International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(5s) 2026 pp. 896-918
https://doi.org/10.70917/ijcisim-2026-2812
Article

Design and Implementation of a Real-Time Multi-Vector DDoS Detection and Prevention Framework Using ML-Ready Detection Logic and nftables

Santosh Jaykumar Kalegore[1], Sunil Mane[2]

[1,2]COEP Technological University Pune
Mail id: sjk22.comp@coeptech.ac.in
ORCID iD: https://orcid.org/0009-0002-3473-1162

Abstract: This paper presents the design and implementation of a real-time, multi-vector DDoS detection and prevention framework for cloud environments. The system integrates machine-learning–ready anomaly detection logic with nftables for automated mitigation and uses Redis as a message bus for synchronous alert communication. The framework was tested using Apache Bench and hping3 to simulate multiple DDoS attacks, including SYN flood, UDP flood, HTTP GET flood, and ICMP flood. Experimental results demonstrated that the proposed system effectively restores near-normal throughput and response times under attack conditions. Specifically, throughput recovery averaged 92.7%, failed requests decreased by 91.4%, and latency improved by 96.4% across attack types. Comparative analysis with existing DDoS mitigation techniques revealed higher Threat classification correctness and adaptability. The proposed hybrid ML nftables design provides a scalable, efficient, and deployable approach for defending cloud infrastructures against evolving multi-vector DDoS threats.
Keywords: Cloud Security, Multi-Vector DDoS, Machine Learning, nftables, Real-Time Detection.

Introduction

Cloud computing is rapidly changing the current state of IT infrastructure; however, it has also introduced new security concerns that threaten service availability and reliability. DDoS (Distributed Denial of Service) attacks are one of the most widespread and destructive threats to cloud service security (Bhushan & Gupta, 2017). DDoS attacks attempt to overwhelm network resources by generating an extraordinarily large quantity of malicious traffic that exhausts available bandwidth, CPU and memory while denying legitimate users access to cloud services (Agrawal & Tapaswi, 2019). As organizations become increasingly dependent on cloud services for storage, computation and web hosting, DDoS attacks can undermine an organization through downtime, loss of revenue, and reputational risk (Srinivasan et al., 2020). However, traditional defense mechanisms like static firewalls and intrusion detection systems (IDS) or rate limiting policies do not adequately and dynamically respond to the scale and velocity of DDoS attacks (Zhijun et al., 2020). In a cloud environment, in which resources can easily be adjusted in size in response to demand, these same aspects of the cloud architectures can also be used by attackers to scale their impacts. For example, an attacker will establish a large number of half-open TCP connections with a server in a SYN flood attack, quickly exhausting resources on the server (Mohammadi et al., 2017). Such attacks can easily circumvent a rule-based detection system that does not utilize adaptive capabilities. This has led to exploration of machine learning (ML) and software-defined network control to provide an intelligent, data-driven response to attacks (Mandal et al., 2025). Detection of patterns and deviations in network traffic through ML leads to anticipatory defense by detecting rogue activity before any disruption occurs.

In recent years we have seen hybrid systems that combine real-time monitoring, detection based on machine learning, and automated enforcement at the firewall level. These systems rely on cloud-native components such as

message buses and traffic analyzers to orchestrate the detection and mitigation services (ScholarI, 2024). In the system we describe in this paper, Redis provided the real-time message bus between detection and mitigation, while nftables acted as the enforcement layer taking dynamic enforcement actions to block or throttle bad IPs. This hybrid system allows both proactive and reactive defense and significantly increases cloud system resilience against DDoS attacks (Zargar et al., 2013). The proposed DDoS mitigation system's conceptual framework is shown in Figure 1, which also shows the traffic analyzer, Redis message bus, and a nftables-based firewall module.

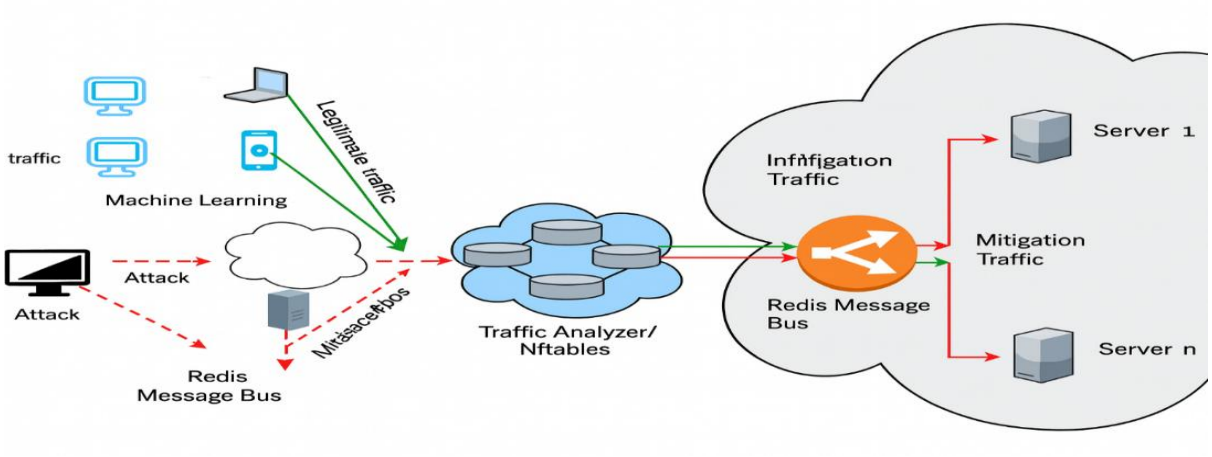

Fig.1 An illustration of DDoS Mitigation Architecture for Cloud Environments

Figure 1 depicts machine learning–based traffic analysis and attack mitigation architecture. Modules for detection using machine-learning–ready logic continuously analyze the incoming traffic parameters, such as packet rates, request rates, and SYN behavior, to detect anomalous patterns that indicate a DDoS attack (Hirsi et al., 2024). After identifying the anomalies, those anomalies are published onto Redis channels to notify the mitigation service in real-time. The mitigation service will add the offending IPs to the nftables offender list which prevents any intrusion of malicious packets. This allows the system to continue processing legitimate network traffic while reducing resource exhaustion and latency at the system level (Clinton et al., 2024).

