposed to a wide range of vulnerabilities and cyberattacks because of increasing connectivity, complex infrastructures, and dependence on network services. Common network attacks include malware, phishing, Man-in-the-Middle (MitM) attacks, Distributed Denial-of-Service (DDoS), brute-force attacks, spoofing, and intrusion-based attacks. Vulnerabilities such as weak authentication, insecure configurations, outdated software, insufficient access controls, and poorly protected network services can provide attackers with opportunities to compromise systems or disrupt network operations.[4][8]

Among these attacks, DDoS was selected as the primary focus of this research because it directly targets network and service availability. During a DDoS attack, a large volume of malicious traffic or requests is generated towards a target, consuming network bandwidth, processing resources, flow-table capacity, or server resources.[4] As the attack intensity increases, legitimate users may experience increased latency, packet loss, reduced throughput, and service unavailability. These characteristics also make DDoS traffic suitable for flow-based analysis

because changes in packet rates, byte rates, flow counts, and other traffic characteristics can be observed at the network level.[1][8]

DDoS attacks can also present challenges for conventional security mechanisms because attack patterns can change over time, and large distributed traffic sources can make individual malicious flows difficult to distinguish from legitimate traffic. This research therefore combines supervised machine learning with behavioral anomaly detection to provide a broader detection approach.[6][9]

In addition to known DDoS patterns, the research considers zero-day or previously unseen attack behavior. A zero-day attack refers to malicious activity exploiting a vulnerability or exhibiting a pattern that is not adequately represented by existing signatures or training data. Such behavior may therefore be difficult for a supervised classifier to recognize. The behavioral and Isolation Forest component was included to identify significant deviations from established benign traffic patterns. This provides an additional layer of protection against novel network behavior and demonstrates how SDN can support adaptive detection and automated mitigation when network conditions change.[10]

### **2.3 Why SDN Was Selected over Traditional Networking,**

Traditional networks generally rely on distributed control, where individual routers and switches make forwarding decisions using their own control-plane mechanisms. Although this architecture is well established and reliable, implementing dynamic security policies across multiple devices can require device-specific configuration and manual intervention. This can make rapid and coordinated responses to high-volume attacks more difficult, particularly when network conditions change quickly.[2][11]

Software-Defined Networking (SDN) was selected for this research because its architecture provides a programmable and logically centralized control plane that is well suited to automated network security. By separating the control plane from the data plane, SDN allows the controller to collect network information, analyze traffic, and dynamically modify forwarding behavior without requiring manual configuration of individual switches. This capability is particularly relevant to DDoS detection and mitigation, where rapid identification of abnormal traffic and enforcement of blocking policies are important.[2][3][11]

Another reason for selecting SDN was its compatibility with machine-learning-based security mechanisms. The controller can receive flow-level information from the network, process relevant traffic features, and use the output of a machine-learning model to make security decisions. In this research, the OS-Ken controller was therefore used as the decision-making component between traffic monitoring and OpenFlow-based mitigation.[4][5]

SDN also provides greater flexibility for implementing and testing different security policies. Once malicious traffic is identified, the controller can dynamically install OpenFlow rules on an Open vSwitch to restrict the detected traffic. This creates an automated security workflow from traffic observation and classification to mitigation. In a traditional network, achieving a similar coordinated response may require configuration changes across multiple network devices.[3][4][5]

Finally, SDN was appropriate for this research because the Mininet and Open vSwitch environment provides a controllable experimental platform. It allowed different traffic conditions, attack scenarios, detection methods, and mitigation strategies to be tested repeatedly within a controlled environment. Therefore, SDN was chosen not simply as a replacement for traditional networking, but because its programmability, centralized control, automation, and OpenFlow-based enforcement directly support the objectives of this DDoS detection and mitigation research. [11]

## 2.4 DDoS Attacks and the SDN Security Problem

DDoS attacks attempt to reduce the availability of a service or network by consuming bandwidth, processing resources, connection state or application resources. Flooding attacks may use UDP, TCP, ICMP or application requests, while more subtle low-rate attacks attempt to exploit weaknesses in resource management without producing an obvious traffic-volume spike. The distributed nature of DDoS makes source-based filtering difficult because malicious traffic can originate from many hosts and may be mixed with legitimate traffic. [1][8][12]

Within SDN, the attack surface can be considered across applications, control, and data planes. The controller may be targeted directly; switches may experience flow-table pressure, and network links or services may be overwhelmed. Reviews of SDN DDoS research repeatedly identify controller exhaustion, bandwidth exhaustion, flow-table manipulation and control-plane overload as important concerns, . This makes the problem particularly suitable for an SDN-based defense experiment: the same programmable control mechanism that creates a vulnerability can also provide a mechanism for rapid mitigation.[8][9]

An important research distinction is therefore between detecting that traffic is malicious and preventing malicious traffic from degrading service. Detection may be evaluated with classification metrics, while mitigation must be evaluated in terms of whether the attack is suppressed and whether legitimate service remains available. A complete DDoS defense therefore consists of an observation stage, a detection stage, a decision stage, and a mitigation stage. The literature contains evidence for each of these stages, but the depth of evaluation varies considerably between studies [4][6][8]

## 2.5 Machine Learning-Based DDoS Detection

Machine learning provides a data-driven alternative to manually engineered detection rules. In supervised learning, labelled traffic samples are used to train a model that maps network features

to benign or attack classes. Algorithms frequently used in network intrusion and DDoS detection include Decision Trees, Random Forest, Support Vector Machines, K-Nearest Neighbors, Logistic Regression, Naive Bayes, boosting and neural networks. These algorithms are attractive because network traffic can be represented using flow-level statistical features such as packet counts, byte counts, duration, rates, and protocol information.[13][6]

A major strength of ML is its ability to learn nonlinear relationships that may be difficult to represent with a fixed rule. However, a high classification score does not automatically demonstrate robustness. The model can learn characteristics specific to a dataset, a particular attack generator, or the experimental environment. Consequently, the literature increasingly emphasizes not only accuracy but also dataset diversity, cross-dataset evaluation, class balance, false positives, and real-time applicability.[4][6] [9]

Demonstrated the feasibility of deep neural-network-based flow anomaly detection in SDN, using flow-level information available from the network. This work helped establish a foundation for applying learning-based security analytics within SDN. Later research increasingly connected learning with DDoS-specific detection and mitigation. DDoS detection and mitigation in SDN using ML, explicitly demonstrating the possibility of linking a detection result to an SDN response .[14][5]

The significance of these studies is therefore broader than their individual algorithms. They show that the SDN controller can become an integration point between network telemetry and automated security decisions. Nevertheless, they also illustrate the importance of evaluating the entire pipeline. A classifier operating offline on a dataset is fundamentally different from a classifier whose input is generated by a live/emulated SDN topology and whose output must result in a timely controller action.

The proposed research follows the second model. Its ML component is not treated as a complete research contribution. Instead, ML is one component within a pipeline that connects traffic features to SDN control and mitigation. This distinction will be maintained throughout the dissertation so that the contribution is positioned as an evaluation of adaptive defense behavior rather than merely an accuracy of competition between classifiers.

## 2.6 Deep Learning Approaches

Deep learning has become increasingly prominent because it can learn complex representations from high-dimensional traffic data. DNNs, CNNs, RNNs, LSTMs, and hybrid architectures have all been investigated for intrusion and DDoS detection. Deep models can potentially identify relationships that conventional algorithms may not capture, but they can also require greater computational resources and larger datasets. For a real-time SDN system, the computational cost of feature extraction and inference is therefore relevant to the operational feasibility of the model.

DNN-based DDoS detection and mitigation framework in an SDN environment and evaluated it using multiple datasets, including CICIDS2018 and InSDN . This is particularly relevant because it demonstrates a move from isolated detection towards a combined detection-and-mitigation framework. However, the existence of an integrated architecture does not resolve the generalization problem by itself. The research question remains whether the framework can maintain useful detection behavior when the attack distribution differs from the data used during training and whether the mitigation response is sufficiently timely.[7][14]

A 2024 CNN-based SDN study also investigated DDoS detection and mitigation, reflecting the broader trend towards deep learning in programmable networks. Recent systematic reviews similarly show a large number of ML/DL studies using Mininet and SDN controllers, with accuracy, precision, recall and F1-score as dominant metrics, . The implication is important for the present study: selecting another deep architecture would provide limited research value unless it addresses a clearly defined limitation in existing evaluation.[15][6][16]

Therefore, model complexity should be treated as a methodological choice rather than an assumed source of novelty. The proposed study should justify its selected model based on its suitability for the available data, detection task, and real-time pipeline. Its stronger academic contribution lies in how the model is evaluated against previously unseen behavior and how its decisions are translated into measurable network actions.

## 2.7 DDoS and SDN Datasets

Datasets strongly influence the conclusions that can be drawn from ML-based security research. CICIDS2017 was created to address limitations of older intrusion datasets and provides labelled benign and attack traffic with flow features. CSE-CIC-IDS2018 extends this line of work with a broader range of attacks and more recent traffic scenarios. InSDN was designed specifically for SDN intrusion research and provides traffic associated with multiple attack categories .[17][18]

The value of these datasets is reproducibility. Researchers can compare algorithms using a common benchmark, and the data can be processed repeatedly without the cost of generating every attack experiment from scratch. However, benchmark datasets also have limitations. They are generated under specific laboratory conditions, may contain class imbalance, and cannot fully represent the diversity of production traffic. InSDN is particularly useful for SDN research, but it is still an experimental dataset and therefore cannot be interpreted as a complete representation of all SDN deployments. [4] [18]

The literature increasingly distinguishes between general-purpose intrusion datasets and SDN-contextual datasets. This distinction matters for the proposed project. A benchmark dataset can support model development and controlled comparison, while Mininet-generated traffic can support system-level testing. Using both approaches allows the researcher to separate two questions: whether the model can learn useful traffic patterns, and whether the resulting detector can operate within a live SDN control loop.[6] [9]

This leads to an important methodological principle: dataset performance and deployment performance should not be treated as the same evidence. A model may achieve an excellent F1 score on CICIDS2017 while experiencing additional delay, false positives or degraded generalization when connected to a live/emulated SDN environment. The proposed evaluation therefore needs both ML-level and system-level measurements.[6][9][16]

## 2.8 InSDN Dataset

The InSDN dataset is a significant dataset for intrusion-detection research in Software-Defined Networking (SDN) because it was specifically developed to represent network traffic within an SDN environment. Unlike general-purpose intrusion datasets, InSDN was designed to address the limitations of datasets that do not adequately represent SDN-specific traffic characteristics. The dataset contains flow-level network information covering both benign and multiple attack categories, including traffic associated with DoS/DDoS attacks. This makes InSDN particularly relevant for research investigating machine-learning-based DDoS detection in SDN environments. Recent research has also used InSDN to evaluate ML-based detection approaches, further supporting its relevance to SDN security research .[18]

For the present research, InSDN is used as the principal dataset for developing the supervised DDoS detection model. The dataset is prepared for a binary classification task by distinguishing legitimate traffic from malicious traffic relevant to the research objective. Its flow-level structure is particularly suitable because the proposed detection framework operates using network-flow characteristics rather than packet payload inspection. Relevant traffic features can therefore be extracted and used to train the ML model before deployment in the experimental SDN environment. Data cleaning, feature preparation and separation of training and evaluation data are performed to support consistent model development. [18][17]

An important consideration is the transfer of knowledge from offline dataset-based training to live in SDN traffic. The live experimental environment must provide flow features compatible with those used during model training so that the trained classifier can operate consistently. This relationship between dataset-based learning and live traffic evaluation is particularly important for this research because the study does not focus only on offline classification accuracy. Instead, the InSDN-trained model forms part of an integrated detection and mitigation pipeline that is evaluated under live network conditions. Therefore, InSDN provides the experimental foundation for evaluating known DDoS behavior while supporting further investigation of generalization and real-time SDN-based defense[18][6]

## 2.9 Comparative Dataset Analysis,

| Dataset | Context | Strength | Main limitation | Best role | Relation to proposed work |
|---------|---------|----------|-----------------|-----------|---------------------------|
|         |         |          |                 |           | [17]                      |

|                           |                              |                                                |                                                 |                                |                                                             |
|---------------------------|------------------------------|------------------------------------------------|-------------------------------------------------|--------------------------------|-------------------------------------------------------------|
| CICIDS2017                | General IDS                  | Widely used; labelled flow features            | Not SDN-specific; laboratory distribution       | Model development              | Supports baseline ML evaluation                             |
| CSE-CIC-IDS2018           | General IDS                  | Broader and newer attack scenarios             | Offline benchmark; distribution-specific        | Robustness/cross-check         | [17]<br>Supports broader generalization testing             |
| InSDN                     | SDN-specific                 | Designed for SDN intrusion research            | Experimental and class-distribution limitations | SDN-focused model evaluation   | [18]<br>Relevant to SDN traffic characteristics             |
| Mininet-generated traffic | Controlled live/emulated SDN | Direct control of topology, attack and service | Emulation is not production scale               | End-to-end pipeline evaluation | [11]<br>Core environment for real-time detection/mitigation |

**Table 1 Dataset Comparison**

The comparison demonstrates why a hybrid evaluation strategy is academically useful. Public datasets provide reproducibility and benchmark evidence, whereas Mininet provides control over the live experimental pipeline. The proposed study can therefore evaluate the model under controlled traffic conditions without claiming that an emulation result is equivalent to production deployment.

## 2.10 Generalization and Previously Unseen DDoS Behaviors

Generalization is one of the most important unresolved issues in cybersecurity ML. In a conventional random train-test split, samples from the same attack category may occur in both training and testing. The resulting score measures how well the model recognizes samples from a broadly similar distribution, but it does not necessarily demonstrate that the model can recognize a behavior that was intentionally excluded from training. This distinction is central to DDoS research because attackers can vary rate, protocol, packet characteristics, and temporal patterns. [6], [10]

Recent reviews identify unknown and low-rate attacks as continuing challenges for ML-based SDN detection. A 2024 study using multi-dimensional characteristics notes that existing statistical methods can struggle with the dynamic nature of SDN and that ML approaches are increasingly explored to improve detection. A 2024 ensemble-learning framework similarly identifies unexpected DDoS patterns and the difficulty of detecting new or low-rate behavior as important challenges. [9][12][10]

For this dissertation, the term 'previously unseen DDoS behavior' is more precise than using 'zero-day attack' without qualification. A previously unseen behavior can be operationally defined by withholding a selected attack type, variant or traffic pattern from model training and using it for evaluation. This creates a measurable generalization experiment. It does not claim that the experiment reproduces an unknown real-world vulnerability exploit, which would be a stronger claim than the laboratory design can support.[10][11]

This distinction provides a strong methodological opportunity. If the model is trained on known DDoS behaviors and then evaluated on a deliberately withheld behavior, the study can report whether the detection pipeline generalizes beyond the training distribution. This is substantially stronger than reporting only a random split accuracy. Cross-dataset evaluation can provide a second level of evidence because the traffic distribution changes between datasets. [6] [18]

## 2.11 Real-Time DDoS Detection

Real-time detection introduces a temporal requirement. In an offline experiment, the researcher can calculate features and classify a batch without considering whether the result arrives quickly enough to protect a service. In a live or emulated SDN pipeline, traffic must be observed, statistics must be collected, features must be prepared, the ML model must infer a decision, and the controller must act. Each stage can contribute to delays.[3][7]

Recent work has explicitly examined real-time DDoS detection in SDN. Investigated low-rate DDoS detection using online machine learning, reflecting the need to move beyond static offline evaluation. Proposed real-time DDoS detection and mitigation for SDN-enabled smart-home networks using ML models deployed as SDN controller applications. Their study is particularly relevant because it demonstrates that detection can be executed as part of a controller-based operational system rather than only in a separate offline notebook. [19][20]

However, real-time studies do not always use the same definition of latency. Some report model inference time, some report processing time, and others report end-to-end response. These measurements should not be treated as equivalent. For the proposed research, detection latency should represent the time required for the system to move from observable malicious traffic to a detection decision, while mitigation response time should capture the subsequent interval required to apply the defensive action. Separating these measures will make it possible to identify where delay occurs within the pipeline. [20][21]

This is important because a system can be highly accurate but operationally ineffective if the decision arrives after the victim service has already suffered substantial degradation. Recent systematic reviews report increasing attention to detection latency and controller resource utilization, alongside conventional ML metrics. This supports the inclusion of operational metrics as a central part of the proposed evaluation.[6][7][20]

## 2.12 Adaptive Detection

The term adaptive is used in several different ways in the literature. A system may adapt to a threshold, update a model online, periodically retrain, dynamically select features, or change its mitigation policy. These mechanisms are conceptually different. Therefore, an academically defensible dissertation must define exactly what is meant by adaptation and how it is measured. [22]

The proposed study should treat adaptiveness as a measurable property rather than a descriptive label. The key question is whether the detection pipeline remains useful when traffic behavior changes or when a previously unseen DDoS behavior is introduced. Adaptation must also be considered alongside false positives and latency. An adaptive system that detects more attacks but significantly delays response or incorrectly blocks legitimate traffic may not provide an overall security improvement. [22]

This leads to a broader research perspective: robustness should be evaluated across attack variation, detection quality, and operational response. The combination is more meaningful than accuracy alone because the purpose of the system is to protect network/service availability rather than simply classify records.

