ORIGINAL ARTICLE UHD JOURNAL OF SCIENCE AND TECHNOLOGY

An Ensemble-based Machine Learning Framework for Advanced Distributed Denial of Service Attack Detection in Software Defined Networks

Aram Saleem[1], Hakem Beitollahi[2]

[1]MSc Student, Department of Computer Science, Soran University, Soran, Iraq, [2]Assistant Professor, Department of Computer Science, Soran University, Soran, Iraq

ABSTRACT

Distributed Denial of Service (DDoS) attacks pose a significant threat to modern network architectures, especially Software Defined Networking (SDN) due to its centralized controller. This study proposes an advanced framework for DDoS attack identification and prediction using state-of-the-art machine learning (ML) techniques in an SDN architecture. A comprehensive dataset was generated through a two-stage traffic generation procedure, simulating attack and normal scenarios over a 6-day period, from which fifteen were extracted to characterize network behavior. Multiple classifiers including Gradient Boosting Ensemble methods such as LightGBM, XGBoost, CatBoost, and Gradient Boosting Decision Trees, as well as additional ensemble methods such as AdaBoost and Bagging were evaluated alongside with One-Class SVM and Bayesian Networks. They were trained and evaluated using rigorous cross-validation. The results demonstrate near-perfect performance of ensemble models, achieving up to 99.98% accuracy with outstanding precision, recall, and area under curve metrics. To achieve efficient mitigation, the detection mechanism is deployed on local web servers, and a certificate authority-based secure communication channel transmits malicious IPs to the SDN controller, enabling low-latency, scalable, and real-time DDoS attack mitigation. This paper discusses the promise of applying cutting-edge ML models to enhance the robustness of SDN infrastructures against sophisticated cyber-attacks and offers a template for further research in dynamic network defense strategies.

Index Terms: Software Defined Networking, Distributed Denial of Service, Machine Learning, Ensemble Models, Traffic Classification

1. INTRODUCTION

Distributed denial of service (DDoS) attacks have evolved into a critical threat to the stability and security of modern network infrastructures [1], [2]. These attacks aim to render services unavailable to legitimate users by overwhelming network resources with malicious traffic. The impact of DDoS attacks extends beyond service disruption, causing significant financial losses, reputational damage, and disruptions across various sectors, including finance, healthcare, and cloud computing. Consequently, network administrators face increasing challenges in mitigating the sophistication and multi-layered nature of these attacks. Software Defined Networking (SDN) has revolutionized network management by decoupling the control plane from the data plane, enabling centralized control, programmability, and dynamic resource allocation [3], [4]. This architectural shift offers significant benefits, such as enhanced scalability,

Access this article online

DOI:10.21928/uhdjst.v9n2y2025.pp184-197

E-ISSN: 2521-4217
P-ISSN: 2521-4209

Corresponding author's e-mail: Hakem Beitollahi, Assistant Professor, Department of Computer Science, Soran University, Soran, Iraq
hakem.beitollahi@soran.edu.iq

Received: 19-06-2025 Accepted: 01-09-2025 Published: 04-10-2025

adaptability, and operational efficiency, making SDN well-suited for complex network environments such as data centers and enterprise systems [3], [4]. However, the inherent centralization of SDN introduces critical security vulnerabilities. Specifically, the SDN controller acts as a single point of failure, the network's central control point. DDoS attacks targeting the controller can easily exhaust its resources, leading to network-wide outages. Fig. 1 illustrates the DoS attack on various planes of a SDN network. This vulnerability underscores the urgent need for effective DDoS attack detection and mitigation mechanisms within SDN environments [5], [6].

Traditional security controls, such as firewalls, signature-based intrusion detection systems, and rate-limiting techniques, often demonstrate limited effectiveness against contemporary DDoS attacks. These methods frequently fail to detect attacks that exploit protocol vulnerabilities or mimic legitimate traffic patterns. Furthermore, reactive security measures are typically implemented after significant damage has already occurred. In high-availability environments, where minimal downtime is crucial, such reactive approaches are inadequate [7], [8].

ML has emerged as a transformative approach to DDoS attack detection and prediction. Unlike traditional security solutions, ML algorithms analyze extensive network traffic data to identify patterns and anomalies that indicate malicious activity, including 0-day exploits and adaptive attack strategies [9]. Advanced techniques, such as deep learning and ensemble models, can detect subtle deviations from normal network behavior by analyzing complex features such as packet inter-arrival times and flow duration [10]. Furthermore, the predictive capabilities of ML enable proactive mitigation, allowing network administrators to neutralize threats before they escalate into full-scale attacks [6], [11].

Integrating ML-based detection mechanisms within SDN environments facilitates dynamic network responses to evolving attack patterns [12]. The centralized control of SDN allows ML models to provide real-time insights, enabling automated decisions that enhance network resilience and security [13], [14].

This study aims to address the aforementioned challenges by proposing a robust framework for DDoS attack detection and prediction specifically designed for SDN environments. The study's objectives are threefold:

1. Creation of a Comprehensive Statistical Dataset: To develop and publicly share a realistic dataset comprising both normal and attack traffic patterns relevant to SDN environments, thereby facilitating further research in this area.
2. Evaluation of ML Models: To implement and evaluate advanced ML techniques, including deep learning architectures and ensemble methods, to assess their effectiveness in real-time DDoS attack detection and prediction

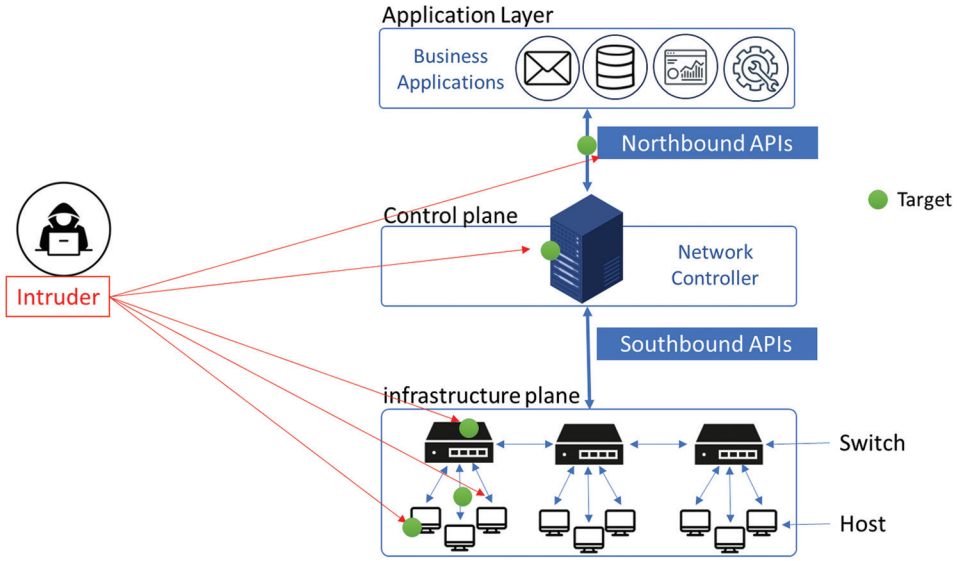

Fig. 1. Illustration of a denial of service attack targeting components of the Software Defined Networking architecture, including the centralized controller and data plane.