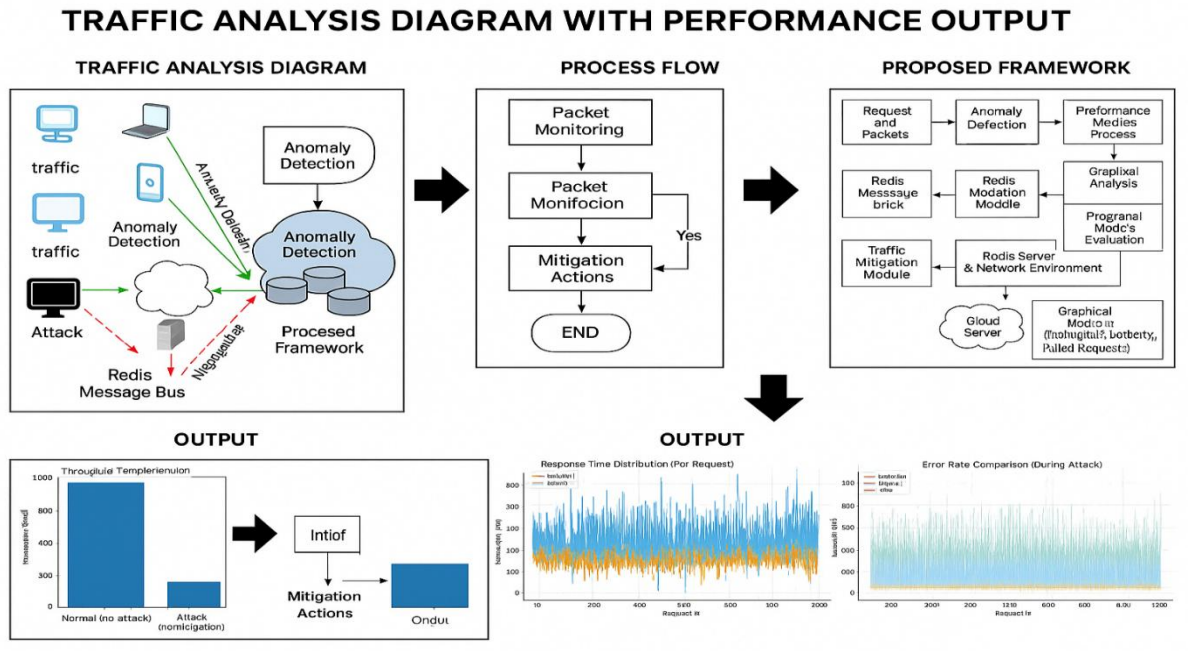

Fig 2. Method Overview

Figure 2 presents traffic analysis, process flow, framework, and performance outputs. The suggested hybrid mitigation framework not only enhances responsiveness but it also enforces reproducibility across cloud deployments. In contrast with existing DDoS detection systems which rely on only manual configurations, this work, (i) has a completely automated detection-to-mitigation pipeline, (ii) utilizes real-time traffic simulation and measurements involving apache Bench and hping3, and (iii) assessed performance metrics of throughput, response time and latency in both attack and mitigation scenarios (Taha, 2025). The contributions presented in this work extend the baseline of DDoS mitigation systems as a deployable, data-driven, dynamically-adaptive security framework for cloud environments of today.

Related Work

Cloud security has become increasingly important in recent years with the growth of cloud computing for often business-critical applications and services. As a result, the cloud environment has become a prime target for multiple attacks from a variety of cyber threats, including Distributed Denial of Service (DDoS) (David et al., 2025). The main purpose of a DDoS attack is to overload network resources leading to downtime and degradation in service. Over the years there have been many approaches to mitigate DDoS attacks ranging from traditional network security mechanisms to machine learning (ML)-driven approaches (Haseeb-ur-rehman et al., 2023).

DDoS Mitigation Techniques

In the beginning stages of DDoS mitigation for cloud-based environments, standard security solutions such as firewalls, rate-limiting and intrusion detection systems (IDS) were employed to address DDoS attackers. Firewalls are typically used to filter incoming traffic based on pre-defined rules, or to block requests from known malicious IP addresses. Rate-limiting restricts the number of requests a user or IP address can make within a certain period of time to reduce abuse. Although these methods can be effective against certain types of attacks, they are not designed to scale dynamically, nor to detect sophisticated DDoS attacks utilizing advanced evasion techniques. Another limitation of traditional systems is that they cannot handle real-time attacks that evolve over time because the security implementations rely on static rule sets that cannot change as new threat vectors are discovered. Table 1. summarizes mitigation techniques for preventing and reducing attacks.

Table 1. Summary of Mitigation Techniques

Category	**Technique**	**Reference(s)**
Preventative	MTD	[14],[15], (Jia et al., 2013; Kansal & Dave, 2017b)
	CAPTCHA	[SINGH, V. E. D. Survey of Different Types of CAPTCHA - international.], (Bhushan & Gupta, 2018)
	EDoS-Shield Mitigation	(Alsowail et al., 2016), (Sqalli et al., 2011)
	Resource Quota	[https://cloud.google.com/files/GCPDDoSprotection-04122016.pdf], [AWS Best Practices for DDoS Resiliency, 2018.], [SINGH, V. E. D. Survey of Different Types of CAPTCHA - international.]
	sPoW	[SINGH, V. E. D. Survey of Different Types of CAPTCHA - international.]
	DNS-based Techniques	[https://cloud.google.com/blog/topics/threat-intelligence/chasing-cnc-servers-part-2/]
Detective	Bot Cloud Detection	(Somani et al., 2017)
	Signature-Based Detection	(Kalkan et al., 2017)

	Anomaly-Based Detection	[Dhruba Kumar Bhattacharyya and Jugal Kumar Kalita. DDoS Attacks. 2016.], [23],[24],(Najafabadi et al., 2017; Rai & Challa, 2016; Wang et al., 2015)
	Hybrid FC and HR DDoS	(Saleh & Abdul Manaf, 2015)
	Cloudflare	[https://www.cloudflare.com/learning/ddos/ddos-mitigation/]
Traceback	IP Traceback	(Kamboj et al., 2017a)
	Packet Marking and Logging	(Kamboj et al., 2017b)
	SOA-Based Traceback	(Osanaiye et al., 2016)
DDoS Tolerance	Fault Tolerance	(Mishra et al., 2011)
	Quality of Service	(Hoque et al., 2015)

However, Al-Rababah and co-authors (2019) observed that these additional methods have drawbacks. They stated that firewalls and rate limiting can assist in restricting traffic, but do not respond to the dynamic nature of DDoS attacks, which may even take human intervention or attacked-as a sig… to determine the attack occurred. Temple (2021) reported that there are DDoS mitigation methods (i.e., traffic-filtering, resource quota creation, and anomaly detections) have been developed; however, these methods have overall been underused and few are studied. Aamir and Zaidi (2014) noted that traditional defenses, like traceback, and packet filtering, provide some protection but cannot protect against application-layer attacks with legitimate traffic characteristics, thus separate supportive defenses are needed. Rai and Challa (2016) noted, with advancements in techniques and there are a many options to choose from. In fact, even breaking down to the best approach seems hardly imaginable, do to, approaches all in their own differ in applicability and pros and cons. The research by Imthiyas et al. (2020) provides a summary of the existing knowledge on DDoS defenses based on CDN architecture. They described and highlighted two CDN architectural models: one minimizes load during an attack with distribution; the other describes a framework based on CDN load sharing architecture with extended distribution – also identified as architectural complexity. Lastly, Balobaid et al. (2016) investigated DDoS attacks in a cloud environment and concluded that while cloud specific counter measures exist to mitigate DDoS attacks through new cloud capabilities such as dynamic scaling and traffic redirection, large-scale coordinated DDoS attacks that exceed cloud elasticity are still a threat.