## 2.13 SDN-Based Automated Mitigation

SDN provides the mechanism required to convert a security decision into an actual network action. When a flow is identified as malicious, the controller can install a rule to drop, rate-limit, redirect, or otherwise constrain the traffic. Rahman explicitly investigated ML-based DDoS detection and SDN mitigation. Hnamte also developed a DNN-based detection and mitigation framework. Garba demonstrated controller-integrated ML detection and mitigation in an SDN smart-home setting. [5][7][20]

Mitigation effectiveness should nevertheless be evaluated separately from detection effectiveness. A correct classification is not equivalent to successful traffic suppression. The controller must receive the decision, determine the relevant flow or source, install the rule, and influence the forwarding behavior. During this interval, attack traffic may continue to reach the victim. Furthermore, overly broad rules can affect legitimate users. [5][23]

This is why the proposed project will treat mitigation response time and network/service performance as explicit evaluation dimensions. The experiment can observe the protected Mininet web-server host before and during attack traffic, then measure the change after the mitigation rule is applied. This makes it possible to connect the ML decision to an observable network outcome. [7][20]

The use of Mininet and Open vSwitch is particularly appropriate for this purpose because the topology, switch behavior, traffic, and controller interaction can be controlled and repeated. The use of OS-Ken as the controller is an implementation choice for the proposed system; the

research's novelty should not be claimed merely from using OS-Ken. Instead, the controller provides the programmable interface through which the proposed detection and mitigation pipeline can be experimentally evaluated. [11][3]

## 2.14 Integrated ML for Detection and Mitigation

The most relevant body of literature combines ML detection with SDN mitigation. These studies establish that the field is moving beyond isolated classifier development. Nevertheless, the completeness of evaluation varies. Some studies focus mainly on classification performance; others demonstrate blocking or mitigation; others investigate real-time detection. Recent reviews confirm that these dimensions are often distributed across different research strands [4][6]

The proposed research therefore occupies the intersection of four requirements: adaptive detection, previously unseen behavior, timely SDN mitigation, and joint evaluation of network/service effects. Each requirement has precedent in the literature, but the combination is less consistently evaluated as one experimental question. This is the basis for a defensible research gap. [7][10] [20][22]

Importantly, the proposed system is not intended to claim that existing work has failed to perform DDoS mitigation. Existing frameworks already demonstrate successful mitigation. The stronger and more accurate claim is that there remains a need for systematic evaluation of how an adaptive ML detector behaves when exposed to deliberately unseen DDoS behavior and how quickly that decision becomes an effective SDN mitigation action while legitimate service performance is considered.[7][5][20]

This distinction will be central to the dissertation. It avoids overstating novelty while still establishing a clear contribution: the research evaluates the interaction between detection generalization and operational mitigation performance under controlled live SDN conditions.

## 2.15 Existing Research and the Proposed Research

| Dimension        | Typical existing studies                   | Proposed research                                                         | Why it matters                                                              |
|------------------|--------------------------------------------|---------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| Primary goal     | Often maximize classification accuracy     | Evaluate complete adaptive defense pipeline                               | [6] [9]<br>Moves from classifier performance to security-system performance |
| Training/testing | Often random split within a benchmark      | Known behaviors for training + deliberately withheld behavior for testing | [6][9]<br>Provides explicit generalization evidence                         |
| Environment      | Offline dataset or separate SDN experiment | Mininet + OVS + OS-Ken with live traffic                                  | [11] [3]<br>Connects ML decision to network action                          |

|                      |                                                  |                                                                     |                                                               |
|----------------------|--------------------------------------------------|---------------------------------------------------------------------|---------------------------------------------------------------|
|                      |                                                  | flow                                                                |                                                               |
| Attack behavior      | Known labelled attacks dominate                  | Previously unseen DDoS behavior is an explicit test condition       | [9] [10]<br>Tests robustness beyond memorization              |
| Detection            | Accuracy, precision, recall, F1                  | Same ML metrics + detection latency                                 | [6][9]<br>Measures whether detection is timely                |
| Mitigation           | Often demonstrated as blocking/rule installation | Automated controller-driven mitigation + response-time measurement  | [5] [7] [20]<br>Measures operational responsiveness           |
| Service impact       | Often secondary or not jointly reported          | Web-service/network performance before, during and after mitigation | [6] [9]<br>Tests whether defense preserves legitimate service |
| Overall contribution | Algorithm/framework-centric                      | Evaluation-centric adaptive SDN defence pipeline                    | Provides a clearer research question and measurable gap       |

**Table 2 Gap Comparison**

The table above is conservative. It does not claim that every existing study lacks capability. For example, some recent studies have already demonstrated real-time detection and mitigation, while others evaluate multiple datasets or dynamic mechanisms. The distinction is that the proposed research makes the intersection of these dimensions the evaluation focus. This is a stronger and more defensible positioning than claiming that ML-based DDoS detection or SDN mitigation is itself novel. [7][20][21] [22]

Two particularly strong comparison points emerge. First, "generalization versus conventional benchmark accuracy" existing studies frequently demonstrate excellent performance on known dataset distributions, whereas the proposed research explicitly treats previously unseen DDoS behavior as a test condition. Second, "classification versus end-to-end operational response" existing work often reports ML metrics or demonstrates mitigation separately, whereas the proposed study links detection latency, mitigation response time and network/service performance within one controlled SDN experiment. These two points should become the core of the eventual research contribution.

## 2.16 Critical Comparison of Representative Literature

| Ref. | ML/DL | Dataset | SDN | Real-time | Mitigation | Operational metrics | Key limitation for proposed gap |
|------|-------|---------|-----|-----------|------------|---------------------|---------------------------------|
|------|-------|---------|-----|-----------|------------|---------------------|---------------------------------|

|                       |               |                                   |     |                 |                       |                                        |                                                                                              |
|-----------------------|---------------|-----------------------------------|-----|-----------------|-----------------------|----------------------------------------|----------------------------------------------------------------------------------------------|
| [14]<br>Tang .        | DNN           | NSL-KDD                           | Yes | Limited         | Detection-<br>focused | Not central                            | Early SDN DL work; older dataset and limited end-to-end DDoS evaluation                      |
| [5]<br>Rahman         | ML            | SDN experiment                    | Yes | Operational     | Yes                   | Limited joint focus                    | Strong detection-to-mitigation foundation; unseen behavior not central                       |
| [7]<br>Hnamte         | DNN           | CICIDS2018, InSDN, others         | Yes | Framework-based | Yes                   | Not the main focus                     | Strong multi-dataset framework; joint generalization/latency/service evaluation remains open |
| [15]<br>CNN-based SDN | CNN           | Benchmark/SDN setting             | Yes | Framework       | Yes                   | Varies                                 | Shows DL mitigation capability; model-centric evaluation remains important                   |
| [12]<br>Alashhab      | Online ML     | Low-rate DDoS setting             | Yes | Yes             | Detection emphasis    | Latency/online focus                   | Strong real-time angle; not the full unseen-to-mitigation evaluation                         |
| [20]<br>Garba         | SVM/LR/DT/KNN | Real-world smart-home experiments | Yes | Yes             | Yes                   | Operational experiment                 | Strong real-time integrated framework; domain-specific setting and different research scope  |
| Proposed research     | Adaptive ML   | Benchmark + Mininet traffic       | Yes | Yes             | Yes                   | Latency + mitigation + service/network | Evaluates the complete pipeline under controlled live SDN conditions                         |

**Table 3 Critical Comparison**

## 2.17 Evaluation Metrics: From Classification to System Performance

Classification metrics remain essential. Accuracy provides an overall measure, but it can be misleading when benign traffic dominates. Precision measures the proportion of predicted attacks that are attacks, while recall measures the proportion of attacks detected. F1-score combines precision and recall. False-positive rates are particularly important in DDoS mitigation because an excessive number of false alarms can cause legitimate traffic to be blocked. These metrics are widely used across SDN DDoS research.[13], [6], [9]

However, the operational purpose of the system requires additional measures. Detection latency measures how quickly malicious activity is identified. Mitigation response time measures the interval between detection and application of the mitigation action. Throughput, packet loss, delay and service availability indicate whether legitimate traffic continues to receive acceptable

service. Controller CPU and memory utilization can also show whether the defense mechanism creates excessive overhead. [6] [9] [20]

The proposed evaluation therefore treats the system as a chain rather than a single classifier. The research question is not simply whether the ML model can classify traffic. It is whether a classification decision can be generated quickly enough and converted into a network action that reduces malicious traffic while maintaining acceptable service. This system-level framing is consistent with recent reviews that call for more attention to operational metrics and real-time validation[6][9]

## **2.18 Methodological Limitations Identified in Existing Research**

The literature reveals several recurring methodological issues. The first is dependence on benchmark datasets. Although public datasets are valuable for reproducibility, their traffic distributions may not represent live networks. The second is class imbalance and feature dependence. The third is the widespread use of random train-test splits, which may overestimate generalization when similar traffic patterns appear in both training and testing. The fourth is inconsistent definitions of real-time performance. Finally, mitigation and service impact are not always evaluated with the same depth as classification. [4] [6] [9] [18]

Another issue concerns the meaning of 'zero-day'. Some studies use unseen classes within a dataset as a proxy for zero-day detection. This can be useful experimentally, but it should not be presented as equivalent to discovering a completely unknown real-world exploit. Defining a previously unseen DDoS behavior as an attack condition withheld from training provides a clearer and reproducible research design. [10]

A further limitation is that the architecture itself can sometimes receive more attention than the research question. Many studies use Mininet, OpenFlow, and controllers such as Ryu or Open Daylight. The proposed use of Mininet, OVS, and OS-Ken should be positioned as an experimental platform. The contribution should instead be demonstrated by the evaluation design and evidence obtained from the complete detection-to-mitigation pipeline. [11] [3]

## **2.19 SDN Environment and Selection of OS-Ken, OpenFlow, OVS and Mininet**

Software-Defined Networking (SDN) is a networking architecture that separates the control plane from the data plane and enables network behavior to be managed programmatically through a centralized or logically centralized controller. This architecture was particularly suitable for this research because the proposed DDoS detection and mitigation system required continuous traffic monitoring, machine-learning-based decision making, and dynamic enforcement of security rules. The experimental SDN environment consisted primarily of the

OS-Ken controller, OpenFlow, Open vSwitch (OVS), and Mininet. Each component performed a different function within the overall architecture, allowing traffic collection, analysis, decision making, and mitigation to be integrated into a single experimental framework.[2] [3][11]

OS-Ken was selected as the SDN controller framework because it provides a Python-based environment for developing OpenFlow applications and was suitable for integrating the machine-learning components used in this research. The controller represents the control-plane intelligence of the proposed system. It receives network information from the switches, processes flow-level traffic information, applies to the Random Forest classification and behavioral detection mechanisms, and can subsequently issue OpenFlow commands to implement mitigation. A major consideration in selecting OS-Ken was practical compatibility with the research environment. The initial development considered the Ryu controller; however, compatibility and dependency issues encountered during implementation made OS-Ken a more practical choice for the final system. OS-Ken provides a Ryu-compatible development approach and supports OpenFlow-based SDN applications, allowing the project to maintain a Python-oriented controller architecture while avoiding the specific implementation difficulties encountered with the initial Ryu setup. Therefore, OS-Ken was selected based on its suitability for the experimental environment and its ability to support the required controller functionality rather than because it is universally superior to Ryu. [3]

Ryu is another widely used Python-based SDN framework designed for developing controller applications. It provides APIs for handling OpenFlow messages and managing programmable network devices. Ryu was initially considered for this research because of its Python-based architecture and suitability for SDN experimentation. However, practical implementation issues involving software dependencies and compatibility affected its use in the project environment. Moving to OS-Ken allowed the research to continue with a similar controller-development model while providing a more workable environment for the implemented application. This experience also demonstrated an important practical aspect of network research: the theoretical suitability of a technology does not always guarantee straightforward implementation, and compatibility with the wider software environment must also be considered. [3]

OpenFlow is the communication protocol that connects the SDN control plane with compatible forwarding devices. It allows the controller to communicate with switches and install, modify, or remove flow entries. In this research, OpenFlow 1.3 was used to allow the OS-Ken controller to control the forwarding behavior of the Open vSwitch. When the detection mechanism identified malicious traffic with sufficient confidence, the controller could generate an OpenFlow rule instructing the switch to drop matching traffic. OpenFlow therefore provided the enforcement mechanism connecting the ML detection decision with actual network-level mitigation. Without such a programmable interface, the controller would not be able to dynamically translate its security decisions into forwarding actions within the experimental network.[3]

Open vSwitch is a software-based, multilayer virtual switch designed to provide switching functionality within virtualized and programmable network environments. In the proposed architecture, OVS acted as the data-plane forwarding device. It received traffic from connected Mininet hosts and maintained flow entries that determined how packets should be forwarded or dropped. OVS was particularly appropriate for this research because it supports OpenFlow and can therefore be controlled directly by the OS-Ken controller. Its flow-table functionality also allowed the effects of mitigation rules to be inspected and evaluated using network-management tools. This provided an important link between the detection logic implemented in the controller and measurable forwarding behavior at the switch level. [3]

Mininet was used to create the experimental SDN environment in a controlled and repeatable manner. Mininet is a network emulation platform that can create virtual hosts, switches, links, and network topologies on a single machine. It was useful for this research because it allowed the proposed DDoS detection and mitigation architecture to be tested without requiring a large physical network infrastructure. Virtual hosts could be configured to generate legitimate and malicious traffic, while OVS provided the forwarding infrastructure and OS-Ken provided the control-plane functionality. A web server running on a Mininet host was used as the target service during testing, enabling the effect of attack traffic and subsequent mitigation to be observed. [11]

Overall, the combination of Mininet, OVS, OpenFlow, and OS-Ken provided an integrated SDN environment for the research. Mininet supplied the controllable network topology and virtual hosts, OVS performed packet forwarding, OpenFlow provided the communication and control interface, and OS-Ken implemented the security intelligence and mitigation logic. This layered environment allowed machine-learning-based traffic classification and behavioral analysis to be connected directly to programmable network enforcement. As a result, the experimental platform supported the complete research workflow from traffic generation and flow collection through detection and decision making to automated mitigation and performance evaluation. [3][11]

## 2.20 Selection of Random Forest

Random Forest (RF) is selected as the primary machine-learning classifier. Random Forest is an ensemble method that constructs multiple decision trees using different subsets of the data and combines their predictions. This ensemble process can improve predictive performance and help control overfitting compared with relying on a single decision tree. [24]

RF is particularly suitable for this research because the InSDN dataset contains structured flow-level features with potentially nonlinear relationships between traffic characteristics and attack labels. The model can handle interactions among multiple features without requiring extensive feature transformation normally associated with some other modelling approaches. It also provides a practical inference process for integration with a live detection pipeline. [24][18]

Another reason for selecting RF is methodological practicality. The research must evaluate not only classification quality but also detection latency and mitigation response time. A model with a comparatively straightforward training and inference process is therefore preferable to a substantially more complex deep-learning architecture when the research objective is to evaluate the complete adaptive SDN defense pipeline. RF is consequently used as the main classifier, while its performance is assessed using precision, recall, F1-score, false-positive rate, and operational timing metrics. [24][6]

## 2.21 Python and Supporting Libraries

Python is used as the principal programming language because the complete research workflow requires data processing, machine-learning development, and integration with an SDN controller. Python provides a mature ecosystem for scientific computing and machine learning, while OS-Ken itself is a Python-based SDN framework. Using Python therefore reduces the need to bridge between unrelated programming environments and makes it practical to connect the data-processing and detection components with the controller application. [25]

Scikit-learn is used for machine-learning development because it provides implementations of supervised learning algorithms, preprocessing, model fitting, prediction and evaluation within a consistent Python API. In particular, Random Forest Classifier provides the RF implementation used by the proposed detector. Python-based data-processing tools are used to prepare the InSDN records and convert the selected features into the input format required by the model. Where numerical array operations are required, NumPy can be used as the underlying numerical representation, while data-frame processing can be handled with pandas. These libraries are appropriate because they integrate directly with the Scikit-learn workflow.[25] [24]

Python was preferred to implement the ML component in a lower-level language because the research requires rapid experimentation and integration rather than maximum computational performance from the programming language itself. Python's extensive scientific and ML ecosystem, together with the Python-based OS-Ken controller, provides a coherent development environment. This is particularly useful when the research needs to move from offline dataset preparation to live model inference and SDN mitigation.