3. Comparative Analysis of Performance: To analyze the performance of selected ML models using key metrics such as accuracy, precision, computational efficiency, and scalability.

The main contributions of this study are:

- Realistic Dataset Creation: A publicly available dataset specifically designed for SDN environments, addressing the existing gap in realistic and diverse traffic patterns.
- Insights into ML Techniques: A comprehensive analysis of advanced ML methods, providing actionable insights into their respective strengths and weaknesses for DDoS attack detection.
- Enhanced Defense Framework: A practical, scalable, and adaptive solution for real-time DDoS attack mitigation, enhancing the resilience and reliability of SDN networks.

The rest of the paper is organized as follows: Section 2 reviews the related work with respect to the current techniques of DDoS detection and mitigation concerning the SDN environments and the respective limitations of those techniques. Section 3 describes in detail the proposed framework that also covers the creation of a dataset, ML methodologies, and how these techniques will be integrated into the SDN architecture. Section 5 discusses the experimental results, including the performance evaluation of different models and their comparison. Moreover, the discussion of the results, implications, and limitations is also discussed in this section. Finally, Section 6 concludes the paper.

2. RELATED WORKS

Numerous studies have explored the detection and mitigation of DDoS attacks in SDN environments using ML, deep learning, or hybrid techniques [15]–[18]. Although these methods demonstrate potential, they still exhibit certain limitations that necessitate further investigation.

Garba *et al.* [13] developed a framework that integrated ML-based detection into signature-based intrusion detection systems, achieving 99% classification accuracy with Decision Tree algorithms. However, their approach heavily relies on predefined attack signatures and supervised learning models, which significantly limits its adaptability to novel attack patterns not present in the training data.

Anley *et al.* [2] proposed an adaptive transfer learning-based CNN approach that improved detection accuracy across datasets. Nevertheless, this method incurs high computational overhead and requires extensive hyperparameter optimization, rendering it less suitable for real-time applications, particularly in resource-constrained environments.

Swami *et al.* [14] introduced an interquartile range threshold-based statistical approach for detecting spoofed flooding DDoS attacks. Although this method demonstrated reduced detection times, it struggles to accurately identify sophisticated attacks that closely resemble legitimate traffic patterns in dynamic scenarios, leading to a higher rate of false positives (FP). The reliance on static thresholds also limits the method's adaptability across diverse network environments. Al-Fayoumi and Abu Al-Haija [1] presented a lightweight detection model for MQTT-based low-rate DDoS attacks, achieving high accuracy. However, this model's narrow focus on specific attack types restricts its scalability and applicability to other DDoS categories.

Deep learning-based techniques are increasingly being utilized for DDoS detection in SDN environments. Clinton *et al.* [10] transformed network traffic into image data and achieved over 99% classification accuracy using image classification methods. However, this approach introduces significant computational complexity. The conversion of raw traffic to image formats increases processing time and is resource-intensive, making it unsuitable for high-throughput networks.

Songa and Karri [19] proposed a unified SDN framework employing feature elimination, clustering, and timeseries analysis, achieving 99.92% detection accuracy. Nevertheless, their work heavily relies on the CICDDoS2019 dataset, raising concerns about its generalization capability for practical implementation in diverse traffic scenarios.

Hybrid models that integrate multiple ML/DL techniques show considerable promise. For example, Mhamdi and Isa [6] explored a hybrid approach using a Deep Autoencoder in conjunction with a Random Forest classifier, achieving an anomaly detection rate exceeding 98%. However, the reliance on a centralized detection mechanism renders the system susceptible to controller-specific DDoS attacks, highlighting a critical vulnerability in SDN environments where the controller is a primary target. Furthermore, the substantial computational demands of hybrid approaches raise concerns about their scalability and feasibility for implementation in large, resource-constrained networks.

While ML-based approaches offer significant promise for DDoS attack detection, several challenges hinder their effective implementation within SDN environments. These challenges include high FP rates in anomaly detection, limited scalability of centralized detection, susceptibility of the SDN controller to targeted floods, and the inability of static models to adapt to evolving attack patterns in real-time. A key limitation is the scarcity of comprehensive and up-to-date datasets that accurately represent the unique characteristics of SDN traffic and modern DDoS attack strategies [10], [13]. Existing datasets often lack completeness or fail to encompass the wide range of attack scenarios, which limits the generalization capability of ML models. Furthermore, many studies do not evaluate advanced ML techniques in real time, dynamic SDN environments, resulting in substantial gaps in practical applicability [7], [20].

To address these limitations, our research focuses on the real-time deployment of trained ML models within an SDN architecture. In contrast to previous work, this study emphasizes scalable and adaptive detection mechanisms that operate efficiently in dynamic and heterogeneous network environments. The contributions presented in the following sections aim to mitigate the limitations of dataset dependency, centralized detection, and resource-intensive approaches without compromising detection accuracy and efficiency.

3. METHODOLOGY

This section details the design and implementation of our proposed framework for the detection of DDoS attacks within an SDN environment. The framework comprises an SDN controller, a comprehensive dataset encompassing both normal and malicious network traffic, a carefully selected set of statistical features to characterize traffic behavior, and various ensemble ML modules that utilize these features for real-time traffic classification. The subsequent subsections will explain the specifics of the SDN environment setup, the two-stage dataset creation methodology, the feature extraction process, and the integration of the ML-based classification engine with the SDN controller.

The key stages of the methodology are visually represented in the diagram depicted in Fig. 2, which highlights the sequential flow from the SDN environment setup to the immediate network-wide blocking of malicious traffic.

a. SDN setup: Depicts the "Ryu" controller, OpenFlow switch, and host configuration, including a dedicated web server.

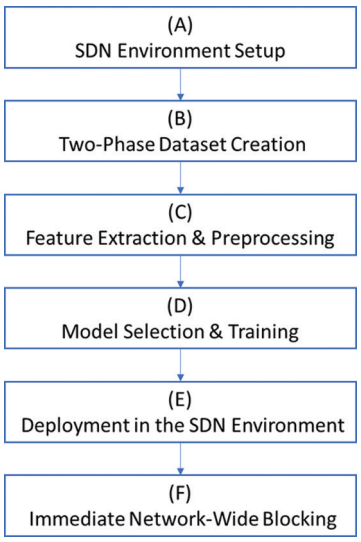

Fig. 2. Overview of the proposed methodology for distributed denial of service detection in Software Defined Networking, including dataset generation and model integration.

b. Two-phase dataset creation: Demonstrates the sequential generation of normal traffic and subsequent injection of malicious traffic, with packet captures labeled to create a comprehensive dataset.
c. Feature extraction and preprocessing: Details the computation of sixteen key features, followed by data cleaning and scaling procedures.
d. Model selection and training: Describes hyperparameter tuning, cross-validation techniques, and the selection of the optimal ML model from multiple candidates.
e. Deployment in SDN: Explains the distributed deployment of local ML classifiers on each server, the certificate authority (CA)-based encryption of IP lists, and their transmission to the controller for rule enforcement.
f. Immediate blocking: Illustrates the controller's rapid deployment of drop rules to the OpenFlow switch to mitigate attacks network-wide.