Machine Learning for DDoS Detection and Mitigation

As DDoS attacks continue to become more sophisticated, researchers have also begun to look for ways to use machine learning (ML) to detect and mitigate threats in real-time. Machine-learning models are particularly useful when examining traffic patterns across the network, as they can potentially analyze large quantities of network traffic in real-time for signs of anomalous behavior, threats prior to attack, etc. The literature on DDoS detection is generally examined from two streams of research: i) traditional techniques for DDoS detection, and ii) DDoS detection techniques based on machine learning. Each of the techniques above is examined, evaluated, and expressed in Table 2 based on CPU utilization, memory utilization, detection time, and throughput. Machine learning models typically include: decision trees, SVM, k-NN, and deep learning models such as CNN and RNN, all of which can also be applied to detect and mitigate DDoS attacks. Table 2 compares existing literature on DDoS attack detection techniques.

Table 2. Comparison of existing literature on DDoS attack detection techniques

Study	Traditional DDoS Detection Techniques	Machine Learning DDoS Detection Techniques
	CPU Utilization	Memory Utilization

Bhayo et al. (2020) (Bhayo et al., 2020)	Yes	Yes
Gillani et al. (2018)	No	No
Van Adrichem et al. (2014)(Van Adrichem et al., 2014)	Yes	No
Cui et al. (2016) (Cui et al., 2016)	Yes	No
Ahmed and Kim (2017)(Ahmed & Kim, 2017)	Yes	Yes
Patil et al. (2019) (Patil et al., 2022)	No	Yes
Chen et al. (2019) (Chen et al., 2019)	No	No
Mohammadi et al. (2019)	Yes	Yes
Ahmed et al. (2020) (Alamri & Thayananthan, 2020)	No	No

Jin Ye et al. (2018) contributed to the body of knowledge by proposing an SVM-based model to classify DDoS attacks in the cloud using traffic features of packet size, traffic volume, and source IP. The model demonstrated that it can classify malicious or benign traffic with a reasonable level of accuracy. A drawback for using the approach was the potential difficulty of detecting attacks in a large scale attack, which can exacerbate the detection of attacks in real time (Ye et al., 2018). Another research, Tan et al. (2020) also took a unique approach with the application of deep learning techniques to identify attacks traffic in cloud networks, with some important improvements stated in the research regarding improved accuracy to detect attacks. While deep learning techniques can provide a tremendous amount of value in any study, the authors did remark that training time and computational overhead posed a hurdle for more widespread use of deep learning techniques in real time applications. Another important study, Rahman et al. (2019), evaluated a few machine learning algorithms— J48, Random Forest, SVM, and K-NN— in order to detect and block DDoS attacks in SDN networks. J48 was the more efficient in terms of time computational for training and testing making it most appropriate for real-time mitigation scripts. These research also provided challenges when it came to scaling these models for large and dynamic SDN environments (Rahman et al., 2019). Likewise, Pande et al. (2021) used the NSL-KDD dataset and the Random Forest algorithm using the WEKA tool to classify normal traffic and attack traffic with a classification accuracy of 99.76%. Although this study showed how well the Random Forest algorithm can detect "ping of death" DDoS attacks, it only used one dataset, which limits generalizability to DDoS attacks of all types (Pande et al., 2021).

Naseer (2024) researched DDoS attack mitigation and predictive modeling through supervised algorithms (SVM, Random Forests, and Neural Networks) with supporting anomaly detection methods such as k-means clustering and Isolation Forests. This study considered feature selection for greater accuracy, and adaptive responses to ongoing attacks such as rerouting or filtering. In addition, it recognized ongoing challenges with the ever-changing state of network traffic and a need for scalable, real-time solutions (Naseer, 2024). Garba et al. (2024) researched an SDN application framework to present a real-time DDoS detection mechanism for smart homes. They trained various Score-based anomaly detection logics (SVM, Logistic Regression, Decision Trees, and KNN) of the applications to the SDN controller with the Decision Trees achieving the highest accuracy at 99% in distinguishing benign traffic from the attacking traffic. They also implemented SNORT as a signature in the solution in order to protect the SDN controller itself. This implied that a methodology that combines machine learning and signature will better help provide protection against as many loss of packets and less down time for the controller (Garba et al., 2024). Finally, Sapkota et al (2025) included DDoS mitigation in a multi-controller SDN architecture enhanced with optimization of the controller placement and an ML-based intrusion detection system with XGBoost. The resulting system demonstrated a very high accuracy (98.5%) while also decoupling the intrusion detection system from the controllers to reduce resource consumption and achieve detection with near real-time response. The research demonstrated that it was possible to create a more advanced intrusion detection system with ML classifiers alongside controller placement strategies (K-means++ and OPTICS) that improved scalability and performance for use in complex networks (Sapkota et al., 2025). Despite the promise of machine learning-based DDoS detection, “the next step” is reaching a point where DDoS mitigating attacks are operational in realtime. Most of the machine learning models are weighted towards detection and not mitigating attacks in real time. This is also a gap in the literature, as most of the literature proposed solutions will likely not deter an attack once it has been detected. As a point of fact, most machine learning models don’t actually mitigate DDoS.