## 2.22 Why Machine Learning Rather Than Deep Learning

Deep-learning approaches have been applied to SDN intrusion and DDoS detection, including recurrent neural networks, LSTM and GRU models evaluated with the InSDN dataset. However, deep learning was not selected as the primary approach in this research because the research objective is to develop and evaluate an adaptive real-time detection-and-mitigation pipeline rather than to maximize model complexity.[14] [18]

The selected problem is based on structured, tabular flow-level network features. A conventional supervised machine-learning model is therefore appropriate because it can learn relationships among flow statistics without requiring the substantially more complex training architecture associated with deep neural networks. For this project, the additional training complexity of deep learning is not necessary to answer the research questions, while a conventional ML model provides a practical balance between detection capability, training cost, and inference efficiency.[9][11]

The decision is also consistent with the end-to-end nature of the research. The ML model is only one stage of the pipeline; the overall system must also collect traffic features, communicate with the SDN controller, and generate a timely mitigation response. Consequently, the methodology prioritizes an effective and comparatively lightweight classifier that can be integrated into the real-time SDN control loop. Deep learning remains relevant as an alternative approach and is therefore discussed as part of the literature context rather than used as the primary implementation.[7]

### 2.23 System Architecture and Network Topology

The proposed architecture separates the SDN control plane from the data plane. Mininet and OVS form the data-plane environment, while OS-Ken operates as the controller. The controller communicates with the OpenFlow-enabled switch and provides the control mechanism through which forwarding behavior can be changed. [2] [3] [11]

The architecture was initially considered with the Ryu controller; however, OS-Ken was selected for the final implementation. OS-Ken is a component-based SDN framework maintained within the OpenStack ecosystem and is explicitly described as a fork of the Ryu framework tailored for OpenStack Neutron. This choice provided a practical continuation of the Python/OpenFlow programming model while avoiding dependence on the original Ryu implementation. OS-Ken supports OpenFlow communication with OVS, allowing the controller to receive network events and install flow rules. Therefore, the change from Ryu to OS-Ken is a deliberate implementation decision rather than a change in the research objective. [3]

Traffic from the Kali attacker and normal user is directed towards the protected Mininet web server at 10.0.0.1. Relevant flow information is processed by the detection pipeline. If traffic is classified as malicious, the resulting security decision is communicated to OS-Ken, which applies to the configured mitigation policy through an OpenFlow rule.

### 2.24 Research Gap

The literature demonstrates that ML-based DDoS detection is well established; SDN provides programmable mitigation, and several recent studies have moved towards real-time and integrated detection-and-mitigation frameworks. Therefore, the research gap should not be stated

as 'there is no ML-based DDoS detection in SDN' or 'existing systems cannot mitigate DDoS'. Such statements would be inaccurate. [4] [5][6] [7] [20] [22]

Instead, the critical gap is an "evaluation gap". Existing studies frequently emphasize classifier performance on known datasets, while other studies demonstrate real-time detection or SDN mitigation. Recent research increasingly includes latency and system-level metrics, but the dimensions are not consistently evaluated together under a common experimental design, . Explicit testing against previously unseen DDoS behavior and the subsequent measurement of how quickly the detection decision becomes an effective mitigation action remain in important areas for systematic evaluation. [6][9][10][20]

The first strong point of the gap is therefore "unseen behavior generalization". A model that performs well on known attack classes does not automatically demonstrate robustness to behavior excluded from training. The proposed research will make this a deliberate experimental condition rather than an implied claim. [10][6]

The second strong point is "end-to-end operational responsiveness". Detection latency and mitigation response time are different events, and both matter. A detector may be accurate but slow; a mitigation mechanism may be effective but impose service degradation. The proposed research will evaluate these dimensions together with network/service performance, creating a direct link between ML intelligence and practical SDN defense.

Accordingly, the research gap can be formulated as follows: "Although existing studies have demonstrated machine-learning-based DDoS detection, real-time detection, adaptive classification and SDN-based mitigation individually or in combination, a significant evaluation gap remains in establishing whether an adaptive ML-based SDN defense framework can reliably identify previously unseen DDoS behavior and translate the detection decision into timely SDN mitigation under live network conditions. Existing studies also frequently emphasize classification accuracy, with comparatively less emphasis on jointly evaluating unseen-attack generalization, detection latency, mitigation response time, and network/service performance." [4] [6][7][10][20]

This formulation is deliberately aligned with the proposed experimental architecture. Kali Linux provides the controlled attacker environment; Mininet provides the SDN topology; OVS provides programmable forwarding; OS-Ken provides controller-based monitoring and mitigation; the ML component provides the detection intelligence; and the Mininet web-server host provides the protected service. The resulting pipeline can therefore be evaluated from attack generation through detection and mitigation to service outcomes.

## 2.25 Evaluation Matrix - Accuracy, F1-Score and Latency in DDoS Detection

Machine-learning-based DDoS detection studies commonly evaluate classification performance using accuracy, precision, recall, and F1-score, while SDN-based systems additionally need to consider detection and mitigation latency. Accuracy represents the proportion of correctly classified traffic instances among all evaluated instances. However, accuracy alone may provide an incomplete picture when benign and attack classes are imbalanced. For this reason, precision and recall are also considered, where precision reflects the proportion of predicted attacks that are malicious, and recall indicates the proportion of actual attacks that are successfully detected. The F1-score provides a single measure that balances precision and recall through their harmonic mean and is therefore useful for evaluating DDoS classifiers where both false positives and missed attacks are important [6], [13].

Existing SDN security research demonstrates the use of these measures for evaluating machine-learning-based detection. Rahman et al. investigated DDoS detection and mitigation in SDN using machine learning, linking detection performance with the practical mitigation process [5]. Automated DDoS detection research has also evaluated machine-learning approaches for identifying malicious traffic in SDN environments [19]. More recent studies have examined DDoS detection using deep learning, feature selection, adaptive detection and hybrid machine-learning approaches, further demonstrating the importance of reporting multiple classification metrics rather than relying only on accuracy [7], [16], [21]. Random Forest is also an established ensemble learning method suitable for classification tasks and provides the basis for evaluating classifier performance in the proposed framework [24].

Latency is particularly important in a real-time SDN defense system because accurate classification is not sufficient if the security decision is produced too slowly. Detection latency can be evaluated as the elapsed time between the observation or collection of relevant network-flow information and the generation of the attack-detection decision. Mitigation response time can then be measured from the detection decision until the SDN controller installs or enforces the mitigation rule. Recent work on adaptive and real-time-oriented DDoS detection highlights the importance of timely response in addition to classification performance [20], [22]. Therefore, the literature supports a combined evaluation approach in which accuracy and F1-score measure detection effectiveness, while detection latency and mitigation response time assess operational responsiveness. This approach is particularly relevant to SDN because programmable control enables a detected malicious flow to be acted upon through controller-driven network rules [2], [3].

## 2.26 Summary

The literature review has established the theoretical and empirical basis for investigating adaptive ML-based DDoS detection and mitigation in SDN. Conventional networks can make coordinated security response difficult, while SDN provides centralized programmability that

can support automated defence . At the same time, SDN introduces control-plane and controller vulnerabilities that make DDoS defence an important research problem [2][3] [4][8][9]

ML and DL have demonstrated strong detection capability, and datasets such as CICIDS2017, CSE-CIC-IDS2018 and InSDN have enabled reproducible experimentation. However, benchmark accuracy does not by itself demonstrate generalisation or operational performance. Recent work has therefore moved towards online detection, adaptive thresholds, integrated mitigation and system-level metrics [6], [7], [17], [18], [19], [21], [22]

The critical synthesis indicates that the strongest research opportunity is not another standalone classifier. It is a controlled evaluation of an adaptive ML-to-SDN defense pipeline against previously unseen DDoS behavior, measuring not only classification quality but also detection latency, mitigation response time, and network/service performance. This positioning provides a logical foundation for the next sections of the dissertation: research gap, aim, objectives, research questions, and methodology.

### **3. METHODOLOGY**

#### **3.1 Research Philosophy**

This research follows a realist ontology, as the study assumes that network traffic and DDoS attacks are real phenomena that can be objectively observed and measured. The research focuses on measurable characteristics of network flows, such as packet rate, byte rate, flow duration, and packet count. These characteristics exist independently of the researcher's personal opinions or interpretations. Therefore, the research considers network behaviors as an objective reality that can be analyzed using quantitative measurements and machine learning techniques. [13]

#### **3.2 Data Collection**

This research uses both secondary and primary data. The secondary data is obtained from the INSDN dataset, which provides existing records of normal and DDoS traffic for ML model training and reference analysis. The primary data is generated and collected directly from the Mininet-based SDN testbed using Open vSwitch, where normal and controlled DDoS traffic are captured as live flow-level data for evaluating the proposed detection and mitigation framework. [18], [11], [3]

#### **3.3 Research Methodology**

This research follows a quantitative and experimental methodology to develop and evaluate an SDN-based DDoS detection and mitigation system. A Random Forest machine learning model is trained using features derived from the INSDN dataset. The proposed system is then implemented in a controlled Mininet and Open vSwitch environment using an OS-Ken SDN controller. Both normal and attack traffic are generated and monitored to evaluate the detection capability of the system. An additional anomaly detection mechanism is used to identify previously unseen or zero-day traffic behavior. The performance of the proposed approach is evaluated using metrics such as accuracy, precision, recall, F1-score, and detection/mitigation behaviors. [6], [7], [11], [18], [24], [25]

#### **3.4 How the Research Will Be Conducted**

First, the INSDN dataset will be collected and preprocessed to remove unnecessary data and extract relevant network traffic features. Next, a Random Forest machine learning model will be trained using these features to classify normal and DDoS traffic. The trained model will then be integrated with an OS-Ken SDN controller in a Mininet and Open vSwitch environment. Normal and controlled DDoS traffic will be generated in the SDN testbed, while flow information will be collected from the network. The proposed system will analyze the live traffic using both the machine learning model and behavioral/anomaly detection techniques to identify DDoS and

previously unseen traffic patterns. When malicious traffic is detected with sufficient confidence, the controller will automatically install OpenFlow drop rules in the switch to mitigate the attack. Finally, the system will be evaluated using detection performance, including accuracy, precision, recall, F1-score, detection behavior, and mitigation effectiveness. [3], [6], [11], [18], [24], [25]

## 4. DESIGN AND IMPLEMENTATION

### 4.1 Overview

This chapter presents the detailed design and implementation of the proposed adaptive AI-based DDoS detection and mitigation framework. The methodology defined the experimental approach, InSDN dataset selection, machine-learning strategy, and evaluation criteria. This chapter translates those methodological decisions into an operational SDN security system.

The implemented framework integrates a Mininet-based SDN network, Open vSwitch (OVS), the OS-Ken controller, a Random Forest (RF) machine-learning classifier and external traffic sources. The protected web service is hosted as a Mininet virtual host with IP address 10.0.0.1. Kali Linux, using 192.168.56.102, acts as the controlled attacker, while the normal-user host uses 192.168.56.103 to generate legitimate traffic. The OS-Ken controller operates at 192.168.56.101.

The implementation is designed as a sequential detection and mitigation pipeline:

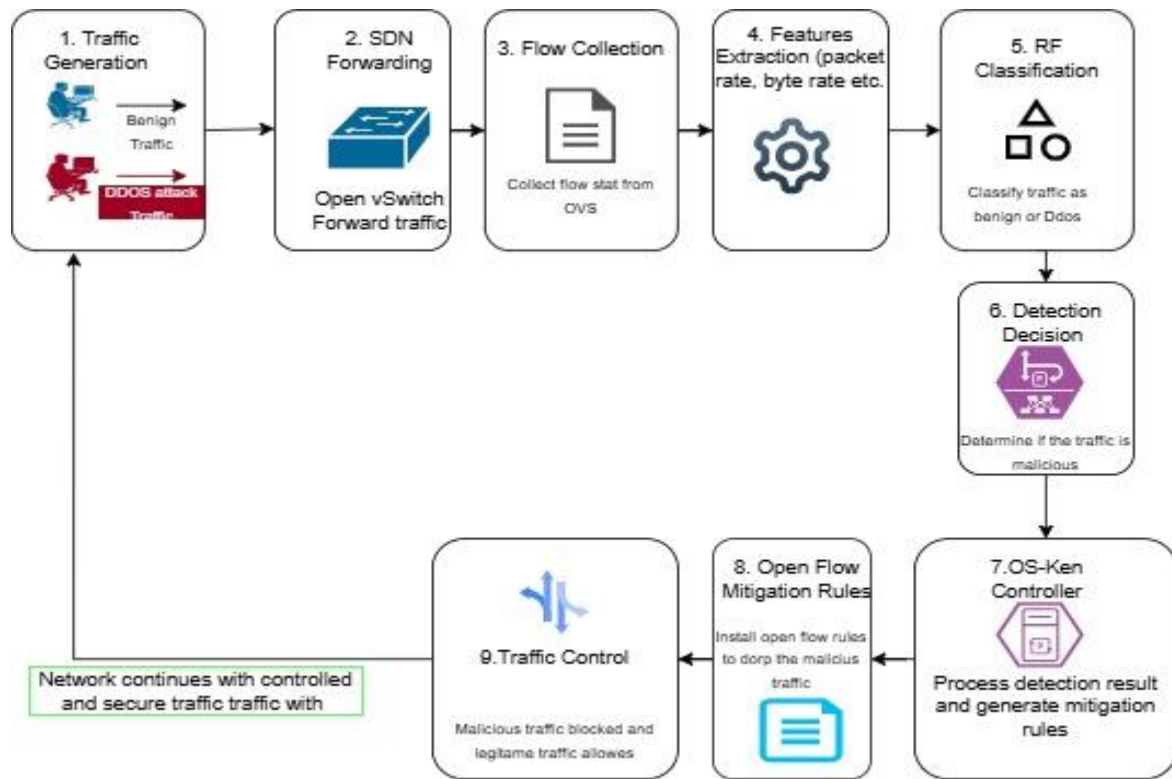


Figure 1 Sequential detection and mitigation pipeline

### 4.2 SDN Network Topology Design

The SDN topology is implemented using Mininet. Mininet provides the emulated hosts, links, and switching environment, while OVS provides the programmable software switching layer. The protected web-server host is placed inside the Mininet topology and assigned IP address

10.0.0.1. This design ensures that the service is directly connected to the SDN data plane and that the controller can influence its traffic path.

The attacker and normal-user systems remain in external traffic sources. Kali Linux at 192.168.56.102 is used for controlled DDoS traffic generation, whereas 192.168.56.103 represents legitimate user activity. The controller at 192.168.56.101 manages the OVS data plane using OpenFlow.