3.1. The Proposed Framework

The proposed framework leverages the centralized control plane paradigm inherent in SDN. The "Ryu" controller is employed in this study because of its Python-based modularity, adaptability for custom application development, and interoperability with various OpenFlow-enabled switches. The experimental network topology comprises multiple virtual hosts, a single OpenFlow-compliant switch, and the "Ryu" controller, which manages all flow rules. Specifically, one host functions as a web server hosting a basic HTTP application, while the remaining hosts generate both normal

and malicious traffic. OpenFlow facilitates communication between the switch and the hosts. The switch transmits flow-level statistics to the "Ryu" controller, which, in turn, dictates forwarding policies, installs flow rules, and implements traffic shaping or blocking. This centralized control plane empowers network administrators to enforce granular security policies and perform real-time anomaly detection. Furthermore, the host with interface "h2-eth0" is designated as a monitoring point to observe traffic, analyze packets, and relay aggregated statistics to the classification module. This design aims to provide the "Ryu" controller with comprehensive network awareness, enabling it to apply immediate countermeasures upon detecting malicious traffic sources.

To accurately characterize the behavior of both normal and attack traffic, the system extracts 15 statistical features from observed flows. Following each capture period, the packet capture script gathers and normalizes a range of metrics for each active source IP (excluding the web server). These metrics are then transformed into a set of features that serve as the basis for traffic classification. Each feature is designed to act as a potential indicator for distinguishing between normal and malicious DDoS traffic.

The 15 attributes constitute a comprehensive set of measures intended to quantify both high-level and fine-grained characteristics of network traffic. Specifically, rate-based attributes (e.g., Request Rate, Packet Arrival Rate) are used to identify volumetric traffic surges, while connection-oriented metrics (e.g., TCP SYN Count, Failed Connection Attempts) are used to detect manipulations of the TCP handshake process commonly exploited in DDoS attacks. Furthermore, distribution-based attributes (e.g., Unique IP Count, Port Usage Distribution) provide insights into the scope of malicious host scanning activities. Each attribute's mathematical derivation, practical significance, and specific utility in DDoS detection and mitigation are detailed in the subsequent subsections.

1. Request rate: The Request Rate measures the number of application-level requests (e.g., HTTP, FTP, or other protocol-specific requests) made by a source IP per unit time. Formally,

$$R_{req} = \frac{N_{req}}{T_{elapsed}} \tag{1}$$

Where N_{req} is the total number of requests observed during the capture interval and Telapsed is the duration of the interval (in seconds). High request rates are generally a sign of application-layer DDoS (e.g., HTTP floods). By monitoring (R_{req}), it is straightforward to detect hosts generating more requests than normal.

2. Packet arrival rate: The packet arrival rate (R_{pkt}) quantifies how frequently a host transmits packets in seconds:

$$R_{pkt} = \frac{N_{pkt}}{T_{elapsed}} \tag{2}$$

In volumetric DDoS attacks, attackers flood the network with an unusually large volume of packets. A sudden surge in (R_{pkt}) thus indicates beforehand that there can be a flood, marking normal bursts apart from more sustained malicious traffic.

3. Download rate: The download rate (R_{bytes}) is the average rate of data being transferred from a host over time:

$$R_{bytes} = \frac{B_{total}}{T_{elapsed}} \tag{3}$$

Here, B_{total} is the quantity of bytes received by the host during the capture window. Bandwidth-depletion DDoS attacks attempt to exhaust a victim's network link capacity by transferring large volumes of data. Tracking (R_{bytes}) helps identify this behavior.

4. Uptime: The uptime (U_{time}) measure indicates that the duration a host has been actively sending traffic during the period observed:

$$U_{time} = T_{last\_packet} - T_{first\_packet} \tag{4}$$

DDoS attacks in other instances are composed of high activity sustained over an extended period of time. Alternatively, legitimate clients will have more sporadic or short sessions. Thus, continually high Uptime can be an indicator of ongoing attack traffic as opposed to regular user-initiated behavior.

5. Flow duration variability: (V_{flow}) is the standard deviation of flow durations across hosts. If each flow i has duration Di, and $\bar{D}$ is their mean, then:

$$V_{flow} = \sqrt{\frac{1}{N_{flow}} \sum_{i=1}^{N_{flow}} (D_i - \bar{D})^2} \tag{5}$$

Although there are legitimate long-lived and persistent flows (e.g., media streaming), the flows that result from

DDoS attacks are normally high rates of extremely short or unusual flows. Therefore, analyzing (Vf low) reveals patterns indicating bursty or inconsistent traffic typical of malicious activity

6. Retransmission rate: The retransmission rate ($R_{retrans}$) is the amount of packets retransmitted by a host per second:

$$R_{bytes} = \frac{N_{retrans}}{T_{elapsed}} \tag{6}$$

Misbehaving activities such as SYN floods or poorly misconfigured bots could generate a lot of duplicate packets or half-opened connections. An elevated ($R_{retrans}$) can signal malicious floods or repeated attempts at scanning for a victim server without completing up connections.

7. Unique IP count: The unique IP count ($C_{Unique\ IP}$) reveals how many distinct destination IP addresses a source host target:

$$C_{Unique\_IP} = \left|\left\{IP_{dest}\right\}\right| \tag{7}$$

In certain DDoS cases (or scanning attacks), an aggressive host rapidly probes many different IPs. However, a normal user tends to connect to fewer places in a short span of time, thus making an unusually high (CUnique IP) suspicious.

8. Sequential request patterns: Sequential Request Patterns (S seq) capture repeated requests to the same port from one host. Specifically,

$$S_{seq} = Max\left(\left\{N_{req}(P)\right\}\right) \tag{8}$$

Where Nreq(P) is the frequency of a host's repeated requests for the same port PPP consecutively. Automated scripts usually use repeated instances of the same actions that repeatedly hammer the same service endpoint a classic indication of possible DDoS or brute-force attacks.

9. Flow count per host: The flow count per host (Nflows) is the sum of flows originated by a source:

$$N_{flows} = \sum_{i-1}^{N_{sessions}} 1 \tag{9}$$

Malicious hosts may establish dozens or hundreds of ephemeral flows in an attempt to overwhelm a server's bandwidth. High rates of flows over a brief period are therefore an extremely strong indication of possible DDoS or network scanning.