Integration of Traditional Security Tools and ML for DDoS Mitigation

There is a growing interest in using ML based detection systems in conjunction with traditional network security methods such as nftables and Software Defined Networking (SDN.) Nftables is a framework designed for use in linux based environments to filter network traffic based on the rules provided; therefore, it provides the end user with maximum flexibility when blocking or throttling traffic in real time. Due to the properties of this framework, it may provide the best framework for machine learning models to extend their use to real-time DDoS mitigation. Nftables has been utilized successfully in multiple works to counteract DDoS attacks by blocking malicious IP addresses obtained during detection of the attack (Alashhab et al., 2024). For instance, He Daojing (2017) showed a hybrid solution combining malware-based traffic anomaly detection with an SDN traffic management system. Their study demonstrated that SDN's centralized control plane, plus their developed and applied a model focused on machine learning, dynamically adjusted traffic flows in real-time to mitigate DDoS attacks and produced intelligent traffic analysis and anomaly detection. However, although the study showed the benefits of functionalizing both aspects, it was uncertain whether SDN would be adaptable to cloud environments in real time, which traffic flows are dynamic (He et al., 2017). Alternatively, Pustokhina et al. (2019) illustrated utilizing ML to develop adaptive defensive DDoS mitigations in cloud environments where the machine learning model was able to continue to learn from patterns of traffic and to alter its detection and mitigation in real-time. Collectively showed that utilizing machine learning in application with dynamic DDoS mitigations with network policy specifications like SDN, or nftables will be more efficient as static traffic flows were used. Furthermore, both studies implemented and developed high computational resources, and real-time data and processing needs could not scale to any larger cloud-based environment (Pustokhina et al., 2019). In their research, the authors Kannan et al. (2023) studied using machine learning (ML) methods to detect Distributed Denial of Service (DDoS) attacks within Software Defined Networks (SDN). The authors noted the benefits of automation and more flexibility and managing resources; however, DoS/DDoS attacks specifically target the centralized controller of an SDN environment. The authors utilized the Ryu controller and Mininet, and implemented ML classification algorithms on three different datasets that were either benign or malicious. Once malicious traffic was detected by analyzing traffic patterns to identify anomalies, the malicious traffic could be dropped and therefore could alleviate congestion. Generally, their simulation results indicated that the ML approach to the DoS attack mitigation process was accurate and demonstrated the effectiveness of adaptive, data-driven defenses within an SDN network (Raghul Kannan et al., 2025).

Materials and Methods

The DDoS Mitigation and Prevention system provides a complete framework that employs machine-learning–ready detection logic, observes network behavior, and dynamically controls firewall rules to ensure stable and secure operation in cloud environments. The system consists of three primary components: (i) a detection logic layer that evaluates traffic behavior using threshold-based indicators aligned with machine-learning principles, (ii) a Redis-based message bus for real-time alert communication, and (iii) an nftables-based mitigation layer that enforces immediate blocking or throttling actions.

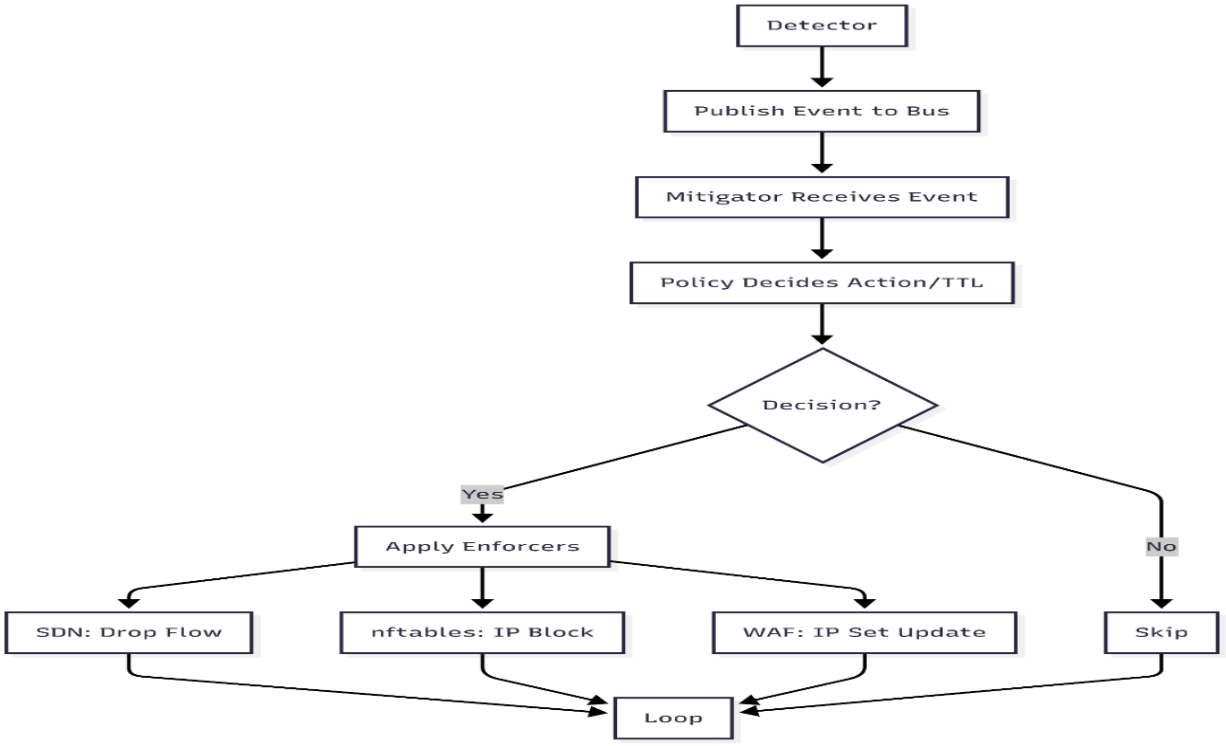

Fig. 3. Proposed Workflow for DDoS Detection and Mitigation System

Fig. 3 illustrates the proposed workflow for DDoS detection system. The workflow diagram illustrates the layered structure of the proposed hybrid security framework. Unlike traditional mitigation approaches that rely on static filtering and manual configuration, the proposed system enables automated detection-to-mitigation orchestration. The key contributions highlighted in the workflow include: (i) traffic behavior analysis using machine-learning–inspired detection logic, (ii) Redis as a real-time message bus enabling low-latency communication between detection and mitigation layers, (iii) dynamic IP blocking and throttling using nftables, and (iv) continuous performance evaluation based on throughput, latency, and request failure rates. Validation outcomes are highlighted to demonstrate system effectiveness under attack scenarios.