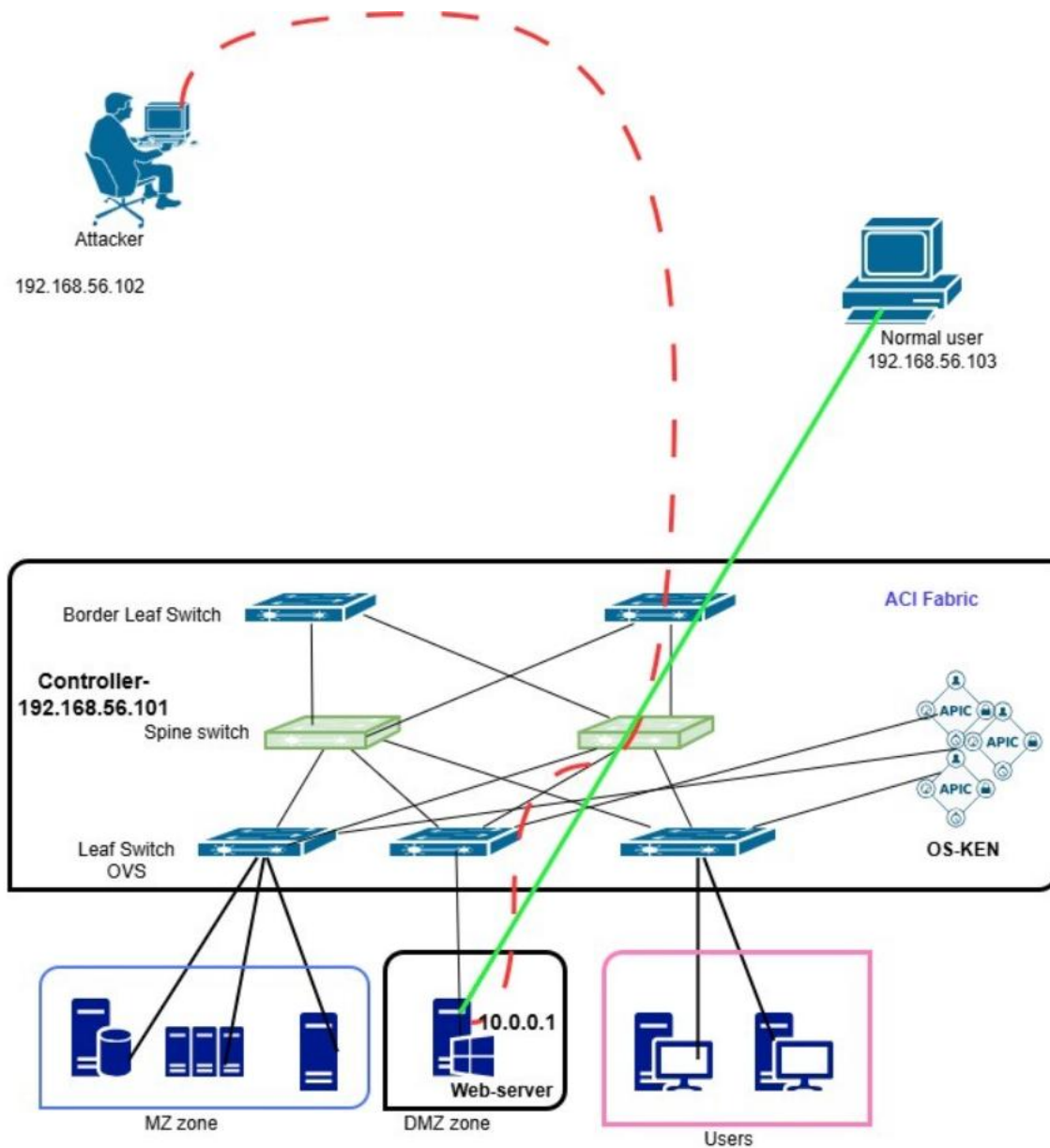


**Figure 2 Mininet/OVS topology and IP addressing are used in the experimental environment.**

Before security experiments, basic connectivity and web-service accessibility are verified. This ensures that later measurements reflect the behavior of the security framework rather than an underlying topology or configuration problem.

### 4.3 Web-Service Implementation

The protected web service is implemented as a Mininet virtual host rather than as a separate Ubuntu web server. The host uses IP address 10.0.0.1 and acts as the target service for both legitimate and malicious traffic.

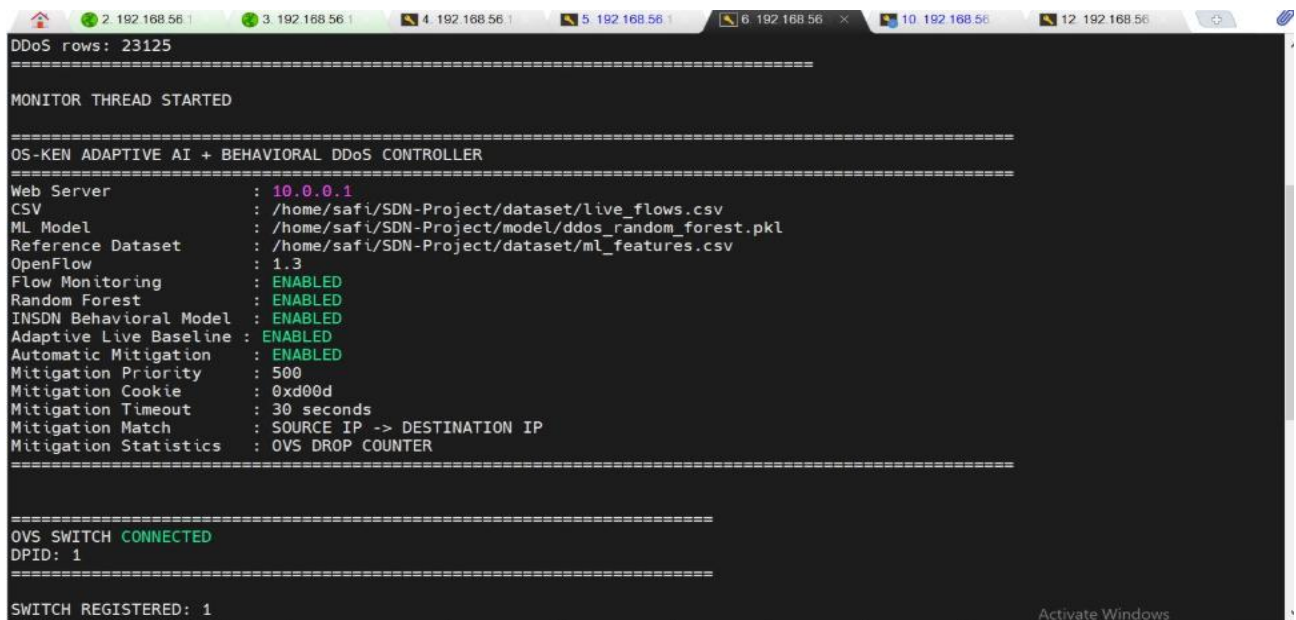
The service is started within the Mininet environment, and its accessibility is checked from the normal-user and attacker environments. This arrangement provides a consistent service endpoint for measuring availability, responsiveness, and network impact during normal traffic, DDoS traffic, and post-mitigation conditions.

### 4.4 OS-Ken Controller Implementation

OS-Ken is implemented as the central SDN controller at 192.168.56.101. The controller communicates with the OVS switching layer through OpenFlow and provides the control logic required by the security framework.

The controller performs several functions: establishing communication with the switch, receiving relevant switch events, obtaining flow-level information, interacting with the detection component, processing security decisions and installing appropriate flow rules.

The implementation uses OS-Ken instead of the originally considered Ryu controller. OS-Ken provides a compatible Python-based SDN programming model and OpenFlow support while fitting the final experimental environment.



```
DDoS rows: 23125
=====
MONITOR THREAD STARTED
=====
OS-KEN ADAPTIVE AI + BEHAVIORAL DDoS CONTROLLER
=====
Web Server      : 10.0.0.1
CSV             : /home/safi/SDN-Project/dataset/live_flows.csv
ML Model        : /home/safi/SDN-Project/model/ddos_random_forest.pkl
Reference Dataset : /home/safi/SDN-Project/dataset/ml_features.csv
OpenFlow        : 1.3
Flow Monitoring  : ENABLED
Random Forest    : ENABLED
INSDN Behavioral Model : ENABLED
Adaptive Live Baseline : ENABLED
Automatic Mitigation : ENABLED
Mitigation Priority : 500
Mitigation Cookie   : 0xd00d
Mitigation Timeout  : 30 seconds
Mitigation Match    : SOURCE IP -> DESTINATION IP
Mitigation Statistics : OVS DROP COUNTER
=====
=====
OVS SWITCH CONNECTED
DPID: 1
=====
SWITCH REGISTERED: 1
Activate Windows
```

**Figure 3 OS-Ken controller interaction with the OVS data plane and OpenFlow control path**

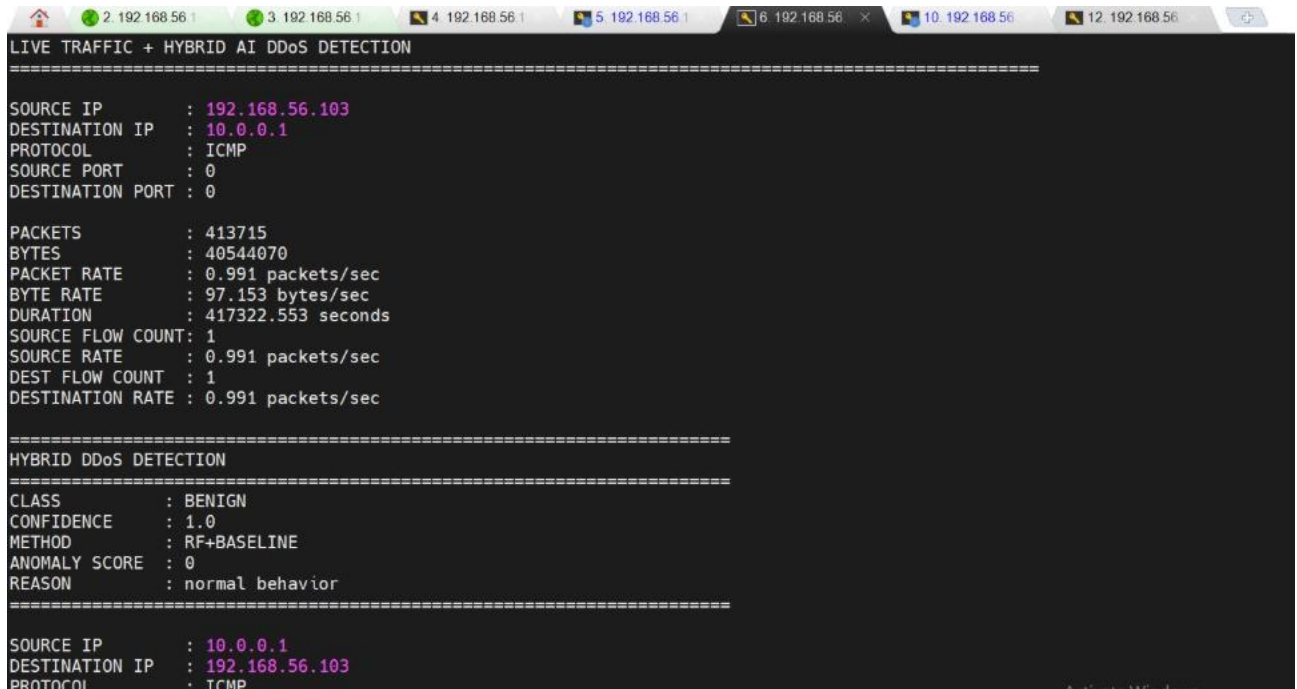
This figure shows the output obtained after successfully running the OS-Ken controller and establishing a connection with the OVS-based Mininet switch. Therefore, communication between the SDN controller and the Mininet SDN environment has been successfully established.

#### 4.5 Flow Monitoring and Feature Extraction

A key implementation requirement is obtaining traffic information in a form that can be processed by the trained machine-learning model. The SDN switch maintains flow-level information that can be queried through the controller. The implementation collects the required flow statistics and transforms them into the feature representation expected by the Random Forest model.

The general processing sequence is OVS Flow Statistics → Flow Information Collection → Feature Extraction → Feature Pre-processing → Random Forest Input.

Relevant characteristics may include packet counts, byte counts, flow duration, packet rate, and other traffic statistics represented in the selected InSDN feature space. The live features must correspond to the feature structure used during model training so that the classifier receives compatible input.



```
LIVE TRAFFIC + HYBRID AI DDoS DETECTION
=====
SOURCE IP      : 192.168.56.103
DESTINATION IP : 10.0.0.1
PROTOCOL       : ICMP
SOURCE PORT    : 0
DESTINATION PORT : 0

PACKETS        : 413715
BYTES          : 40544070
PACKET RATE    : 0.991 packets/sec
BYTE RATE      : 97.153 bytes/sec
DURATION       : 417322.553 seconds
SOURCE FLOW COUNT : 1
SOURCE RATE    : 0.991 packets/sec
DEST FLOW COUNT : 1
DESTINATION RATE : 0.991 packets/sec

=====
HYBRID DDoS DETECTION
=====
CLASS          : BENIGN
CONFIDENCE     : 1.0
METHOD         : RF+BASELINE
ANOMALY SCORE  : 0
REASON         : normal behavior
=====

SOURCE IP      : 10.0.0.1
DESTINATION IP : 192.168.56.103
PROTOCOL       : ICMP
```

**Figure 4 Flow-statistics collection and feature-extraction pipeline.**

In here it is shown that after capture the packet which features are extracted.

## 4.6 InSDN Dataset Processing

The InSDN dataset is used to develop the machine-learning detection component. The dataset contains SDN-oriented network traffic and attack information and is therefore aligned with the SDN environment used in this research.

The dataset is loaded into the Python processing environment and prepared through data cleaning and feature processing. Relevant features are selected according to their availability and compatibility with the live SDN flow-monitoring process. The resulting records are transformed into model inputs and corresponding traffic labels.

The prepared data is divided into appropriate training and evaluation subsets. The training data is used to construct the Random Forest model, while independent evaluation data is retained for testing. Selected attack behavior is also reserved for the previously unseen attack experiment so that the evaluation is not simply a random repetition of the training distribution.

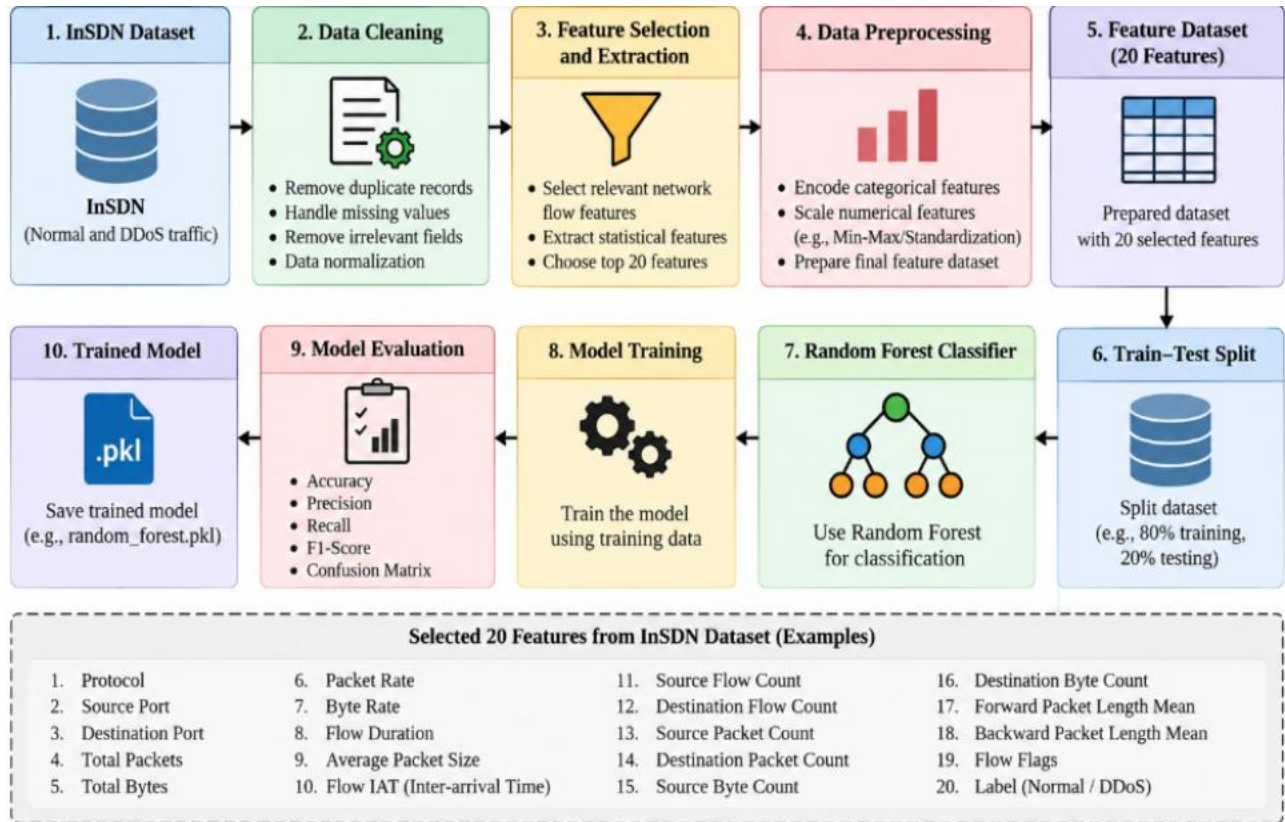


Figure 5 InSDN data-preparation and model-training pipeline.

## 4.7 Random Forest Model Implementation

The Random Forest classifier is implemented using Scikit-learn. The model learns relationships between the selected network-flow features and traffic labels. Random Forest is appropriate for

the structured, tabular flow-level data used in this research and provides an ensemble of decision trees for classification.