10. Host communication frequency: Host Communication Frequency (fcomm) measures how many unique hosts a source communicates with per second:

$$f_{comm} = \frac{N_{hosts}}{T_{elapsed}} \tag{10}$$

This feature indicates whether a source is quickly propagating traffic to multiple targets (e.g., in a spread-based attack or scanning approach). In benign use cases, one typically sees smaller values that rise more slowly.

11. Port usage distribution: The Port Usage Distribution (Dports) gives the number of distinct destination ports utilized:

$$D_{ports} = |\{P_{dest}\}| \tag{11}$$

Attackers might attempt multi-port scanning or flooding to identify open services to attack. If one source is connecting to an unusually high number of different ports, it is either malicious probing or an advanced DDoS attack targeting multiple services.

12. TCP SYN count: The TCP SYN Count (Nsyn) measures how many TCP SYN packets a host sends.

$$N_{syn} = \sum_{i-1}^{N_{packets}} \delta_{SYN}(i) \tag{12}$$

Where δSYN (i) is 1 if the i-th packet is a SYN flag set, and 0 otherwise. SYN flood attacks are one of the most common types of DDoS, therefore large SYN counts within a short time interval a sure sign of potential flooding.

13. TCP ACK count: The TCP ACK Count (NACK) is the analogous measure for ACK packets:

$$N_{ACK} = \sum_{i-1}^{N_{packets}} \delta_{ACK}(i) \tag{13}$$

Although the ACK flag alone is not always malicious, analyzing its relationship to SYN packets (e.g., the ratio of SYN to ACK) can highlight incomplete handshakes or unusual session behaviors that typify DDoS attempts.

14. Connections per second: Connections per second (Rconn) is the sum of new connections established by a host in one second:

$$R_{conn} = \frac{N_{new\_conn}}{T_{elapsed}} \quad (14)$$

Flooding attackers or active scanners have the tendency to establish many connections in quick succession to overload a target. Thus, a high (Rconn) distinguishes probable attackers from valid clients, which establish connections at a slower pace.

15. Failed connection attempts: Failed Connection Attempts (N_{failed}) refer to the number of connections attempted which fail to finish the TCP handshake. Monitoring can be done as follows:

$$N_{failed} = \frac{Failed\ Attempts}{Telpased} \quad (15)$$

Where NSYN is the number of SYN packets that were sent, and NSYN_ACKN is the number that was responded to with a SYN-ACK. In partial-handshake attacks (e.g., SYN floods), the attacker never finishes the server's response, so there are many incomplete or failed sessions.

The selection of features for this study was guided by a literature review as well as domain knowledge pertaining to DDoS attack patterns. Request Rate, Packet Arrival Rate, TCP SYN Count, and Port Usage Distribution are a few of the features used extensively in previous studies on DDoS detection in SDN and traditional networks [2], [13], [21], [22]. But on top of that, we also have some sophisticated or innovative features such as Sequential Request Patterns and Failed Connection Attempts that exceed the typical ones in benchmark research. The reasoning behind these features is to attack application-layer session behavior and incomplete handshake patterns, especially relevant to SDN environments but commonly omitted in earlier research works. That combined approach provides more comprehensive and SDN-focused detection capacity.

3.2. Dataset Creation

The development of a labeled dataset is crucial for the effective training and evaluation of ML models in DDoS attack detection. To generate realistic traffic data, we implemented a two-phase approach over 6 days: An initial phase for normal traffic generation, followed by a phase dedicated to creating malicious traffic.

As shown in Fig. 3, the normal traffic phase spanned 2 days, during which typical user interactions with the web server were simulated. 11,186 aggregated samples which users sent normal traffic to the web server during these 2 days. These activities included browsing, page requests, occasional form submissions, and SSH interactions. Concurrently, background operations such as normal file downloads and MySQL queries were initiated to emulate a more diverse network environment.

Upon capturing sufficient normal traffic data, the network configuration was transitioned to a malicious environment, which was sustained for 4 consecutive days. During this phase, various DDoS attack techniques were employed. A custom, multi-threaded Python script was utilized to generate continuous multi-threaded HTTP floods targeting the web server. In addition, high-volume SYN floods were launched using tools such as hping3 and iperf, alongside application layer resource exhaustion attacks. These attacks were designed to mimic common DDoS characteristics, 5 such as rapid request generation, incomplete TCP handshakes, and excessive consumption of server resources. In total, 11,469 aggregated samples in which bot machines participate in creating malicious traffic during the 4 days.

Throughout both phases, packet capture was consistently performed on the monitor host using Scapy. This ensured that all network activity originating from a source IP within a defined time window was recorded. Each captured network session was then labeled as either normal (0) or attack (1), resulting in a comprehensive dataset of benign and malicious network traffic.

The complete dataset used in this study is publicly available at https://www.kaggle.com/datasets/aramsaleem21/sddos-sdn-2025.

To clarify our choice of using a custom dataset instead of public alternatives like CICDDoS2019. A few of the engineered features used in this study are not computable or aggregable directly from the CICDDoS2019 dataset due to the granularity and content limitations of the available data. For example, Request Rate (req/sec) and Sequential Request Patterns depend on the application layer to recognize and count the distinct user interactions and their sequence. CICDDoS2019 does not have this application-level visibility.

Retransmission Rate depends on monitoring TCP sequence numbers to detect duplicate transmissions, which is not feasible without raw packet captures and sequence monitoring

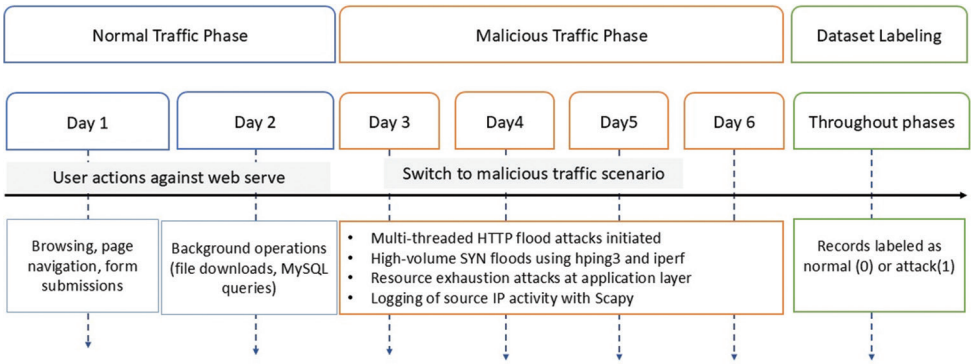

Fig. 3. (Dataset creation) highlights the two-phase normal vs. malicious data gathering.

not available in CICDDoS2019. Failed Connection Attempts Rate must maintain connection state on a per-host basis, i.e., unmatched SYNs or aborted handshakes through RST/FIN flags. CICDDoS2019 does not maintain this level of TCP control information.

In contrast, our dataset was built from full packet captures of a controlled SDN setup and captures both application and transport-layer behavior for a diverse set of traffic scenarios. This provides the ability to extract more expressive, real-time, and SDN-specific traffic features and hence provides a more fertile ground for evaluating advanced machine learning (ML) models under conditions more closely aligned with real-world network operation.