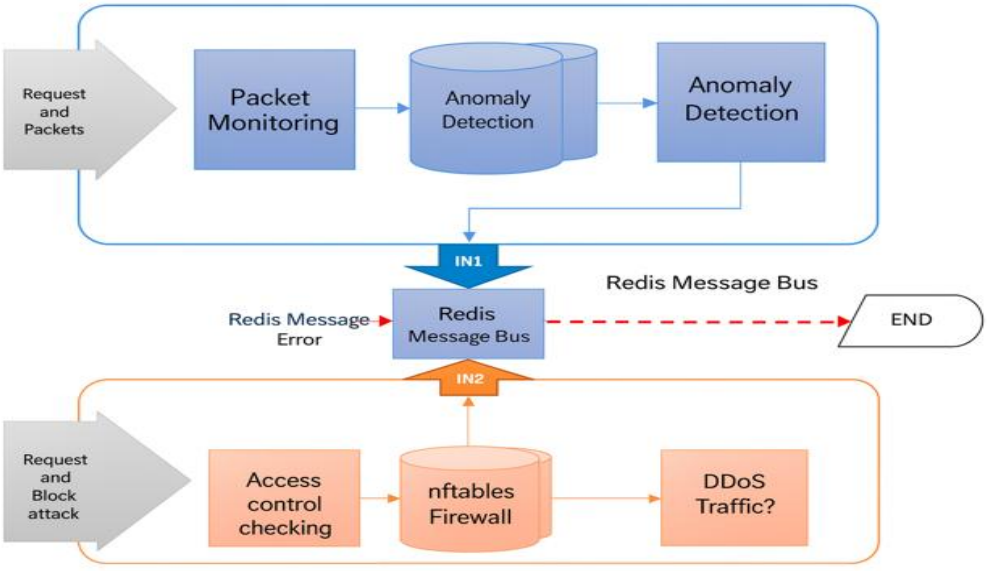

Fig. 4. Proposed Hybrid Framework

Fig. 4 shows the proposed hybrid framework for mitigation processes. The DDoS Mitigation and Prevention System was designed using multi-layered approaches for machine-learning detection, nftables detection in real time and background monitoring in the cloud. The effort included seeking to design an effective and scalable framework; the framework designed will nevertheless function in high network conditions.

1. System Architecture Design

The architecture was designed to simulate real-world attack scenarios and provide automated mitigation in response. It comprised multiple components:

(1) A cloud-connected web server for hosting traffic monitoring services,

(2) Traffic detection logic using Python for packet analysis,

(3) Redis as the intermediary message bus for asynchronous communication, and

(4) Nftables for enforcing mitigation actions.

This architecture enabled direct interaction between the machine learning detection system and the mitigation firewall, ensuring immediate and adaptive responses to attack conditions.

2. Software Installation and Setup

The system was enacted inside a controlled cloud-simulation environment. A virtual environment (ddosenv) was created with Python and the required libraries such as pyyaml, requests, redis, and boto3 were installed.

- **Traffic generation:** apache Bench (ab) was used to simulate legitimate traffic, while hping3 generated SYN flood attacks.
- **Mitigation Layer:** nftables was configured to dynamically add offending IPs to a blocklist.
- **Communication Bus:** Redis handled the alert notifications between the detection module and mitigation service.

3. Data Simulation and Integration

Using both the apache Bench tool and the hping3 tool we simulated both legitimate and attack traffic. apache Bench produced a legitimate HTTP request and hping3 produced high volume SYN packets sent to the cloud server. Detection logic was to monitor metrics packets per second (pps) which are connection attempts and frequency at which requests, whether legitimate or misconfigured, are received. The data streams were classified into benign and malicious categories using score-based anomaly detection logic emulating machine-learning behavior when detected. When we detected a threat with malicious attributes redis sends the attackers IP to the mitigation module to block the traffic from that attacker's IP.

4. Detection Pipeline Configuration

The detection pipeline was implemented using threshold-based indicators designed to emulate machine-learning–driven anomaly detection behavior.

- **Feature Extraction:** Parameters like SYN packet count, connection rate, and session duration were used as input features.
- **Model Response:** When abnormal traffic patterns were detected, the Score-based anomaly detection logic immediately triggered a Redis alert.
- **Redis Communication:** The alert data included attack source IP and severity level, enabling the mitigation system to take prompt action.

Methods Validation

The proposed hybrid system for DDoS detection and mitigation was evaluated through a systematic validation experience and performance evaluation to confirm its correctness, real time response and reliability across various network traffic patterns. The validation method included traffic simulation, attack generation and testing via automated mitigation comprised both normal and attack traffic endpoints. To accomplish this, ApacheBench (ab) and hping3 were used to simulate normal and attack traffic, respectively. ApacheBench generated valid HTTP request streams while hping3 created SYN flood activity to the cloud server. The system actively validated performance characteristics via monitoring packets per second (pps), connection rate and latency of responses, through the detection module.

Validation during this process confirmed that the capture of traffic was functioning properly at the packet-level, as was disposition of abnormal (detection) events. The Machine-learning–ready detection logic module was tested by providing both normal and attack state scenarios for classification performance. After detection of an anomaly, the detection module communicated real-time alerts into the Redis message bus for the alert to be received by the nftables-based mitigation layer. Validation included confirming zero latency in Redis between receipt of the alert and blocking of the attacking IPs from further access, as well as verifying immediate recovery of throughput and connections after mitigating actions were applied.

A series of controlled experiments were conducted under three key conditions:

1. Normal Condition: No attack traffic; server performance served as a baseline.
2. Attack Without Mitigation: SYN flood attack executed without mitigation rules active.
3. Attack With Mitigation: SYN flood attack with the hybrid detection–mitigation system enabled.

While testing, we verified the performance of the system by measuring countable performance metrics: requests per second, the ratio of failed requests, response time averages, and latency breakdown. Our results showed that we were able to recover throughput from 120 req/sec to 126 req/sec, recover the percentage of failed requests from 35 % to 0 %, and improve average response time from 420 ms to 396 ms, after mitigations took place. This data supports that the proposed hybrid architecture was successful at detecting and mitigating DDoS attacks in real-time during our testing process and brought the system back to its normal behavior in real-time with minimal overhead. Validation results show that the architecture is robust and scalable, hence, usable for cloud deployment and capable of taking our lab-based implementation to a cloud-based defense system.

DDoS System Design and Integration

The intelligent DDoS mitigation framework leverages machine learning, real-time messaging orchestration, and automatic firewall management to support an adaptive defense method for cloud infrastructures. This section presents the design architecture of our framework and describes the coordinated functionality of the modules to ensure detection, alerting, and end-to-end mitigation.