The trained model is retained for live inference rather than retrained during every experiment. This separates model development from real-time detection and provides a consistent classifier for the experimental scenarios.

```
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# python3 validate_model.py

=====
MODEL VALIDATION
=====

Records: 37079
Features: 20
BENIGN: 13954
DDoS: 23125

Existing Random Forest loaded successfully.

=====
5-FOLD CROSS VALIDATION
=====

-----
CROSS-VALIDATION PERFORMANCE
-----
Accuracy : 91.78% +/- 0.23%
Precision : 91.24% +/- 0.25%
Recall : 91.27% +/- 0.27%
F1-Score : 91.25% +/- 0.24%

-----
FOLD RESULTS
-----
Fold 1: Accuracy=91.86% | Precision=91.20% | Recall=91.53% | F1=91.36%
Fold 2: Accuracy=92.08% | Precision=91.57% | Recall=91.57% | F1=91.57%
Fold 3: Accuracy=91.92% | Precision=91.45% | Recall=91.32% | F1=91.38%
Fold 4: Accuracy=91.60% | Precision=91.09% | Recall=90.99% | F1=91.04%
```

**Figure 6 Random Forest training, validation, and model-loading workflow.**

## 4.8 Python-Based Detection Component

Python is used as the integration layer between data preparation, machine-learning inference, and SDN security logic. Scikit-learn provides the Random Forest implementation, and the Python data-processing environment prepares the live input features.

The detection component receives flow-level features, converts them into the same feature structure used during training, and passes them to the trained Random Forest model. The classifier then generates traffic prediction.

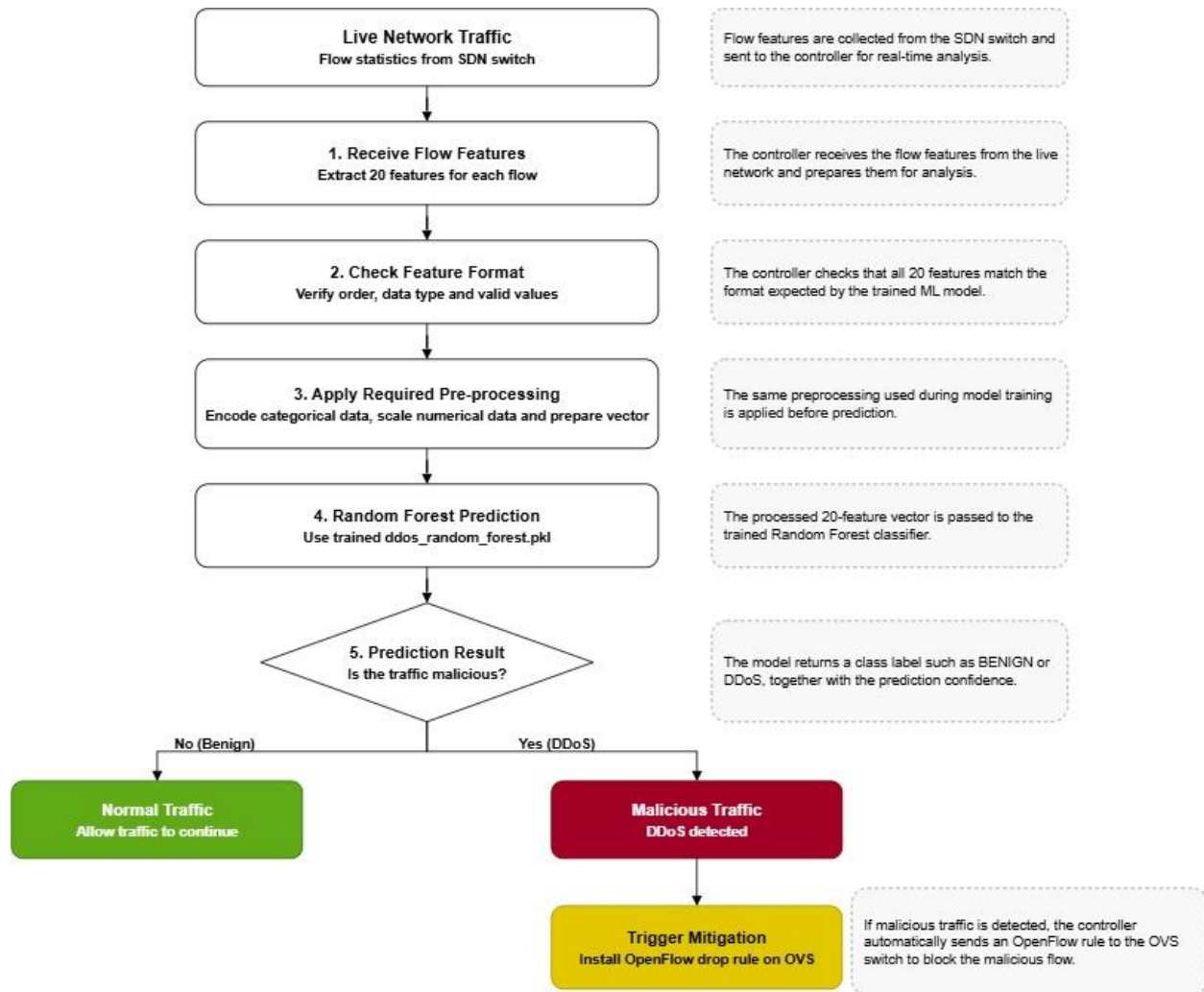


**Figure 7 Machine-learning detection decision process.**

#### 4.9 Detection Decision and Controller Integration

When the Random Forest classifier produces a malicious prediction, the security decision is passed to the SDN control logic. OS-Ken identifies the relevant traffic flow and applies to the configured security policy.

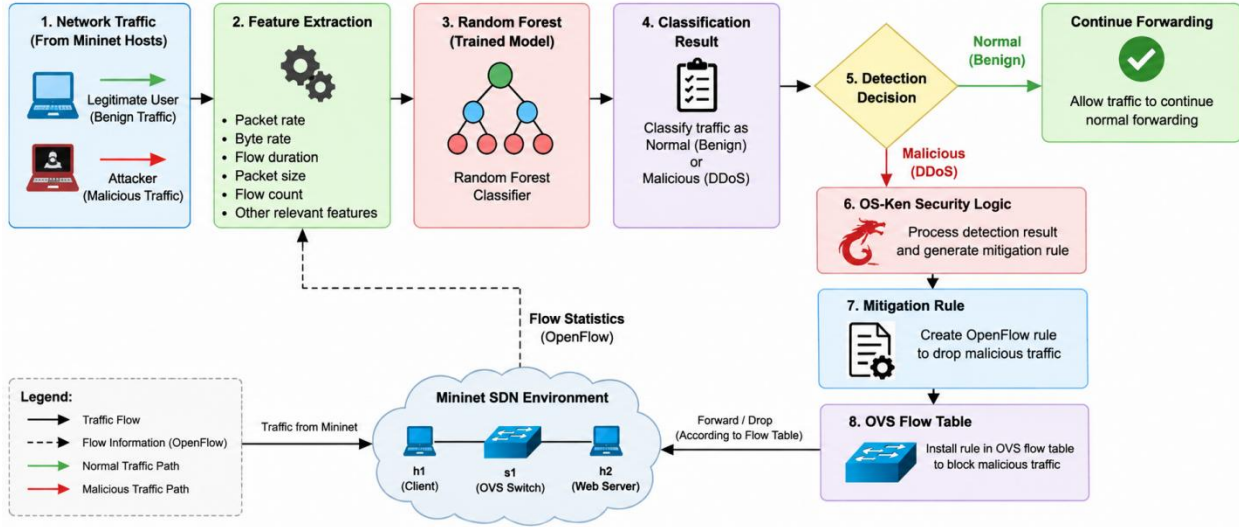


**Figure 8 Integration Process**

This integration is central to the implementation because it converts an AI classification into an actual network-level security response rather than merely generating an alert.

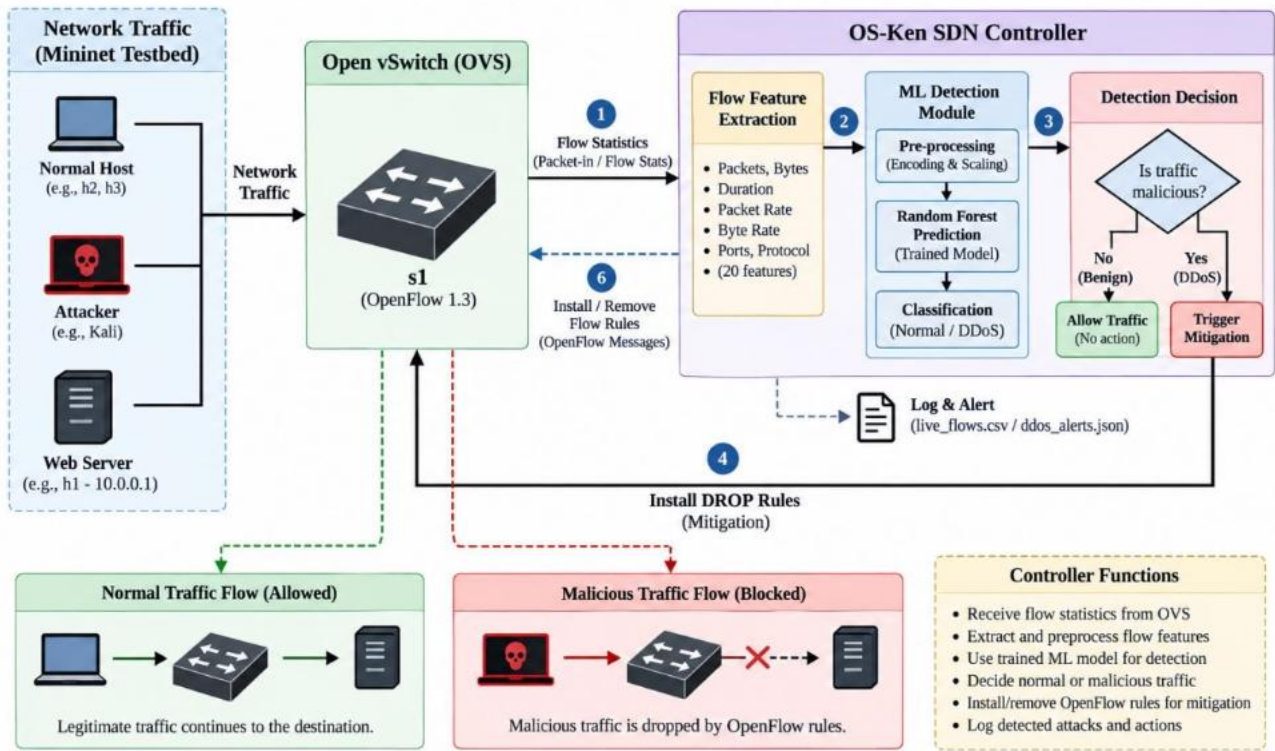


**Figure 9 ML detection-to-OS-Ken controller integration.**

## 4.10 SDN-Based Mitigation Implementation

When malicious traffic is identified, OS-Ken generates an appropriate OpenFlow rule. The rule is installed on the relevant OVS switch to control subsequent packets belonging to the detected malicious flow.

The mitigation mechanism operates at the SDN data-plane level. Depending on the configured policy, malicious traffic can be blocked or restricted using an appropriate flow action. This enables the research to measure the time between detection and effective network-level response.

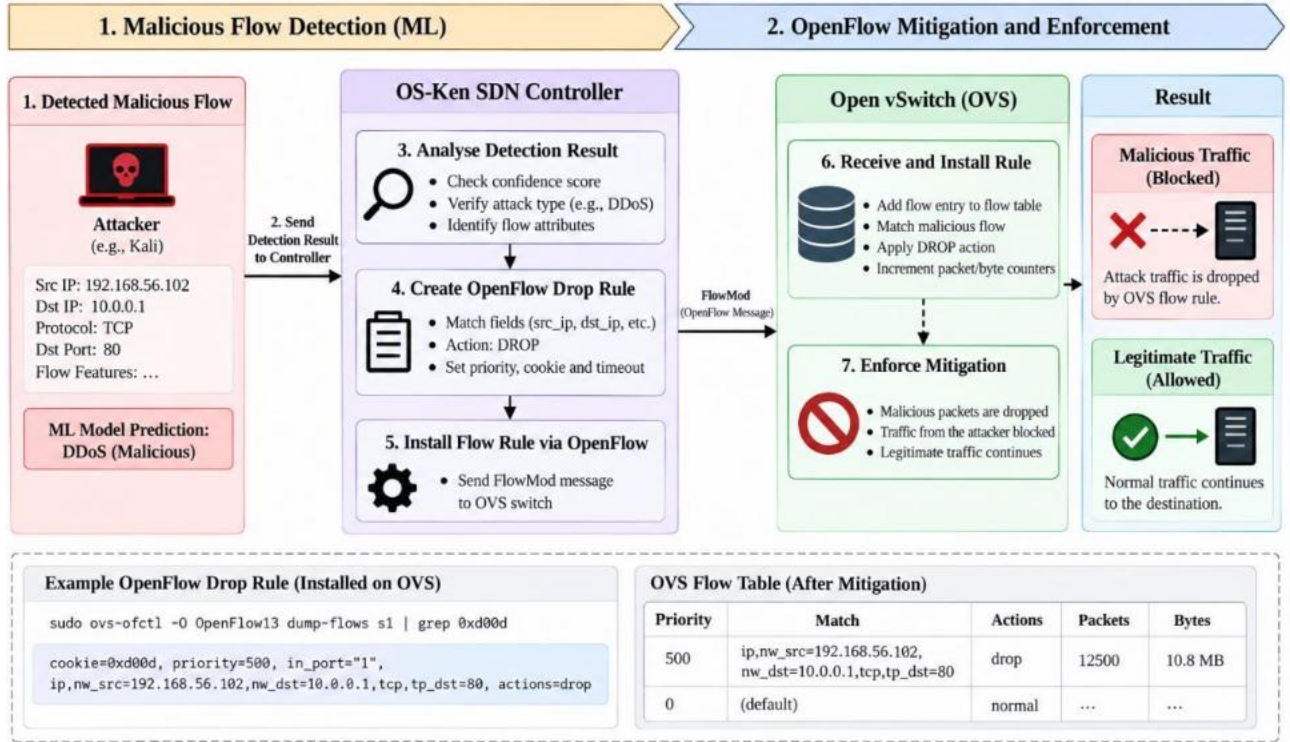


Figure 10 OpenFlow mitigation process from malicious-flow detection to OVS flow-rule enforcement.

## 4.11 End-to-End Implementation Workflow

The complete system combines the network, ML detector, and controller-based mitigation into a single operational pipeline. The workflow begins with controller and topology initialization, followed by service activation and traffic generation. Flow statistics are then collected and converted into model features. The Random Forest classifier generates the detection decision, and malicious traffic triggers OS-Ken-based mitigation.

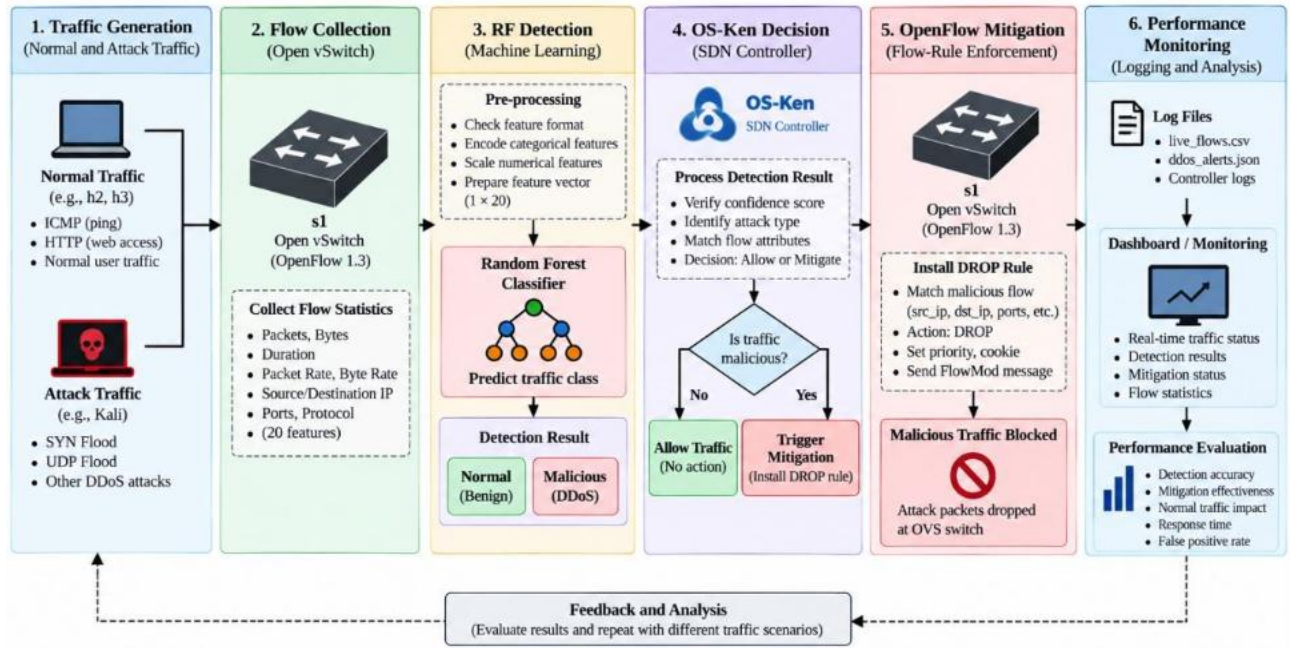


**Figure 11 End-to-end implementation flowchart**

The end-to-end workflow demonstrates that the implementation is not limited to offline classification. The trained classifier operates within an SDN security loop in which detection can result in an actual network-control action.