Following the collection of a comprehensive dataset comprising DDoS and normal traffic, the subsequent step involved data preprocessing. This process transformed raw packet-level data into a structured dataset suitable for ML-based classification. The following subsections detail the preprocessing and validation procedures.

1. Data cleaning: It includes two stages. (a) Removal of Duplicates: Overlapping capture intervals and sniffer buffering occasionally resulted in duplicate entries. These were identified and removed using (Timestamp, Source IP, Destination IP) tuples. (b) Handling Missing Values: Records with undetermined features due to missing session information were either dropped or imputed using the mean/median, depending on whether the ratio of missing data was low (1%).
2. Label encoding (binary labels): Each traffic sample was labeled as "0" for benign or "1" for attack. To mitigate potential label noise (e.g., suspicious-looking but benign traffic), cross-validation was performed using known malicious tools (iperf, hping3) during the attack phase.
3. Normalization/scaling: It includes two steps. (a) Feature scaling: To prevent feature scale sensitivity in certain ML algorithms, MinMax or Standard scaling was applied where necessary. This ensured that no single feature, such as raw byte counts, unduly influenced model training. (b) Outlier Treatment: Extreme values (e.g., an extremely high Packet Arrival Rate from a short, intense attack) can skew model training. While outliers can indicate attacks, they were capped at a maximum value (e.g., the 99.9th percentile) to stabilize training without eliminating the underlying malicious patterns.

This preprocessing pipeline ensured the final dataset's cleanliness, consistency, and appropriate scaling before classification.

3.3. Model Selection and Training Strategy

To comprehensively evaluate the effectiveness of ML techniques for DDoS detection, we selected eight classification models:

- Gradient boosting ensembles: LightGBM, XGBoost, CatBoost, and Gradient Boosting Decision Trees. These tree-based ensembles are recognized for their strong performance in handling diverse feature sets, robustness against overfitting due to boosting mechanisms, and interpretability of feature importance.
- Ensemble methods: AdaBoost and Bagging (Bootstrap Aggregating). While conceptually related to gradient boosting, these methods typically combine multiple weak estimators (e.g., Decision Trees) to achieve a favorable bias-variance trade-off.
- One-class SVM: A classical anomaly detection algorithm that learns the boundary of normal data to identify outliers. Although theoretically appealing for 0-day DDoS attack detection, it may exhibit a high FP rate if legitimate traffic deviates significantly from the training reference.
- Bayesian networks: A graphical probabilistic model that represents dependencies among features. While

often surpassed by ensemble methods in high-volume applications, Bayesian networks offer explainable insights into conditional probabilities of attack versus benign behavior.

The preprocessed dataset was partitioned into training (70%), validation (15%), and test (15%) sets. This partitioning strategy ensures an unbiased evaluation of model performance on unseen data.

Within the training set, k-fold cross-validation was employed. In each iteration, a model was trained on k-1 folds, and performance metrics were computed on the remaining fold. This process was repeated k times, and the resulting average metrics mitigate variance associated with a single train-test split.

We optimized the hyperparameters of each model to maximize performance. For Gradient Boosting models, a grid search was conducted across the number of estimators, learning rate, maximum tree depth (max depth), and subsample ratio to identify optimal configurations. For AdaBoost and Bagging, we varied the base estimator depth and the number of estimators to achieve a balance between model complexity and overfitting. For the One-Class SVM, we experimented with RBF and linear kernels and tuned the "nu" parameter, which controls the fraction of outliers. Finally, for Bayesian Networks, we explored score-based and constraint-based structure learning algorithms and employed partial hill-climbing to identify effective network structures. The accuracy, area under curve (AUC), and F1-score metrics guided the selection of optimal hyperparameter configurations. The final model configurations were then retrained using the combined training and validation sets and evaluated on the held-out test set.

4. IMPLEMENTATION IN THE SDN ENVIRONMENT

During the deployment phase, each web server within the SDN environment is equipped with its own ML model for DDoS detection. Upon detecting an attack signature, a web server promptly notifies the SDN controller. This design choice enhances scalability by distributing computational load to the servers and reduces detection latency through real-time classification of incoming traffic at the network edge, enabling instantaneous detection and immediate reporting of malicious IP addresses.

When a server identifies an IP address as malicious, the SDN controller immediately enforces drop rules across the network to mitigate potential damage. To ensure secure communication of identified malicious IP addresses, the web servers utilize a CA-based security mechanism, restricting participation in the detection and mitigation process to trusted entities. The SDN controller then installs drop rules on the OpenFlow switches throughout the network, effectively isolating attackers from all hosts within the SDN environment. This distributed deployment of classifiers serves two primary functions:

1. Controller load reduction: By delegating attack detection to individual web servers, the SDN controller is relieved of the burden of inspecting every request. This architecture minimizes the risk of overwhelming the controller with high-volume traffic and eliminates latency associated with packet forwarding to and from a centralized analysis unit.
2. Scalability and low-latency detection: As the number of web servers increases (e.g., due to data center expansion), the number of detection nodes scales proportionally. Each server performs traffic classification independently, imposing minimal overhead on the central controller and facilitating real-time or low-latency detection.

4.1. Local ML Detection

Each web server continuously monitors incoming traffic and analyzes features such as request rate, packet size, and unusual TCP flags. Upon detecting suspicious activity within a "five-second" window, the local ML model generates a list of the responsible IP addresses. This localized detection approach ensures rapid response times and avoids bottlenecks associated with centralized traffic analysis.

4.2. Secure IP List Transmission with CA-Based Sessions

Before transmitting the list of malicious IP addresses to the SDN controller, the web server establishes a secure communication channel using CA-issued certificates. The session establishment process involves the server first validating the controller's certificate (issued by a trusted CA). Subsequently, the server encrypts a randomly generated session key with the controller's public key. The controller then decrypts this encrypted session key using its corresponding private key, thereby acquiring the session key required for symmetric encryption. Finally, the server encrypts the list of malicious IP addresses using AES-256 encryption with the established session key, ensuring confidentiality and integrity during transmission.

4.3. Immediate Network-Wide Blocking

Once the IP list is decrypted, the SDN controller enforces network-wide blocking by commanding the OpenFlow

switches to discard all packets from the labeled IP addresses. The blocking occurs in a short time and thus prevents malicious hosts from further flooding or compromising network resources. Because each web server handles traffic classification independently, the network gains horizontal scalability: Adding more servers proportionally enhances the overall detection capability of the network, while the central controller remains focused on coordinating drop rules rather than performing high-volume traffic analysis.

Fig. 4 provides a more step-by-step illustration of the entire process, from when an attack is identified by a server to where the bad traffic is dropped in the network fabric.