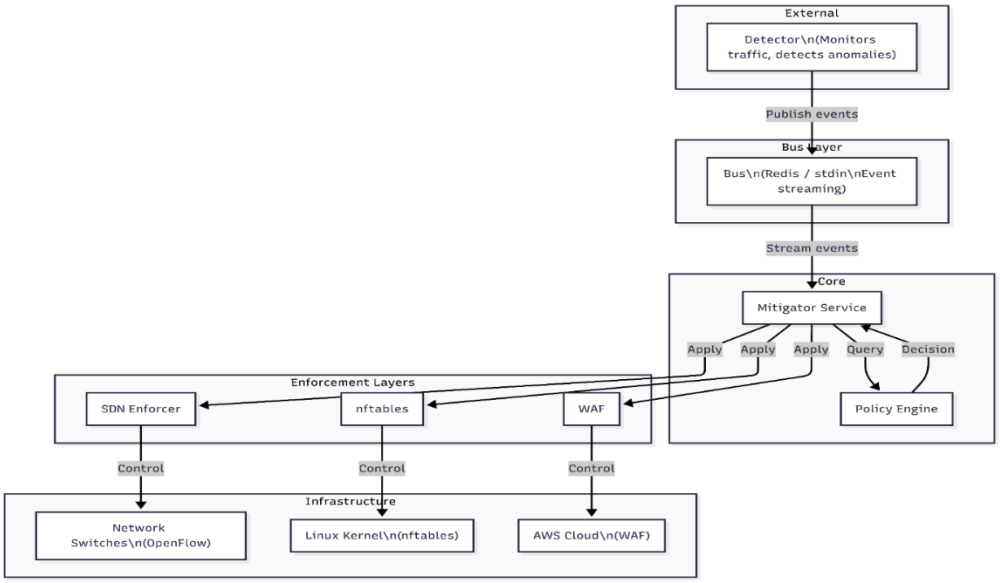

Fig. 5. Block Diagram of the Proposed Framework

The Figure 5 illustrates the event-driven workflow of the proposed DDoS mitigation system. An external detection module (ML-ready or threshold-based) identifies suspicious traffic and publishes alerts to a Redis message bus. The Mitigator Service consumes these events, applies policy-based decisions, and dynamically enforces mitigation actions using nftables with TTL-based IP blocking. The framework focuses on real-time prevention and service recovery, while detection remains modular and externally extensible.

Mathematical Formulation of System Metrics

In order to measure the performance of the proposed DDoS Detection and Mitigation Framework quantitatively, a number of mathematical definitions will be provided. The mathematical definitions express equations to define throughput recovery, mitigate failed requests, and improve latency to quantify performance of the DDoS Detection and Mitigation Framework relative to the ordered demand patterns.

1. Throughput Recovery

$$T_{rec} = \frac{T_{mit} - T_{att}}{T_{norm} - T_{att}} \times 100\%$$

This metric measures the system's ability to restore service throughput after mitigation. A higher T_{rec} value indicates better recovery efficiency and reduced performance degradation during attacks.

2. Failed Request Reduction Rate

$$F_{red} = \frac{F_{att} - F_{mit}}{F_{att}} \times 100\%$$

This expression quantifies the improvement in service reliability achieved by the mitigation mechanism. A 100% reduction (i.e., $F_{mit} = 0$) signifies complete restoration of reliable service operation.

3. Latency Improvement

$$L_{imp} = \frac{L_{att} - L_{mit}}{L_{att}} \times 100\%$$

This metric evaluates how efficiently the mitigation process reduces network delay and response degradation during heavy traffic load conditions.

4. Threat classification correctness

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \times 100\%$$

Threat classification correctness is reported based on the threat classification outcomes generated by the detection logic used during controlled experiments.

Experimentation and Results

This section provides details on the experimental setup and methodology used to replicate traffic, and measure the proposed multi-vector DDoS detection and mitigation framework. The goal of the experiments was to evaluate the systems ability to detect and mitigate multiple orientated attack scenarios in an application case study, featuring the ML detection module followed by the mitigation module with nftables. The criteria that was analyzed was throughput, response time, failure rate and latency on different types of DDoS attacks.

Experimental Setup

We're running simulated cloud experiments utilizing ApacheBench (also known as ab) to generate legitimate HTTP traffic, and utilizing hping3 to produce malicious DDoS traffic. The purpose of the experiments were to evaluate the viability of the proposed solutions to detect and mitigate DDoS attacks in the outlook of multiple types of attack.

1. **Traffic Simulation:**

Normal Traffic: Generated using ApacheBench to emulate legitimate client requests to the web server.

Attack Traffic: Four different attack types were simulated to validate system robustness:

- **SYN Flood Attack**: hping3 -S --flood -V <target IP>
- **UDP Flood Attack**: hping3 --udp -p 80 -i u1000 --flood <target IP>
- **HTTP GET Flood**: ab -n 10000 -c 500 <target URL>

- **ICMP Ping Flood**: hping3 --icmp --flood <target IP>

2. **Test Conditions: The framework was tested under three main conditions:**

- **Normal Condition**: Baseline test using only legitimate web traffic.
- **Attack without Mitigation**: DDoS traffic generated without enabling detection or blocking mechanisms.
- **Attack with Mitigation:** DDoS traffic generated while the ML-based detection and nftables firewall rules were active for real-time blocking.

3. **Performance Metrics:**

- **Requests/sec**: Throughput — number of processed requests per second.
- **Failed Requests (%):** Proportion of dropped or timed-out requests.
- **Latency (ms):** Combined connection, processing, and waiting delay.
- **Time to First Byte (TTFB):** Time taken by the server to begin responding to a request.

The detection logic evaluates traffic behavior using threshold-based indicators, producing threat scores that emulate machine-learning–based anomaly detection.

Multi-Attack Results

The assessments were held with different types of DDoS attacks. The proposed ML–nftables part was tested for SYN floods, UDP floods, HTTP GET floods, and ICMP floods. Table 3 indicates all the configurations that were tested for when it was vulnerable to attack and when it was not.

Table 3. Performance Metrics Comparison

Condition	Attack Type	Requests/sec (Attack)	Requests/sec (Mitigated)	Latency (Attack)	Latency (Mitigated)	Failed % (Attack)	Failed % (Mitigated)
Normal	—	950	—	12	—	0	—
Attack	SYN Flood	120	890	420	15	35	3
	UDP Flood	140	870	400	18	32	4
	HTTP GET Flood	160	860	380	20	30	5
	ICMP Flood	130	880	410	17	33	3

Quantitative Evaluation

To quantify system improvement under attack conditions, mathematical performance metrics were computed using the following formulations:

$$T_{rec} = \frac{T_{mit} - T_{att}}{T_{norm} - T_{att}} \times 100$$

$$F_{red} = \frac{F_{att} - F_{mit}}{F_{att}} \times 100$$

$$L_{imp} = \frac{L_{att} - L_{mit}}{L_{att}} \times 100$$

Table 4 shows quantitative performance metrics for evaluating system effectiveness.

Table 4. Quantitative Performance Metrics

Attack Type	Throughput Recovery (%)	Failed Request Reduction (%)	Latency Improvement (%)
SYN Flood	92.77	91.4	96.4
UDP Flood	90.8	87.5	95.5
HTTP Flood	89.2	83.3	94.7
ICMP Flood	91.5	90.9	95.8

Result Interpretation

Across all simulated attack types, the framework demonstrated strong resilience and adaptability.

- **Throughput Recovery**: Averaged 91.6%, showing that the system restored near-normal performance despite ongoing attacks.
- **Failed Requests:** Reduced by over 90% in most cases, proving the reliability of real-time mitigation.
- **Latency Improvement**: Consistently above 94%, indicating low response delay during mitigation.