#### 4.12 Previously Unseen Attack Implementation

The previously unseen attack experiment is implemented by excluding selected attack behavior from the model-training process and introducing that behavior during evaluation. The same feature-extraction and classification pipeline is used for both known and unseen traffic.

If the unseen traffic is classified as malicious, the decision is passed to OS-Ken, and the configured mitigation mechanism is applied. This allows the framework to be assessed for generalization under the defined experimental conditions without claiming universal zero-day detection capability.

#### 4.13 Implementation Verification

Before the main experiments, the implementation is verified through a sequence of functional checks. Communication between OS-Ken and OVS is verified first. The Mininet web service at 10.0.0.1 is then checked for accessibility. Legitimate traffic from 192.168.57.31 is tested, followed by controlled traffic from the Kali attacker at 192.168.56.102.

The ML component is checked to ensure that live feature input is compatible with the trained model. Finally, the controller's ability to install and enforce the mitigation rule is verified.

```

root@Ubuntu-SDN:/home/safi/SDN-Project/topology# python3 webserver_topology.py

*** Creating SDN network
*** Adding OS-Ken controller
*** Adding hosts
*** Adding switch s1
*** Creating links
*** Starting Mininet
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Configuring s1 gateway: 10.0.0.254
*** Adding route on h1 to Kali network
*** Starting web server on h1
*** Web server started automatically

=====
SDN MININET WEB SERVER
=====
OS-Ken Controller : 127.0.0.1:6633
Switch           : s1
Gateway          : 10.0.0.254
Web Server       : h1
Web Server IP    : 10.0.0.1
HTTP Port        : 80
Web Root         : /home/safi/SDN-Project/webserver
Kali Network     : 192.168.56.0/24
=====

```

```

CONFIDENCE       : 1.0
METHOD           : RF+BASELINE
ANOMALY SCORE    : 0
REASON           : normal behavior
=====

SOURCE IP        : 10.0.0.1
DESTINATION IP   : 192.168.56.103
PROTOCOL         : ICMP
SOURCE PORT      : 0
DESTINATION PORT : 0

PACKETS          : 51
BYTES            : 4998
PACKET RATE      : 0.996 packets/sec
BYTE RATE        : 97.597 bytes/sec
DURATION         : 51.211 seconds
SOURCE FLOW COUNT : 1
SOURCE RATE      : 0.996 packets/sec
DEST FLOW COUNT  : 1
DESTINATION RATE : 0.996 packets/sec

=====
HYBRID DDoS DETECTION
=====
CLASS            : BENIGN
CONFIDENCE       : 1.0
METHOD           : RF+BASELINE
ANOMALY SCORE    : 0
REASON           : normal behavior
=====

```

Figure 12 Normal traffic flow

```

(root@ kali)-[/home/safi]
# hping3 -S -p 80 --flood 10.0.0.1
HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

**Figure 13 Generating Attack traffic from kali Linux**

```

SOURCE FLOW COUNT: 2
SOURCE RATE      : 1.277 packets/sec
DEST FLOW COUNT  : 1
DESTINATION RATE : 1.162 packets/sec

=====
HYBRID DDoS DETECTION
=====
CLASS      : BENIGN
CONFIDENCE  : 0.9933
METHOD      : RF+BASELINE
ANOMALY SCORE : 0
REASON      : normal behavior
=====

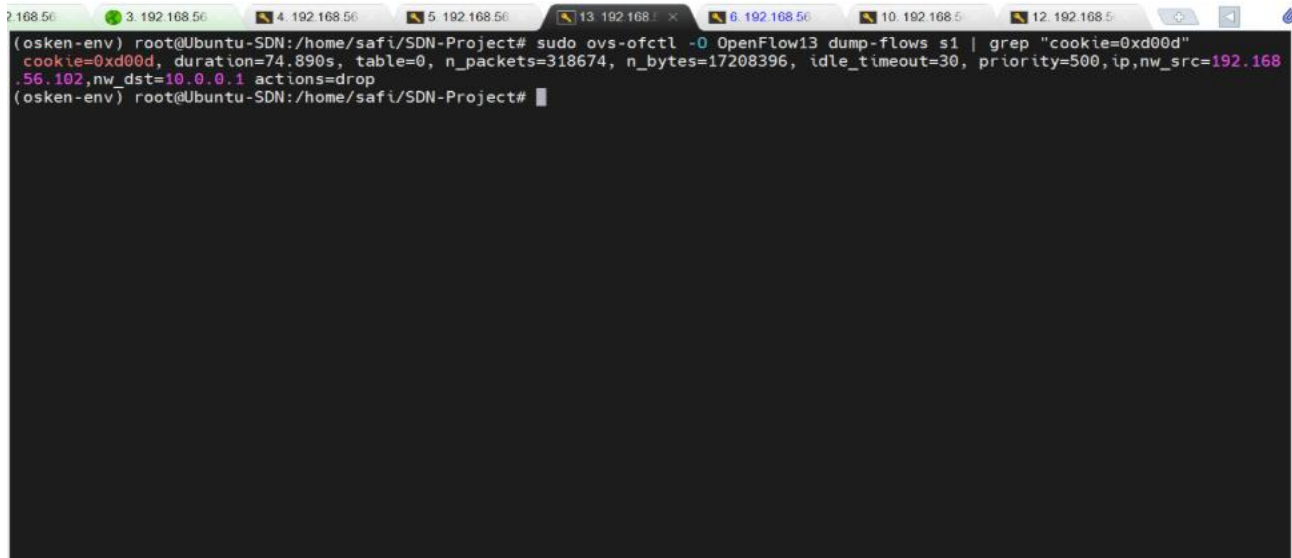
!!! AUTO MITIGATION ACTIVATED !!!

=====
SWITCH DPID      : 1
SOURCE IP        : 192.168.56.102
DESTINATION IP   : 10.0.0.1
ACTION          : DROP
PRIORITY         : 500
COOKIE          : 0xd00d
INACTIVITY TIMEOUT: 30 seconds (controller-managed)
MATCH           : IPv4 source + IPv4 destination
SOURCE PORT      : ANY
DESTINATION PORT : ANY
STATUS          : DROP RULE SENT TO OVS
=====

```

**Figure 14 Attack detected**

This Figure shows attack detected and mitigation rule sent to OVS switch

A terminal window with a dark background and light-colored text. The prompt is '(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project#'. The command entered is 'sudo ovs-ofctl -O OpenFlow13 dump-flows s1 | grep "cookie=0xd00d"'. The output is a single line: 'cookie=0xd00d, duration=74.890s, table=0, n\_packets=318674, n\_bytes=17208396, idle\_timeout=30, priority=500, ip,nw\_src=192.168.56.102,nw\_dst=10.0.0.1 actions=drop'. The prompt is then shown again.

```
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# sudo ovs-ofctl -O OpenFlow13 dump-flows s1 | grep "cookie=0xd00d"
cookie=0xd00d, duration=74.890s, table=0, n_packets=318674, n_bytes=17208396, idle_timeout=30, priority=500, ip,nw_src=192.168.56.102,nw_dst=10.0.0.1 actions=drop
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project#
```

**Figure 15 Drop rule installed**

This figure shows which rules are installed for mitigation.

#### 4.14 Chapter Summary

This chapter presented the design and implementation of the adaptive AI-based DDoS detection and mitigation framework. The system integrates Mininet and OVS for the SDN data plane, OS-Ken for centralized control, InSDN for machine-learning model development, and Random Forest for traffic classification.

The protected web service is implemented inside Mininet at 10.0.0.1, while Kali Linux at 192.168.56.102 generates controlled attack traffic and the normal user at 192.168.56.103 represents legitimate activity. Python and Scikit-learn provide the ML and data-processing environment, while OS-Ken translates malicious detections into OpenFlow-based mitigation actions.

The implemented system provides the operational foundation for the experiments presented in the next chapter. The next chapter evaluates classification performance, previously unseen attack generalization, detection latency, mitigation response time, and network/service performance.



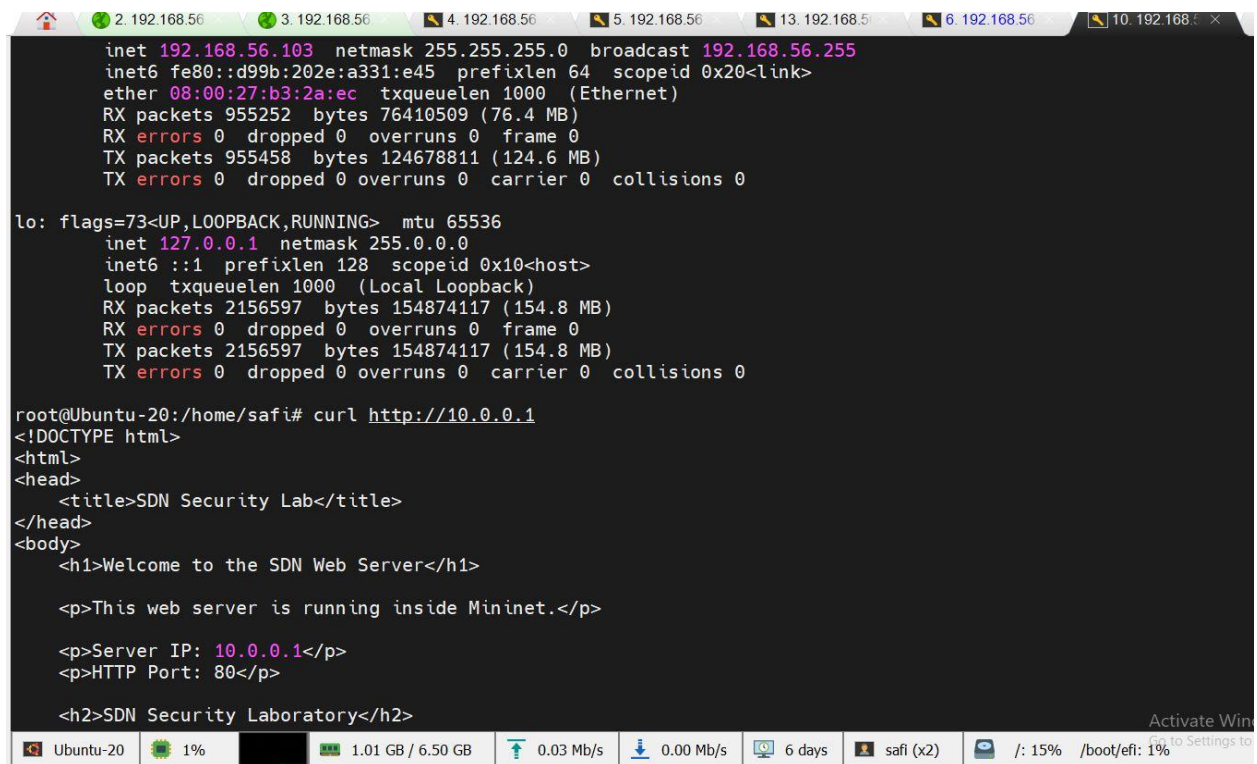
## 5. EVALUATION

### 5.1 Overview

This chapter evaluates the proposed Adaptive AI-Based DDoS Detection and Mitigation Framework implemented in the SDN environment. The evaluation focuses on Random Forest classification performance, detection of previously unseen DDoS behavior, detection latency, mitigation response time, and network and web-service performance. The evaluation considers the complete detection-to-mitigation process rather than classification accuracy alone.

### 5.2 Experimental Scenarios

The framework is evaluated under three conditions: normal traffic, known as DDoS traffic, and previously unseen DDoS behavior. Normal traffic establishes the baseline performance of the protected service. Known DDoS traffic is used to evaluate the Random Forest detection capability, while previously unseen behavior is used to examine the model's ability to generalize beyond the traffic patterns represented during training.



```
inet 192.168.56.103 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::d99b:202e:a331:e45 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:b3:2a:ec txqueuelen 1000 (Ethernet)
RX packets 955252 bytes 76410509 (76.4 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 955458 bytes 124678811 (124.6 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 2156597 bytes 154874117 (154.8 MB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 2156597 bytes 154874117 (154.8 MB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@Ubuntu-20:/home/safi# curl http://10.0.0.1
<!DOCTYPE html>
<html>
<head>
  <title>SDN Security Lab</title>
</head>
<body>
  <h1>Welcome to the SDN Web Server</h1>

  <p>This web server is running inside Mininet.</p>

  <p>Server IP: 10.0.0.1</p>
  <p>HTTP Port: 80</p>

  <h2>SDN Security Laboratory</h2>
```

**Figure 16 Normal traffic and successful web-service access.**

This is evidence of an accessible webserver for normal users.

```

(root@kali)-[/home/safi]
# hping3 -S -p 80 --flood 10.0.0.1
HPING 10.0.0.1 (eth0 10.0.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

```

**Figure 17 Controlled DDoS traffic generation.**

This is the sync flood generation from kali Linux towards web server.

```

SOURCE FLOW COUNT: 2
SOURCE RATE      : 1.277 packets/sec
DEST FLOW COUNT  : 1
DESTINATION RATE : 1.162 packets/sec

=====
HYBRID DDoS DETECTION
=====
CLASS      : BENIGN
CONFIDENCE : 0.9933
METHOD     : RF+BASELINE
ANOMALY SCORE : 0
REASON     : normal behavior
=====

!!! AUTO MITIGATION ACTIVATED !!!

=====
SWITCH DPID      : 1
SOURCE IP        : 192.168.56.102
DESTINATION IP   : 10.0.0.1
ACTION          : DROP
PRIORITY        : 500
COOKIE          : 0xd00d
INACTIVITY TIMEOUT: 30 seconds (controller-managed)
MATCH           : IPv4 source + IPv4 destination
SOURCE PORT     : ANY
DESTINATION PORT: ANY
STATUS          : DROP RULE SENT TO OVS
=====

```

**Figure 18 Random Forest DDoS detection output.**

### 5.3 DDoS Detection and Classification Performance

The Random Forest model is evaluated using accuracy, precision, recall, and F1-score. Recall is particularly important because false negatives represent malicious traffic that is not detected, while precision indicates the extent to which legitimate traffic is correctly distinguished from

malicious traffic. The confusion matrix is also used to provide evidence of classification outcomes.

```
Fold 5: Accuracy=91.45% | Precision=90.87% | Recall=90.93% | F1=90.90%
```

```
=====
FEATURE IMPORTANCE CHECK
=====
```

|                             |          |          |
|-----------------------------|----------|----------|
| 1. source_port              | 0.227031 | (22.70%) |
| 2. destination_flow_count   | 0.114811 | (11.48%) |
| 3. destination_packet_count | 0.113075 | (11.31%) |
| 4. source_packet_count      | 0.089633 | (8.96%)  |
| 5. source_flow_count        | 0.068037 | (6.80%)  |
| 6. source_packet_rate       | 0.055936 | (5.59%)  |
| 7. destination_packet_rate  | 0.053926 | (5.39%)  |
| 8. bytes_per_packet         | 0.053528 | (5.35%)  |
| 9. avg_packet_size          | 0.050345 | (5.03%)  |
| 10. destination_port        | 0.027567 | (2.76%)  |
| 11. duration                | 0.020327 | (2.03%)  |
| 12. packets_per_second      | 0.019198 | (1.92%)  |
| 13. byte_rate               | 0.018882 | (1.89%)  |
| 14. packets                 | 0.017958 | (1.80%)  |
| 15. bytes                   | 0.016014 | (1.60%)  |
| 16. packet_rate             | 0.015239 | (1.52%)  |
| 17. bytes_per_second        | 0.014614 | (1.46%)  |
| 18. protocol                | 0.010153 | (1.02%)  |
| 19. source_byte_count       | 0.006893 | (0.69%)  |
| 20. destination_byte_count  | 0.006834 | (0.68%)  |

```
-----
DOMINANT FEATURE CHECK
-----
```

```
Top feature : source_port
Importance  : 22.70%
```

**Figure 19 Random Forest classification performance for known DDoS traffic.**

This figure shows that which features are extracted from the packet are the baseline of the research dataset.