In such an architecture, real-time detection at the server level prevents making the SDN controller a computational bottleneck. Whereas CA-based session establishment ensures mutual authentication and secure key exchange, AES256 encryption protects the IP lists against eavesdropping and tampering. With fast blocking of malicious traffic at the switch level, the entire network becomes resilient against DDoS disruption, illustrating how a distributed model of detection nicely complements SDN's centralization of policy enforcement.

5. RESULTS

This section outlines the results obtained from evaluating the eight ML models on the SDN-based DDoS dataset. Model performance is assessed using a variety of metrics, and further analysis is conducted using confusion matrices and ROC curves. The findings provide a comparative analysis of each model's ability to effectively detect and classify DDoS attacks. Next, a comprehensive discussion of the results is presented.

5.1. Overall Classification Outcomes

Table 1 provides a comparative overview of the performance of the eight ML models, evaluated on the SDN-based DDoS dataset. The performance of each model was assessed using the following metrics: accuracy, precision, recall, F1-score, AUC, Cohen's Kappa, Matthews correlation coefficient (MCC), FP rate (FPR), false-negative rate (FNR), and cross-validated accuracy.

5.2. Confusion Matrices: Minimal Misclassifications

To better visualize model performance, confusion matrices were generated for each of the eight classifiers as shown in Fig. 5. These matrices illustrate the distribution of true positives, FP, true negatives (TN), and false negatives (FN). The confusion matrices highlight that ensemble methods, particularly gradient boosting techniques, achieve a strong balance between high detection rates and low false alarm rates.

5.3. Cross-Validation and Consistency

Cross-validation accuracies remain consistently above 0.998 for LightGBM, CatBoost, Bagging, and XGBoost. This consistency across different folds indicates that their

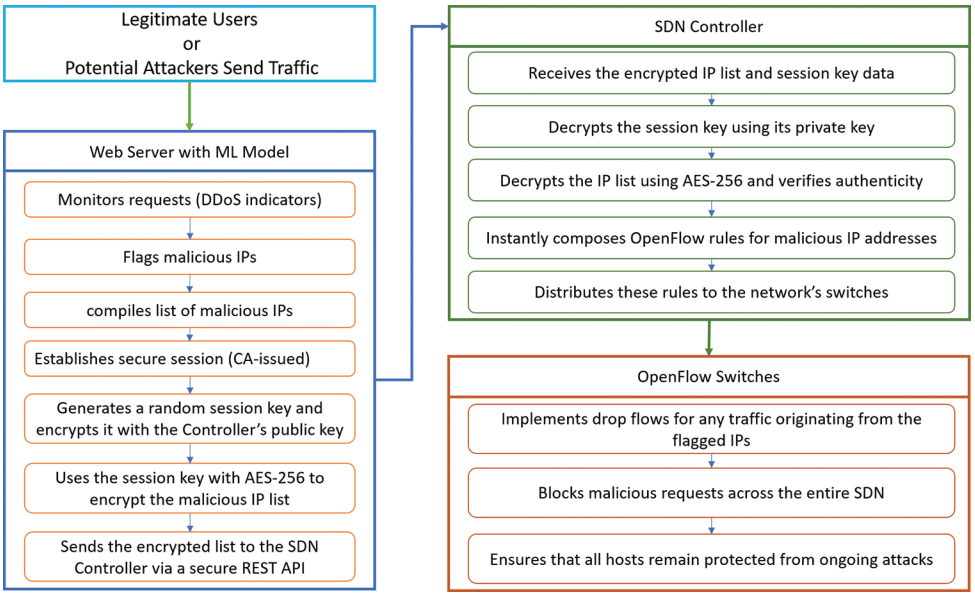

Fig. 4. Server-centric real-time distributed denial of service detection with CA-based security and immediate drop rules installed by the Software Defined Networking controller in this design, real-time detection at the server.

Table 1: Classification performance metrics across different models

Model	Accuracy	Precision	Recall	F1-Score	AUC	Cohen's Kappa	MCC	FPR	FNR	Cross-validated accuracy
LightGBM	0.9998	0.9999	0.9996	0.9998	0.9999	0.9996	0.9996	0.0001	0.0004	0.9992
XGBoost	0.9989	0.9996	0.9983	0.9989	0.9996	0.9978	0.9978	0.0004	0.0017	0.9985
CatBoost	0.9993	0.9999	0.9987	0.9993	0.9999	0.9987	0.9987	0.0001	0.0013	0.9986
Gradient boosting decision trees	0.9988	0.9991	0.9987	0.9989	0.9992	0.9978	0.9978	0.0009	0.0013	0.9993
AdaBoost	0.9987	0.9991	0.9983	0.9987	0.9992	0.9974	0.9974	0.0009	0.0017	0.9989
Bagging (bootstrap aggregating)	0.9991	0.9996	0.9987	0.9991	0.9999	0.9982	0.9982	0.0004	0.0013	0.9992
One-class svm	0.9823	0.9794	0.9998	0.9895	0.9998	0.9322	0.9343	0.1077	0.0002	0.9505
Bayesian Network	0.9982	1.0000	0.9965	0.9983	0.9983	0.9965	0.9965	0.0000	0.0035	0.9984

MCC: Matthews correlation coefficient, FPR: False-positive rate, FNR: False-negative rate, AUC: Area under curve

performance is robust and not attributable to overfitting on a specific train-test split. In contrast, cross-validation accuracy for One-Class SVM drops to 0.9505, revealing higher performance variability and emphasizing the importance of meticulous threshold tuning in anomaly-based detection systems.

6. DISCUSSION

The results illustrate clear differences in what each ML method is capable of doing for DDoS detection on SDN networks. With a dataset containing both traffic volume and behavior attributes, we can see which models are more accurate and more capable of preventing FP. The following section discusses these findings in more detail, starting with ensemble method advantages.

6.1. Ensemble Methods and Their Dominance

The results clearly demonstrate the superior performance of ensemble-based approaches, particularly LightGBM and CatBoost. Their near-perfect scores highlight two key strengths:

1. Robust feature handling: The 15 features, encompassing volumetric, connection, and distribution-based metrics, are well-suited to tree-based algorithms that inherently manage mixed data distributions.
2. Resilience to overfitting: Gradient boosting models iteratively construct weak learners (decision trees), enabling them to effectively identify subtle patterns while mitigating overfitting to noise.

The iterative refinement of gradient boosting, combined with the feature-rich dataset used in this study, likely contributes to the observed 0.9999 AUC scores in many cases. These high AUC values suggest that the malicious and benign traffic classes are distinctly separated within this dataset.

In computational efficiency, ensemble models such as LightGBM and CatBoost had fast convergence during training and low-latency predictions, which is suitable for real-time SDN deployments where response time is critical.

6.2. Trade-Offs with Anomaly Detection (One-Class SVM)

While One-Class SVM worked very well at identifying almost all attacks (recall = 0.9998), the sharp increase in FP (FPR = 0.1077) reflects a serious deficiency of anomaly-based detectors. It suggests that, when legitimate traffic significantly deviates from the baseline model, the model will end up marking benign requests as malicious. In operational scenarios, high FPR can lead to resource overhead in investigating benign events.