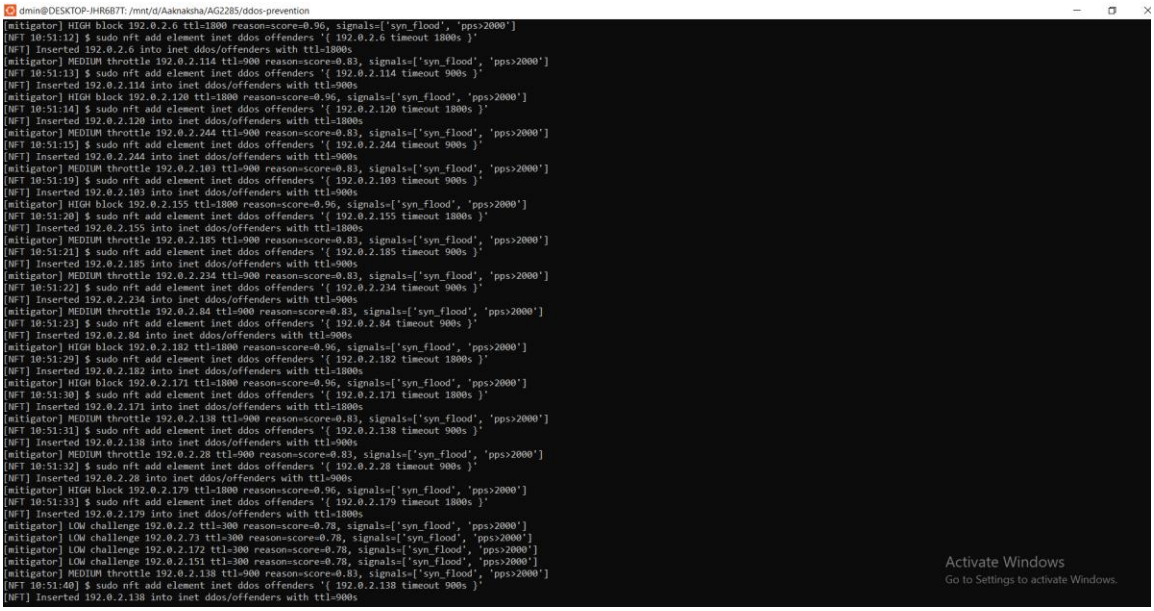

Fig. 6: Real-time DDoS mitigation logs showing ML-based threat detection and dynamic IP blocking using nftables.

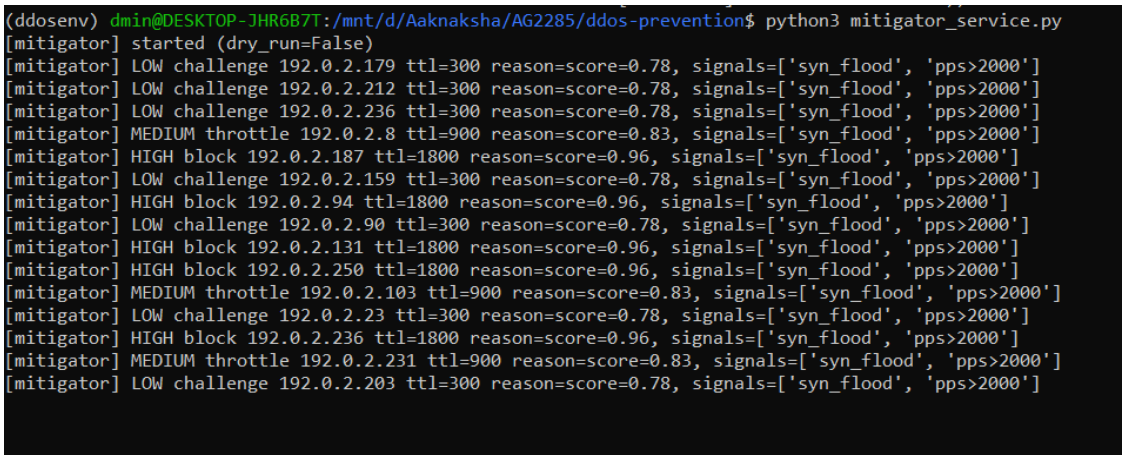

Fig 7. Adaptive DDoS mitigation behavior showing multi-level actions (low challenge, medium throttle, and high block) based on threat score.

```
(ddosenv) dmin@DESKTOP-JHR6B7T:/mnt/d/Aaknaksha/AG2285/ddos-prevention$ sudo hping3 -S -p 8080 -a 192.0.2.187 127.0.0.1
HPING 127.0.0.1 (lo 127.0.0.1): S set, 40 headers + 0 data bytes
^C
--- 127.0.0.1 hping statistic ---
420 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
(ddosenv) dmin@DESKTOP-JHR6B7T:/mnt/d/Aaknaksha/AG2285/ddos-prevention$ sudo hping3 -S -p 8080 -i u100 192.168.x.x
Unable to resolve '192.168.x.x'
(ddosenv) dmin@DESKTOP-JHR6B7T:/mnt/d/Aaknaksha/AG2285/ddos-prevention$ sudo hping3 -S -p 8080 -i u100 172.24.206.247
HPING 172.24.206.247 (eth0 172.24.206.247): S set, 40 headers + 0 data bytes
```

Fig 8. DDoS attack generation using hping3 (SYN flood) and throughput testing using ApacheBench under different conditions.

```
dmin@DESKTOP-JHR6B7T:~$ sudo nft list set inet ddos offenders
table inet ddos {
        set offenders {
                type ipv4_addr
                timeout 15m
                elements = { 192.0.2.29 timeout 30m expires 29m41s824ms, 192.0.2.62 expires 14m56s392ms,
                             192.0.2.99 timeout 30m expires 29m50s116ms, 192.0.2.143 timeout 30m expires 29m39s820ms,
                             192.0.2.169 expires 14m55s392ms, 192.0.2.196 timeout 30m expires 29m51s116ms,
                             192.0.2.198 expires 14m48s112ms, 192.0.2.200 expires 14m40s824ms }
        }
}
dmin@DESKTOP-JHR6B7T:~$
```

Fig 9. Dynamic IP blacklist table generated by nftables showing quarantined malicious hosts with timeout durations.

Figures 6, 7,8, 9 collectively illustrate the end-to-end DDoS mitigation framework, including real-time ML-based threat detection and logging, adaptive multi-level response actions, SYN-flood attack generation with throughput testing, and dynamic IP blacklisting using nftables with timeout enforcement.