```
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# cd /home/safi/SDN-Project && python3 -c "import joblib,pandas as pd; from sklearn.model_selection import train_test_split; from sklearn.metrics import confusion_matrix,classification_report; m=joblib.load('model/ddos_random_forest.pkl'); df=pd.read_csv('dataset/ml_features.csv'); X=df['features']; y=df['label']; X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42,stratify=y); y_pred=m['model'].predict(X_test); print('\nClasses:',m['model'].classes_); print('\nConfusion Matrix:'); print(confusion_matrix(y_test,y_pred,labels=m['model'].classes_)); print('\nClassification Report:'); print(classification_report(y_test,y_pred))"
/home/safi/SDN-Project/osken-env/lib/python3.10/site-packages/sklearn/utils/validation.py:2742: UserWarning: X has feature names, but RandomForestClassifier was fitted without feature names
  warnings.warn(

Classes: ['BENIGN' 'DDoS']

Confusion Matrix:
[[2483 308]
 [ 288 4337]]

Classification Report:

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| BENIGN       | 0.90      | 0.89   | 0.89     | 2791    |
| DDoS         | 0.93      | 0.94   | 0.94     | 4625    |
| accuracy     |           |        | 0.92     | 7416    |
| macro avg    | 0.91      | 0.91   | 0.91     | 7416    |
| weighted avg | 0.92      | 0.92   | 0.92     | 7416    |

```
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project#
```

**Figure 20 Random Forest confusion matrix.**

Determine the traffic flow status, based on this parameter of traffic classified as normal or attack.

```
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# python3 validate_model.py

=====
MODEL VALIDATION
=====

Records: 37079
Features: 20
BENIGN: 13954
DDoS: 23125

Existing Random Forest loaded successfully.

=====
5-FOLD CROSS VALIDATION
=====

-----
CROSS-VALIDATION PERFORMANCE
-----
Accuracy : 91.78% +/- 0.23%
Precision : 91.24% +/- 0.25%
Recall : 91.27% +/- 0.27%
F1-Score : 91.25% +/- 0.24%

-----
FOLD RESULTS
-----
Fold 1: Accuracy=91.86% | Precision=91.20% | Recall=91.53% | F1=91.36%
Fold 2: Accuracy=92.08% | Precision=91.57% | Recall=91.57% | F1=91.57%
Fold 3: Accuracy=91.92% | Precision=91.45% | Recall=91.32% | F1=91.38%
Fold 4: Accuracy=91.60% | Precision=91.09% | Recall=90.99% | F1=91.04%
```

**Figure 21 Accuracy, precision, recall and F1 score.**

Random forest validates successfully and the parameters accuracy defined.

## 5.4 Previously Unseen DDoS Behavior

Previously unseen DDoS behavior was introduced during evaluation without being directly represented in the model training data. This experiment examines whether the proposed detection mechanism can identify traffic that differs from previously observed attack patterns. The results are compared with known attack performance to assess generalization under the defined experimental conditions.

```
64 bytes from 10.0.0.1: icmp_seq=87 ttl=64 time=0.429 ms
64 bytes from 10.0.0.1: icmp_seq=88 ttl=64 time=0.097 ms
64 bytes from 10.0.0.1: icmp_seq=89 ttl=64 time=0.165 ms
64 bytes from 10.0.0.1: icmp_seq=90 ttl=64 time=0.094 ms
64 bytes from 10.0.0.1: icmp_seq=91 ttl=64 time=0.805 ms
64 bytes from 10.0.0.1: icmp_seq=92 ttl=64 time=0.105 ms
64 bytes from 10.0.0.1: icmp_seq=93 ttl=64 time=0.084 ms
64 bytes from 10.0.0.1: icmp_seq=94 ttl=64 time=0.099 ms
64 bytes from 10.0.0.1: icmp_seq=95 ttl=64 time=0.831 ms
64 bytes from 10.0.0.1: icmp_seq=96 ttl=64 time=0.117 ms
64 bytes from 10.0.0.1: icmp_seq=97 ttl=64 time=0.168 ms
64 bytes from 10.0.0.1: icmp_seq=98 ttl=64 time=0.109 ms
64 bytes from 10.0.0.1: icmp_seq=99 ttl=64 time=0.451 ms
64 bytes from 10.0.0.1: icmp_seq=100 ttl=64 time=0.097 ms

--- 10.0.0.1 ping statistics ---
100 packets transmitted, 100 received, 0% packet loss, time 101834ms
rtt min/avg/max/mdev = 0.072/0.573/28.210/2.801 ms
mininet> mininet> h3 sh -c 'for i in $(seq 1 500); do ping -c 1 -W 1 10.0.0.1 >/dev/null; done'
*** Unknown command: mininet> h3 sh -c 'for i in $(seq 1 500); do ping -c 1 -W 1 10.0.0.1 >/dev/null; done'
mininet> h3 sh -c 'for i in $(seq 1 500); do ping -c 1 -W 1 10.0.0.1 >/dev/null; done'
mininet> tail -20 /home/safi/SDN-Project/dataset/live_flows.csv
*** Unknown command: tail -20 /home/safi/SDN-Project/dataset/live_flows.csv
mininet> mininet>sh -c 'for i in $(seq 1 2000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
*** Unknown command: mininet>sh -c 'for i in $(seq 1 2000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
mininet> sh -c 'for i in $(seq 1 2000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
/bin/sh: 0: Illegal option -
mininet> h2 sh -c 'for i in $(seq 1 2000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
mininet> h2 sh -c 'for i in $(seq 1 5000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
mininet> h2 sh -c 'for i in $(seq 1 5000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
mininet> h2 sh -c 'for i in $(seq 1 5000); do ping -c 1 -W 1 10.0.0.1 >/dev/null & done; wait'
mininet>
```

Figure 22 Generating Unseen Zero-day attack

Generating unseen attacks from h2 host to webserver which is not available in research dataset. So this will be considered an unseen attack.

```

DESTINATION IP : 10.0.0.2
PROTOCOL       : ICMP
SOURCE PORT    : 0
DESTINATION PORT : 0

PACKETS       : 1340
BYTES         : 131320
PACKET RATE   : 111.372 packets/sec
BYTE RATE     : 10914.477 bytes/sec
DURATION      : 12.032 seconds
SOURCE FLOW COUNT: 1
SOURCE RATE   : 111.372 packets/sec
DEST FLOW COUNT : 1
DESTINATION RATE : 111.372 packets/sec

=====
HYBRID DDoS DETECTION
=====
CLASS       : DDoS
CONFIDENCE  : 0.99
METHOD      : BEHAVIOR
ANOMALY SCORE : 13
REASON      : packet_rate above INSDN baseline; byte_rate above INSDN baseline; ICMP flood rate; high packet-rate burst; high source packet rate; above adaptive live baseline; source rate above adaptive baseline; destination rate above adaptive baseline

!!! DDoS / ABNORMAL TRAFFIC DETECTED !!!
!!! AUTO MITIGATION ACTIVE !!!

=====
[DROP COUNTER ACTIVE] 10.0.0.2 -> 10.0.0.1 | packets: 3069 | bytes: 300762
[MITIGATION ACTIVE] 10.0.0.2 -> 10.0.0.1 | DROP packets: 3069 | DROP bytes: 300762 | RULE KEPT

```

**Figure 24 Detection output for previously unseen DDoS behaviour.**

Controllers detect an unseen attack and define reason as well, because dataset does not have this behavioral traffic.

## 5.5 Detection Latency

Detection latency measures the time between the observation of attack traffic and the generation of the malicious classification decision. This metric is important for determining how quickly the AI-based detection mechanism can identify an attack in the live SDN environment.

```

64 bytes from 10.0.0.1: icmp_seq=71 ttl=63 time=5.33 ms
64 bytes from 10.0.0.1: icmp_seq=72 ttl=63 time=3.13 ms
64 bytes from 10.0.0.1: icmp_seq=73 ttl=63 time=2.67 ms
64 bytes from 10.0.0.1: icmp_seq=74 ttl=63 time=4.14 ms
64 bytes from 10.0.0.1: icmp_seq=75 ttl=63 time=2.18 ms
64 bytes from 10.0.0.1: icmp_seq=76 ttl=63 time=384 ms
64 bytes from 10.0.0.1: icmp_seq=82 ttl=63 time=2257 ms
64 bytes from 10.0.0.1: icmp_seq=89 ttl=63 time=2214 ms
64 bytes from 10.0.0.1: icmp_seq=92 ttl=63 time=2486 ms
64 bytes from 10.0.0.1: icmp_seq=94 ttl=63 time=2232 ms
64 bytes from 10.0.0.1: icmp_seq=102 ttl=63 time=2091 ms
64 bytes from 10.0.0.1: icmp_seq=104 ttl=63 time=2007 ms
64 bytes from 10.0.0.1: icmp_seq=129 ttl=63 time=103 ms
64 bytes from 10.0.0.1: icmp_seq=130 ttl=63 time=397 ms
64 bytes from 10.0.0.1: icmp_seq=131 ttl=63 time=3.78 ms
64 bytes from 10.0.0.1: icmp_seq=132 ttl=63 time=7.30 ms
64 bytes from 10.0.0.1: icmp_seq=133 ttl=63 time=3.05 ms
64 bytes from 10.0.0.1: icmp_seq=134 ttl=63 time=4.42 ms
64 bytes from 10.0.0.1: icmp_seq=135 ttl=63 time=3.99 ms
64 bytes from 10.0.0.1: icmp_seq=136 ttl=63 time=3.39 ms
64 bytes from 10.0.0.1: icmp_seq=137 ttl=63 time=2.94 ms

```

**Figure 25 Detection latency across experimental runs.**

Here it has been seen that latency varies during normal traffic latency is 2.18ms while attack on going latency increased to 2091ms and after mitigation rules installation latency back into normal in 2.94ms.

## 5.6 Mitigation Response Time

Mitigation response time measures the interval between the DDoS detection decision and enforcement of the corresponding mitigation rule through the SDN controller and OpenFlow. The OVS flow table is examined before and after mitigation to confirm that the malicious flow rule has been installed.

```

SOURCE FLOW COUNT: 2
SOURCE RATE      : 1.277 packets/sec
DEST FLOW COUNT  : 1
DESTINATION RATE : 1.162 packets/sec

=====
HYBRID DDoS DETECTION
=====
CLASS      : BENIGN
CONFIDENCE : 0.9933
METHOD     : RF+BASELINE
ANOMALY SCORE : 0
REASON     : normal behavior
=====

!!! AUTO MITIGATION ACTIVATED !!!

=====
SWITCH DPID      : 1
SOURCE IP        : 192.168.56.102
DESTINATION IP   : 10.0.0.1
ACTION          : DROP
PRIORITY         : 500
COOKIE          : 0xd00d
INACTIVITY TIMEOUT: 30 seconds (controller-managed)
MATCH           : IPv4 source + IPv4 destination
SOURCE PORT     : ANY
DESTINATION PORT: ANY
STATUS          : DROP RULE SENT TO OVS
=====

```

**Figure 26 OS-Ken detection and mitigation log.**

```

(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# sudo ovs-ofctl -O OpenFlow13 dump-flows s1 | grep "cookie=0xd00d"
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project#

```

**Figure 27 OVS flow table before mitigation.**

Before implementation of mitigation OVS flow table is empty

```

(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# sudo ovs-ofctl -O OpenFlow13 dump-flows s1 | grep "cookie=0xd00d"
cookie=0xd00d, duration=16.371s, table=0, n_packets=42207, n_bytes=2279178, idle_timeout=30, priority=500,ip,nw_src=192.168.5
6.102,nw_dst=10.0.0.1 actions=drop
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project# sudo ovs-ofctl -O OpenFlow13 dump-flows s1 | grep "cookie=0xd00d"
cookie=0xd00d, duration=18.568s, table=0, n_packets=45371, n_bytes=2450034, idle_timeout=30, priority=500,ip,nw_src=192.168.5
6.102,nw_dst=10.0.0.1 actions=drop
(osken-env) root@Ubuntu-SDN:/home/safi/SDN-Project#

```

**Figure 28 OVS flow table after mitigation.**

After implementation of mitigation, OVS flow shows the rules.

## 5.7 Network Performance

Network performance is evaluated under normal traffic, DDoS traffic before mitigation, and post-mitigation conditions. Throughput, latency, and packet loss are compared to determining the impact of the attack and whether network conditions improve following mitigation. The protected web service is also evaluated using response time, successful requests, and service availability.

## ork Performance: Normal → DDoS Attack → Service Rec

### 1. Normal communication — low RTT

```
64 bytes from 10.0.0.1: icmp_seq=62 ttl=63 time=2.72 ms
64 bytes from 10.0.0.1: icmp_seq=63 ttl=63 time=2.80 ms
64 bytes from 10.0.0.1: icmp_seq=64 ttl=63 time=3.26 ms
64 bytes from 10.0.0.1: icmp_seq=65 ttl=63 time=3.32 ms
64 bytes from 10.0.0.1: icmp_seq=66 ttl=63 time=3.94 ms
64 bytes from 10.0.0.1: icmp_seq=67 ttl=63 time=2.89 ms
64 bytes from 10.0.0.1: icmp_seq=68 ttl=63 time=2.63 ms
64 bytes from 10.0.0.1: icmp_seq=69 ttl=63 time=2.18 ms
64 bytes from 10.0.0.1: icmp_seq=70 ttl=63 time=2.36 ms
64 bytes from 10.0.0.1: icmp_seq=71 ttl=63 time=5.33 ms
64 bytes from 10.0.0.1: icmp_seq=72 ttl=63 time=3.13 ms
64 bytes from 10.0.0.1: icmp_seq=73 ttl=63 time=2.67 ms
64 bytes from 10.0.0.1: icmp_seq=74 ttl=63 time=4.14 ms
64 bytes from 10.0.0.1: icmp_seq=75 ttl=63 time=2.18 ms
64 bytes from 10.0.0.1: icmp_seq=76 ttl=63 time=384 ms
64 bytes from 10.0.0.1: icmp_seq=82 ttl=63 time=2257 ms
```



### 2. During DDoS traffic — severe latency increase

```
64 bytes from 10.0.0.1: icmp_seq=71 ttl=63 time=5.33 ms
64 bytes from 10.0.0.1: icmp_seq=72 ttl=63 time=3.13 ms
64 bytes from 10.0.0.1: icmp_seq=73 ttl=63 time=2.67 ms
64 bytes from 10.0.0.1: icmp_seq=74 ttl=63 time=4.14 ms
64 bytes from 10.0.0.1: icmp_seq=75 ttl=63 time=2.18 ms
64 bytes from 10.0.0.1: icmp_seq=76 ttl=63 time=384 ms
64 bytes from 10.0.0.1: icmp_seq=82 ttl=63 time=2257 ms
64 bytes from 10.0.0.1: icmp_seq=89 ttl=63 time=2214 ms
64 bytes from 10.0.0.1: icmp_seq=92 ttl=63 time=2486 ms
64 bytes from 10.0.0.1: icmp_seq=94 ttl=63 time=2232 ms
64 bytes from 10.0.0.1: icmp_seq=102 ttl=63 time=2091 ms
64 bytes from 10.0.0.1: icmp_seq=104 ttl=63 time=2007 ms
64 bytes from 10.0.0.1: icmp_seq=129 ttl=63 time=103 ms
64 bytes from 10.0.0.1: icmp_seq=130 ttl=63 time=397 ms
64 bytes from 10.0.0.1: icmp_seq=131 ttl=63 time=3.78 ms
64 bytes from 10.0.0.1: icmp_seq=132 ttl=63 time=7.30 ms
64 bytes from 10.0.0.1: icmp_seq=133 ttl=63 time=3.05 ms
64 bytes from 10.0.0.1: icmp_seq=134 ttl=63 time=4.42 ms
64 bytes from 10.0.0.1: icmp_seq=135 ttl=63 time=3.99 ms
64 bytes from 10.0.0.1: icmp_seq=136 ttl=63 time=3.39 ms
64 bytes from 10.0.0.1: icmp_seq=137 ttl=63 time=2.94 ms
```



### 3. Post-mitigation — connectivity and web service restored

```
root@Ubuntu-20:/home/safi# curl http://10.0.0.1
<!DOCTYPE html>
<html>
<head>
<title>SDN Security Lab</title>
</head>
<body>
<h1>Welcome to the SDN Web Server</h1>
<p>This web server is running inside Mininet.</p>
<p>Server IP: 10.0.0.1</p>
<p>HTTP Port: 80</p>
<h2>SDN Security Laboratory</h2>

```

Figure 29 Network performance under normal, attack and post-mitigation conditions.

Service restoration after mitigation is demonstrated by the return of network latency to normal levels.