6.3. Bayesian Networks and Interpretability

Interestingly, the Bayesian Network achieved perfect precision (1.0000) at the expense of having a fairly elevated FNR (0.0035). That is, while it never misclassified benign traffic as an attack, it occasionally missed real attacks. Bayesian Networks do offer interpretability in the sense that they are graphical models whose conditional dependencies among features can make it easy to understand. For advanced research that aims to discover the underlying causal relations for attack patterns, this feature could provide new analytical contributions beyond bare classification.

Existing research has also explored the use of gradient boosting models such as LightGBM and CatBoost for DDoS detection. The majority of research, however, evaluates these models on typical data such as CICDDoS2019 or UNSW-15 and does not consider SDN-special traffic or real-time deployment. In contrast to our method, which is aimed at real SDN deployments and integrates these models with

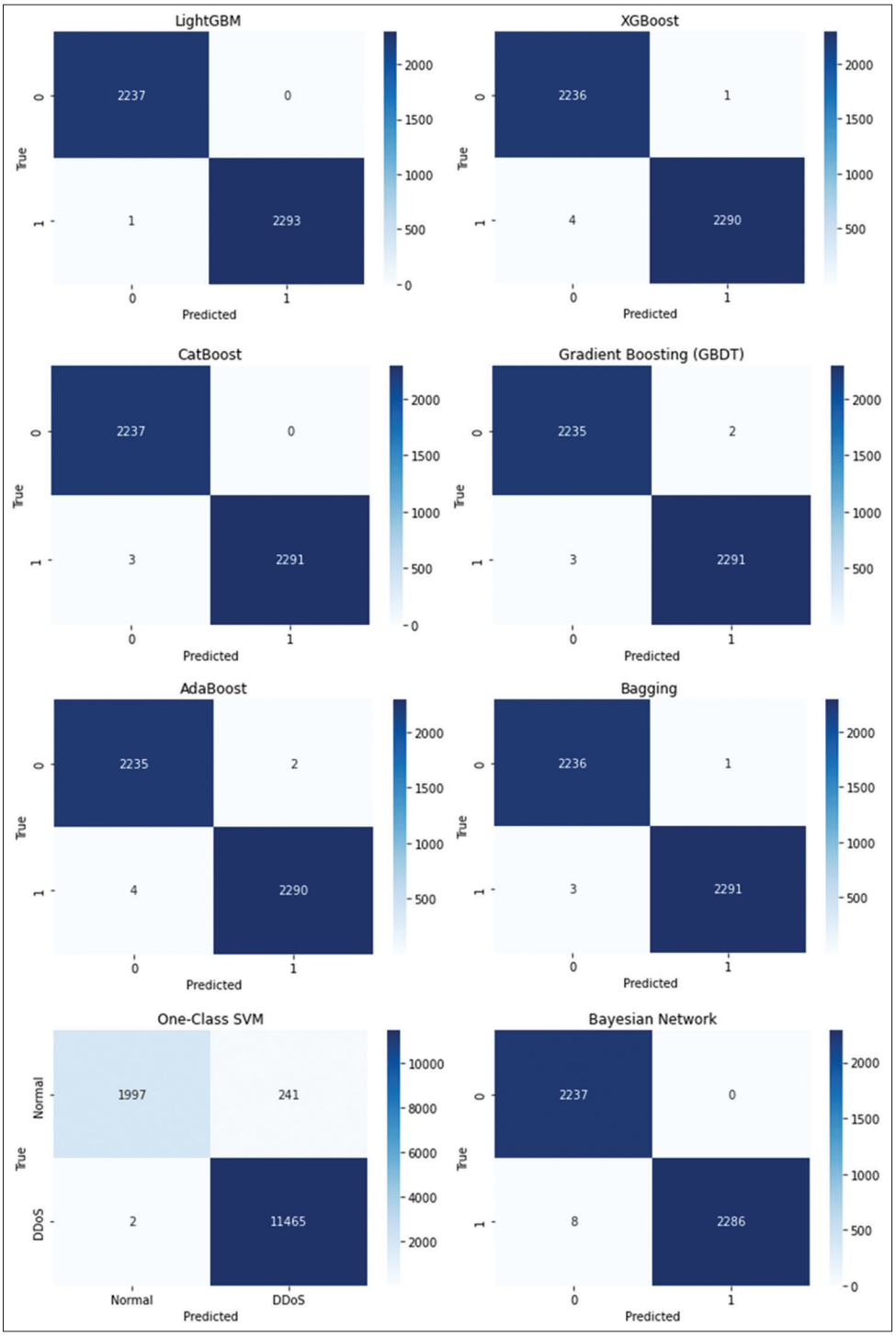

Fig. 5. Confusion matrices for each classifier showing the distribution of true positives, false positives, true negatives, and false.

an ongoing mitigation process, existing literature does not. Table 2 indicates a comparison between this proposal and existing works.

6.4. Implications for Advanced SDN-Based Research

With the network-level perspective of an SDN controller, high-performance ensemble models can be combined to offer

Table 2: Accuracy comparison of LightGBM and XGBoost across prior DDoS detection studies

Study	Model used	Dataset	Accuracy (%)
Han *et al*. [23]	LightGBM	CICDDoS2019	98.39
	XGBoost		97.95
Vaishali and Naik [24]	LightGBM	UNSW-15	99.18
Rozam and Riasetiawan [25]	XGBoost	A time series database	99.84
This study	LightGBM	Dataset used in this study	99.98
	XGBoost		99.89

DDoS: Distributed denial of service

a scalable and precise method for handling evolving DDoS threats. The distributed detection model with each server running the final classifier locally and securely notifying the controller of malicious IPs ensures low latency and effective utilization of resources. For mission-critical or large-scale deployments (e.g., data centers, cloud infrastructures), an extremely low FPR (essentially 0.0000) is required to avoid disrupting rightful traffic. Such minimal FPR (very nearly 0.0001) achieved by LightGBM and CatBoost shows that both models are well positioned for highly available scenarios fulfilling strict service-level agreements.

6.5. Limitations and Future Work

Although the ensemble methods have produced excellent results for this dataset, some of the potential avenues remain available for future work:

4. Temporal and incremental learning: Real networks have continuous traffic fluctuations. Online or incremental learning methods could be further explored to improve adaptability over time.
5. Multi-controller SDN: Even though in this research a single controller (Ryu) has been utilized, modern SDN deployments may have multiple controllers for fault tolerance and load balance. Classifier consistency in Mult controllers is an appealing area of future research.
6. Encrypted traffic analysis: Many application-layer protocols employ TLS/SSL encryption. Future work could be focused on feature extraction from encrypted flows, e.g., analysis of metadata or packet timing without analyzing payloads.