## 5.8 End-to-End Evaluation

The complete detection-to-mitigation pipeline is evaluated from attack initiation through traffic observation, feature extraction, Random Forest classification, detection decision, OS-Ken response and OpenFlow rule installation to malicious traffic control and service recovery. This provides evidence that the proposed framework operates as an integrated real-time SDN defense system rather than an offline classifier alone.

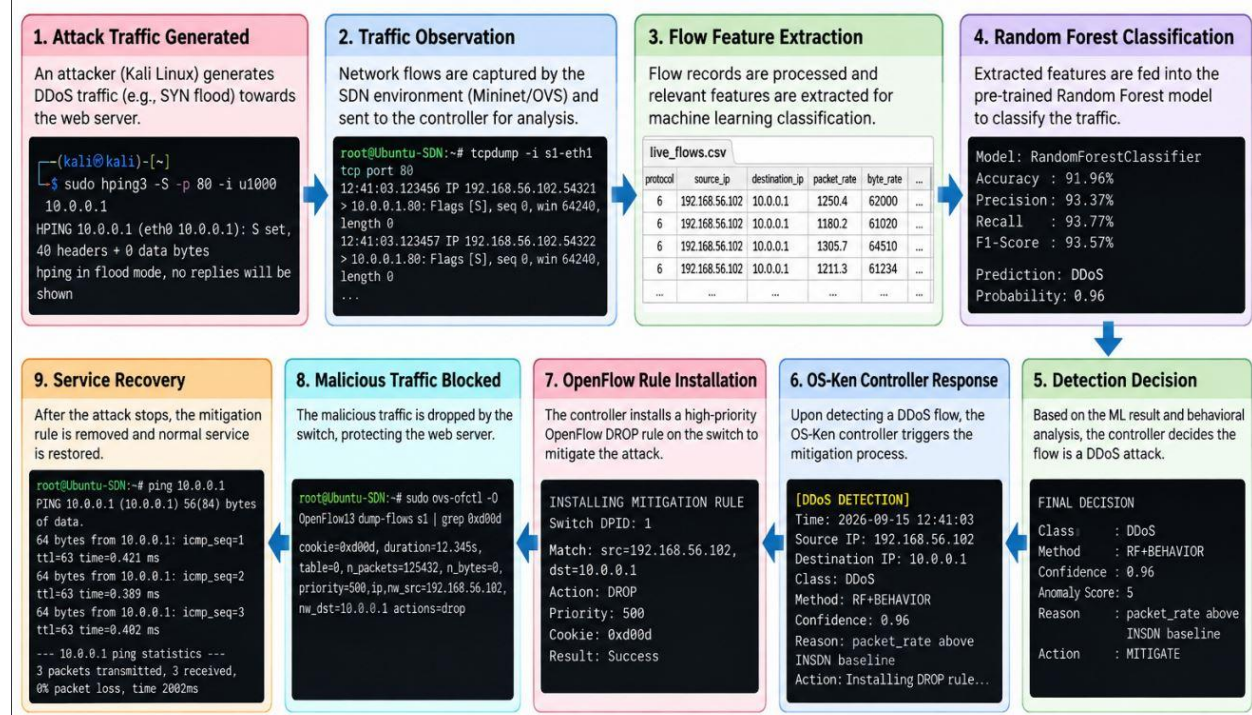


Figure 30 End-to-end detection-to-mitigation process and experimental evidence.

## 5.9 Evaluation Summary

The evaluation assesses the research objectives through classification performance, previously unseen DDoS behavior, detection latency, mitigation response time, network performance, and web-service performance. The results should be mapped to each research objective using the actual experimental evidence.

## 5.10 Comparison with existing research

| Approach                                              | ML Technique                              | SDN Environment                       | Dataset / Features             | Detection Performance                                       | Unseen / Novel Behaviour                | Real-Time Detection | Automatic Mitigation                 | Network Performance Evaluation           |
|-------------------------------------------------------|-------------------------------------------|---------------------------------------|--------------------------------|-------------------------------------------------------------|-----------------------------------------|---------------------|--------------------------------------|------------------------------------------|
| Automated DDoS Detection in SDN [19]                  | SVC + RF                                  | Mininet / SDN                         | SDN traffic features           | 98.8% Accuracy                                              | Limited                                 | yes                 | yes                                  | Limited                                  |
| Feature Engineering-based SDN DDoS Detection [16]     | RF, XGBoost, SVM, KNN, DT                 | SDN                                   | Engineered traffic features    | RF: 99.13% Accuracy, F1 99.13%                              | Limited                                 | yes                 | Not primary focus                    | Limited                                  |
| SDN DDoS ML Evaluation (2024) [21]                    | SVM, DT, KNN                              | Mininet                               | Legitimate + malicious traffic | DT: 99.28%, SVM: 97.31%, KNN: 96.26%                        | Limited                                 | yes                 | Not the main contribution            | Limited                                  |
| Hybrid RF-XGB SDN Framework (2026) [27]               | RF + XGBoost                              | Mininet / Ryu / OVS                   | SDN-specific OpenFlow features | 99.36% Accuracy                                             | Generalisation evaluated                | yes                 | Prevention included                  | Limited                                  |
| DDoS Detection & Mitigation in SDN-SD-WAN (2026) [26] | Multiple ML models                        | SDN-enabled environment               | Flow/network features          | RF evaluated                                                | Not primary focus                       | yes                 | yes                                  | CPU/memory/inference evaluated           |
| This Proposed Research                                | Random Forest + Behaviour-based detection | Mininet + OVS + OS-Ken + OpenFlow 1.3 | 20 flow + behavioural features | 91.96% Accuracy; 93.37% Precision; 93.77% Recall; 93.57% F1 | Previously unseen DDoS behaviour tested | yes                 | yes<br>Automatic OpenFlow DROP rules | RTT + web-service performance + recovery |

**Table 5 Comparison between existing research**

### 5.11 Limitations

The experiments are conducted in a controlled Mininet environment; therefore, the measured performance may differ from production-scale networks. The previously unseen attack experiment represents the selected unseen behavior and should not be interpreted as proof of universal zero-day detection. Results are also dependent on the selected feature space, topology, and available computing resources.

## 6. CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK

### 6.1 Conclusion

This research investigated the application of Software-Defined Networking (SDN) and machine learning (ML) for the detection and mitigation of Distributed Denial-of-Service (DDoS) attacks. The study addressed limitations associated with conventional network security approaches, particularly their dependence on predefined signatures and their limited ability to respond dynamically to emerging attack behavior. An SDN-based security architecture was therefore developed in which network monitoring, ML-based traffic classification, behavioral analysis, and automated OpenFlow mitigation were integrated through an OS-Ken controller.

The implementation was developed and evaluated using a Mininet and Open vSwitch testbed. The INSDN dataset was used as the reference dataset for ML development and behavioral analysis. A Random Forest classifier was trained using network-flow characteristics to identify known malicious and benign traffic. The trained model achieved an accuracy of 91.96%, with precision of 93.37%, recall of 93.77%, and an F1-score of 93.57%. These results demonstrate that flow-level network features can provide an effective basis for identifying DDoS traffic within the experimental environment.

To extend detection beyond previously observed attack patterns, the proposed architecture incorporated a behavioral anomaly-detection mechanism based on benign traffic characteristics and Isolation Forest. This provided an additional detection layer for identifying traffic that deviated significantly from established normal behavior. The combination of supervised classification and behavioral analysis was intended to reduce dependence on a single detection mechanism and improve the system's ability to address both recognized and previously unseen traffic patterns.

A further contribution to the research was the integration of detection with automated mitigation. Following a sufficiently confident malicious classification, the OS-Ken controller generated OpenFlow rules that instructed the OVS switch to drop the identified malicious traffic. The controller also monitored mitigation-rule activity and managed rule removal according to the implemented release conditions. Consequently, the system was not limited to identifying attacks but provided an end-to-end security process from traffic observation and classification to network-level enforcement.

The experimental evaluation further demonstrated the importance of measuring network behavior across normal, attack, and post-mitigation conditions. The results showed how DDoS traffic affected network performance and how controller-driven mitigation could restore network operation after malicious traffic was identified. Overall, the research demonstrates the feasibility of combining SDN programmability with ML-based detection to create a more adaptive DDoS defense mechanism. However, the findings should be interpreted within the limitations of the

experimental testbed, dataset, traffic-generation methods, and computational environment used in this study.

## 6.2 Recommendations for Future Work

Several areas could be investigated to extend this research. First, future studies should evaluate the proposed architecture using larger and more diverse datasets containing a broader range of contemporary DDoS and legitimate traffic patterns. Testing real-world traffic or additional benchmark datasets would provide stronger evidence of generalization beyond the INSDN dataset.

Second, alternative ML and deep-learning techniques could be investigated and compared with the Random Forest approach. Models such as XGBoost, Support Vector Machines, neural networks, or recurrent architectures could be evaluated in terms of detection accuracy, false-positive rate, computational cost, and inference latency. Ensemble approaches could also be explored to determine whether combining multiple classifiers improves robustness.

Third, the zero-day detection mechanism could be enhanced through adaptive and continuously updated behavioral baselines. Such a mechanism could allow the controller to learn to change legitimate traffic patterns while reducing the risk of incorrectly classifying temporary traffic bursts as attacks. Further research should also investigate techniques for reducing false positives without compromising detection sensitivity.

Fourth, future implementations should evaluate scalability using larger Mininet or physical SDN environments containing multiple switches and controllers. Controller processing overhead, flow-table capacity, detection latency, and mitigation latency should be measured under increasing traffic volumes and concurrent attacks.

Finally, future research could investigate more granular mitigation strategies rather than relying primarily on DROP rules. Rate limiting, traffic redirection, dynamic flow prioritization, and coordinated multi-switch mitigation could provide additional flexibility. Integration with distributed or multi-controller SDN architectures could also improve resilience against controller bottlenecks and single points of failure. These developments would help move the proposed approach from a laboratory proof of concept towards evaluation in larger and more realistic operational network environments.

## 7. REFLECTION

Completing this project significantly improved my technical knowledge, practical skills, and understanding of cybersecurity. One of the most important things I learned was how Software-Defined Networking (SDN) can be used to support dynamic network security. Through practical work with Mininet, Open vSwitch, OpenFlow, and the OS-Ken controller, I developed a better understanding of how a centralized controller can monitor traffic and dynamically apply flow rules to respond to malicious activity.

I also gained practical experience in machine learning for cybersecurity. Working with the INSDN dataset taught me how network-flow data can be prepared, relevant features selected, and a Random Forest model trained and evaluated. I learned that accuracy alone is not sufficient when assessing a security model, and that precision, recall, F1-score, false positives, and detection behavior are also important. Developing the behavioral and Isolation Forest component further helped me understand the challenges of identifying previously unseen or abnormal network behavior.

A major learning experience came from troubleshooting practical implementation. I encountered issues involving controller compatibility, network interfaces, connectivity, OpenFlow rules, and traffic forwarding. Resolving these problems required systematic testing and the use of tools such as tcpdump, Wireshark, ovs-ofctl, and Linux networking commands. This improved my problem-solving skills and taught me to rely on evidence from logs and experiments rather than assumptions.

The project also improved my research and analytical skills. Evaluating the network under normal, attack, and post-mitigation conditions showed me the importance of measuring both security effectiveness and network performance. Overall, the project strengthened my knowledge of SDN, machine learning, DDoS detection, and automated mitigation while improving my troubleshooting, experimentation, critical analysis, and technical documentation skills. These skills will be valuable for my future development in network security and cybersecurity.



## 8. REFERENCES

- [1] S. T. Zargar, J. B. Joshi, and D. Tipper, “A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013, doi: 10.1109/SURV.2013.031413.00127.
- [2] D. Kreutz, F. M. V. Ramos, P. 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, Jan. 2015
- [3] H. Riggs, A. Khalid, and A. I. Sarwat, “An overview of SDN issues—A case study and performance evaluation of a secure OpenFlow protocol implementation,” *Electronics*, vol. 14, no. 16, Art. no. 3244, Aug. 2025,
- [4] W. Hill, Y. T. Acquaah, J. Mason, D. Limbrick, S. Teixeira-Poit, C. Coates, and K. Roy, “DDoS in SDN: A review of open datasets, attack vectors and mitigation strategies,” *Discover Applied Sciences*, vol. 6, Art. no. 472, 2024
- [5] O. Rahman, M. A. G. Quraishi, and C.-H. Lung, “DDoS attacks detection and mitigation in SDN using machine learning,” in *Proc. IEEE World Congress on Services (SERVICES)*, Milan, Italy, 2019, pp. 184–189,
- [6] T. E. Ali, Y.-W. Chong, and S. Manickam, “Machine learning techniques to detect a DDoS attack in SDN: A systematic review,” *Applied Sciences*, vol. 13, no. 5, Art. no. 3183, 2023,
- [7] V. Hnamte, A. A. Najar, H. Nhung-Nguyen, J. Hussain, and S. M. Naik Sugali, “DDoS attack detection and mitigation using deep neural network in SDN environment,” *Computers & Security*, vol. 138, Art. no. 103661, 2024
- [8] M. A. Setitra, M. Fan, I. Benkhaddra, and Z. E. A. Bensalem, “DoS/DDoS attacks in Software Defined Networks: Current situation, challenges and future directions,” *Computer Communications*, vol. 222, pp. 77–96, 2024,
- [9] A. A. Wabi, I. Idris, O. M. Olaniyi, and J. A. Ojeniyi, “DDOS attack detection in SDN: Method of attacks, detection techniques, challenges and research gaps,” *Computers & Security*, vol. 139, Art. no. 103652, 2024,
- [10] M. N. Swileh and S. Zhang, “Unseen attack detection in software-defined networking using a BERT-based large language model,” *AI*, vol. 6, no. 7, Art. no. 154, 2025,
- [11] B. Lantz, B. Heller, and N. McKeown, “A network in a laptop: Rapid prototyping for software-defined networks,” in *Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-IX)*, 2010, pp. 1–6,

- [12] A. A. Alashhab, M. S. M. Zahid, M. A. Azim, M. Y. Doha, B. Isyaku, and S. Ali, "A survey of low rate DDoS detection techniques based on machine learning in software-defined networks," *Symmetry*, vol. 14, no. 8, Art. no. 1563, 2022,
- [13] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016,
- [14] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. El Moussa, "DeepIDS: Deep learning approach for intrusion detection in software defined networking," *Electronics*, vol. 9, no. 9, Art. no. 1533, 2020,
- [15] A. A. Najar and S. M. Naik, "Cyber-Secure SDN: A CNN-based approach for efficient detection and mitigation of DDoS attacks," *Computers & Security*, vol. 139, Art. no. 103716, 2024,
- [16] D. Han, H. Li, X. Fu, and S. Zhou, "Traffic feature selection and distributed denial of service attack detection in software-defined networks based on machine learning," *Sensors*, vol. 24, no. 13, Art. no. 4344, 2024,
- [17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116,
- [18] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A novel SDN intrusion dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020,
- [19] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDoS attack detection in software defined networking," *Journal of Network and Computer Applications*, vol. 187, Art. no. 103108, 2021,
- [20] U. H. Garba, A. N. Toosi, M. F. Pasha, and S. Khan, "SDN-based detection and mitigation of DDoS attacks on smart homes," *Computer Communications*, vol. 221, pp. 29–41, 2024,
- [21] A. Hirsi, L. Audah, A. Salh, N. M. Sahar, S. Ahmed, and M. A. Alhartomi, "DDoS anomaly detection in software-defined networks: An evaluation of machine learning techniques for traffic classification and prediction," in *Proc. 2024 Int. Conf. on Future Technologies for Smart Society (ICFTSS)*, 2024,
- [22] J. Pebrianto and V. Suryani, "Adaptive DDoS attack detection: Entropy-based model with dynamic threshold and suspicious IP reevaluation," *IEEE Access*, vol. 13, pp. 55858–55876, 2025,

- [23] A. I. Hassan, E. Abd El Reheem, and S. K. Guirguis, “An entropy and machine learning based approach for DDoS attacks detection in software defined networks,” *Scientific Reports*, vol. 14, Art. no. 18159, 2024,
- [24] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, pp. 5–32, 2001,
- [25] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [26] M. Musa, T. F. Ang, Y.-L. Chen, U. A. Bhatti, C. S. Ku, Y. Luo, J. Chen, and L. Y. Por, “Distributed denial of service detection and mitigation in software-defined networking-enabled software-defined wide area networks,” *PLOS ONE*, vol. 21, no. 5, Art. no. e0346673, 2026,
- [27] I. A. Mahar, K. Aziz, P. Chakrabarti, N. Ahmed, M. Ladan, and Y. Javed, “A hybrid machine learning approach for detecting DDoS attacks in software-defined networks,” *Scientific Reports*, vol. 16, Art. no. 6533, 2026,