7. CONCLUSION

This study presents an ensemble-based DDoS detection framework tailored for SDN environments. It introduces a custom dataset collected in two phases, covering both normal and attack traffic and extracts 15 statistical features relevant to SDN traffic behavior. Eight ML models were evaluated, demonstrating high accuracy and suitability for real-time SDN deployment. Ensemble-based models, particularly LightGBM and CatBoost, significantly surpassed the performance of the other algorithms, achieving exceptional detection metrics (accuracy 0.9998, precision = 0.9999, and 0.0001 false alarms). These results underscore the suitability of advanced, tree-based methods for effectively processing the high-dimensional and heterogeneous data characteristic of network traffic logs. From an architectural perspective, distributing the ML detection process across web servers minimizes the processing load on the SDN controller and facilitates rapid attack detection. The use of CA-based secure delivery of IP lists ensures that only authenticated entities (controller and servers) are involved in the mitigation process. Upon receiving these IP lists, the controller implements network-wide blocking in real time, providing robust protection for the entire SDN environment against large-scale DDoS attacks.

REFERENCES

[1] M. Al-Fayoumi and Q. Abu Al-Haija. "Capturing low-rate DDoS attack based on MQTT protocol in software defined-IoT environment". *Array*, vol. 19, p. 100316, 2023.

[2] M. B. Anley, A. Genovese, D. Agostinello and V. Piuri. "Robust DDoS attack detection with adaptive transfer learning". *Computers and Security*, vol. 144, p. 103962, 2024.

[3] X. Etxezarreta, I. Garitano, M. Iturbe and U. Zurutuza. "Software-defined networking approaches for intrusion response in industrial control systems: A survey". *International Journal of Critical Infrastructure Protection*, vol. 42, p. 100615, 2023.

[4] N. N. Josbert, M. Wei, P. Wang and A. Rafiq. "A look into smart factory for industrial IoT driven by SDN technology: A comprehensive survey of taxonomy, architectures, issues and future research orientations". *Journal of King Saud University - Computer and Information Sciences,* vol. 36, no. 5, p. 102069, 2024.

[5] K. K. Karmakar, V. Varadharajan, M. Hitchens, U. Tupakula and P. Sariputra. "A trust-aware openflow switching framework for software defined networks (SDN)". *Computer Networks*, vol. 237, p. 110109, 2023.

[6] L. Mhamdi and M. M. Isa. "Securing SDN: Hybrid autoencoder-random forest for intrusion detection and attack mitigation". *Journal of Network and Computer Applications*, vol. 225, p. 103868, 2024.

[7] A. T. Phu, B. Li, F. Ullah, T. Ul Huque, R. Naha, M. A. Babar and H. Nguyen. "Defending SDN against packet injection attacks using deep learning". *Computer Networks*, vol. 234, p. 109935, 2023.

[8] X. Qin, F. Jiang, X. Qin, L. Ge, M. Lu and R. Doss. "Cgan-based cyber deception framework against reconnaissance attacks in ICS". *Computer Networks*, vol. 251, p. 110655, 2024.

[9] Z. Chen, M. Simsek, B. Kantarci, M. Bagheri and P. Djukic. "Machine learning-enabled hybrid intrusion detection system with host data transformation and an advanced two-stage classifier". *Computer Networks*, vol. 250, p. 110576, 2024.

[10] U. B. Clinton, N. Hoque and K. Robindro Singh. "Classification

of DDoS attack traffic on SDN network environment using deep learning". *Cybersecurity*, vol. 7, no. 1, p. 23, 2024.

[11] G. Srinivasa Rao, P. Santosh Kumar Patra, V. A. Narayana, A. Raji Reddy, G. N. V. Vibhav Reddy and D. Eshwar. "DDoSnet: Detection and prediction of DDoS attacks from realistic multidimensional dataset in IoT network environment". *Egyptian Informatics Journal*, vol. 27, p. 100526, 2024.

[12] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir and D. Draheim. "Towards a machine learning-based framework for DDoS attack detection in software-defined IoT (SD-IoT) networks". *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106432, 2023.

[13] 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.

[14] R. Swami, M. Dave and V. Ranga. "IQR-based approach for DDoS detection and mitigation in SDN". *Defence Technology*, vol. 25, pp. 76-87, 2023.

[15] A. Hirsi, L. Audah, A. Salh, M. A. Alhartomi and S. Ahmed. "Detecting DDoS threats using supervised machine learning for traffic classification in software defined networking". *IEEE Access*, vol. 12, pp. 166675-166702, 2024.

[16] A. A. Alashhab, M. S. Zahid, B. Isyaku, A. A. Elnour, W. Nagmeldin, A. Abdelmaboud, T. A. A. Abdullah and U. D. Maiwada. "Enhancing DDoS attack detection and mitigation in SDN using an ensemble online machine learning model". *IEEE Access*, vol. 12, pp. 51630-51649, 2024.

[17] H. M. Belachew, M. Y. Beyene, A. B. Desta, B. T. Alemu, S. S. Musa and A. J. Muhammed. "Design a robust DDoS attack detection and mitigation scheme in SDN-edge-IoT by leveraging machine learning". *IEEE Access*, vol. 13, pp. 10194-10214, 2025.

[18] Y. S. N. Fotse, V. K. Tchendji and M. Velempini. "Federated learning based DDoS attacks detection in large scale software-defined network". *IEEE Transactions on Computers*, vol. 74, no. 1, pp. 101-115, 2025.

[19] A. V. Songa and G. R. Karri. "An integrated SDN framework for early detection of DDoS attacks in cloud computing". *Journal of Cloud Computing*, vol. 13, p. 64, 2024.

[20] R. Abu Bakar, L. De Marinis, F. Cugini and F. Paolucci. "FTG-net-E: A hierarchical ensemble graph neural network for DDoS attack detection". *Computer Networks*, vol. 250, p. 110508, 2024.

[21] A. Ghorbanali and M. K. Sohrabi. "A comprehensive survey on deep learning-based approaches for multimodal sentiment analysis". *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1479-1512, 2023.

[22] F. Nawshin, R. Gad, D. Unal, A. K. Al-Ali and P. N. Suganthan. "Malware detection for mobile computing using secure and privacy-preserving machine learning approaches: A comprehensive survey". *Computers and Electrical Engineering,* vol. 117, p. 109233, 2024.

[23] D. Han, H. Li and X. Fu. "Reflective distributed denial of service detection: A Novel Model utilizing binary particle swarm optimization—Simulated annealing for feature selection and gray wolf optimization-optimized LightGBM algorithm". *Sensors*, vol. 24, no. 19, p. 6179, 2024.

[24] R. Vaishali and S. M. Naik. "A Novel LightGBM-Bayesian Approach for DDoS Detection in SDN Environments". In: *2024 Moratuwa Engineering Research Conference (MERCon)*. pp. 7-12. 2024.

[25] N. Rozam and M. Riasetiawan. "XGBoost Classifier for DDoS attack detection in software defined network using sFlow protocol". *International Journal on Advanced Science, Engineering and Information Technology,* vol. 13, pp. 718-725, 2023.