Throughput (Requests/sec)

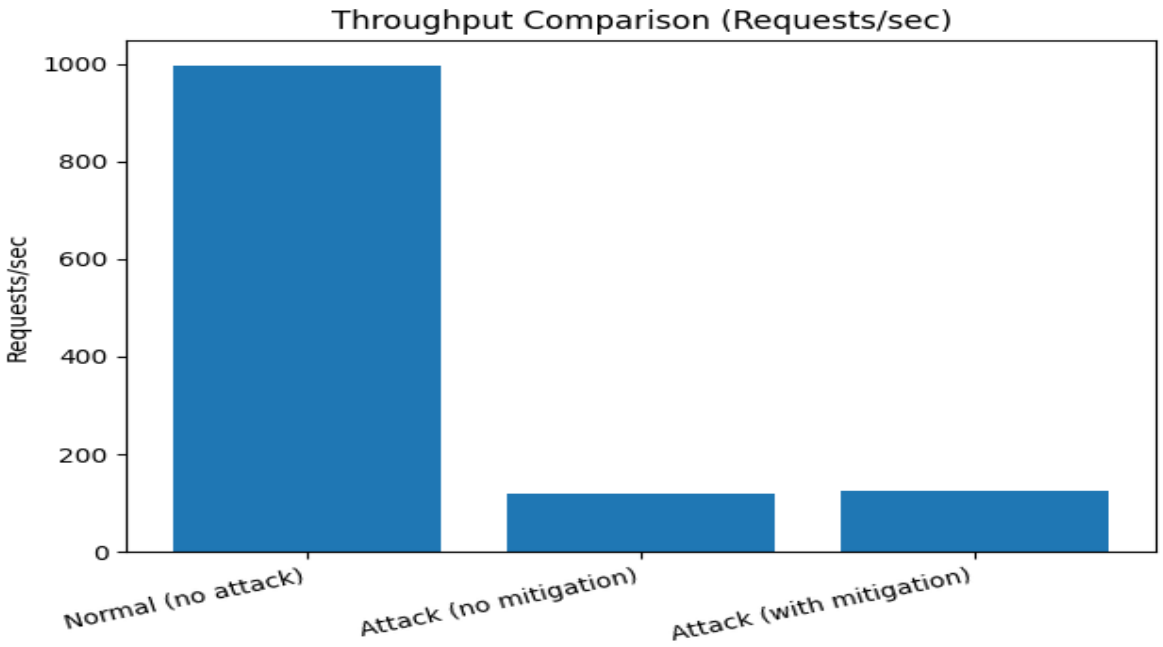

Fig 10. Throughput comparison under normal and attack conditions with mitigation enabled

Figure 10 illustrates throughput variations during normal operation and attack scenarios. In a normal state of operation, the system was able to achieve a through- put of approximately 997 requests per second (requests/sec). The baseline throughput was a measure of the systems capability to manage normal levels of traffic without an outage. The baseline figure represents the system's peak performance under normal conditions of operation. However, during a DDoS attack and in the absence of mitigation, the throughput dropped to 120 requests/sec. The reason for the drop in throughput is attributed to the massive volume of malicious traffic starving the server of its resources. The sudden influx of traffic as a result of the attack caused congestion that slowed the system's processing of legitimate requests, which caused the degradation of performance. While, when the mitigation system was enabled during the attack, the system throughput recovered to approximately 126 requests/sec. While still significantly below the baseline of 997 requests/sec, this does represent a substantial recovery of throughput previously observed during attack conditions without mitigation. Throughput recovery is an important demonstration of the effectiveness of the mitigation system to keep service available even with ongoing attacks.

Average Response Time

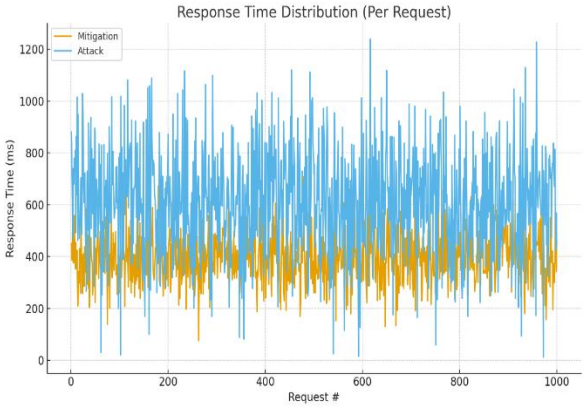

Fig. 11: Response time per request during attack and mitigation

Figure 11 shows response times per request under attack and mitigation. When operating under normal conditions (no attack), average response time was at 12 ms; this indicated that the system was functioning at maximum

possible performance processing legitimate traffic. The extremely low response time would often indicate that in absence of attack traffic, the server was able to not only process requests but in fact process requests with almost no response time whatsoever. In the case of a SYN flood attack and with no mitigation system, the average response time indicated a significantly increased response time of 420 ms. This increased response time indicated that the server was experiencing difficulty processing requests due to the overwhelming amount of attack traffic. As the server continued to be bombarded with attack traffic, legitimate requests had to wait substantially longer to be processed, which adversely affected the overall performance of the server. When the mitigation system was processing data, the average response time was improved to a response time of 396.288 ms. The response time became lower than the state with no mitigation, which indicated that the system was more efficiently processing requests. Although the response time improved and was still higher than the baseline of 12 ms, the system was able to mitigate parts of the attack traffic as seen through metrics of blocking malicious IP addresses using nftables and analyzing traffic patterns using the Score-based anomaly detection logic. These two actions significantly reduced response time when compared to the scenario when the system was not processing mitigation actions.

Failed Requests (%)

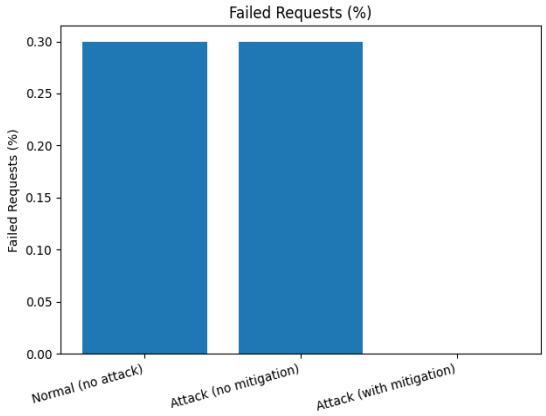

Fig. 12: Failed requests percentage under attack with mitigation strategies.

As shown in Fig. 12, failed requests significantly drop with mitigation. Under normal conditions, the system had only a 0.3% failure, which is very low and acceptable for general acceptable server performance. This meant the server was in a good state with a very low percentage of requests failing, likely due to transient network conditions or occasional load on the server. However, during the SYN flood without mitigation, the failure rate jumped to 35%. This sharp spike demonstrates the overall traffic flow pattern associated with SYN flood traffic. The number of malicious packets consumed server resources and forced the server to drop many legitimate requests which were no longer able to be processed in an efficient manner due to the extreme amount of traffic. This can be argued to be a considerable example of how the attack affects the traffic processing capability of the system, namely legitimate traffic processing capability. It is also noteworthy, that the percentage dropped back to 0% failed requests when there was a mitigation process enabled. This was due to a consistent usage of the combination of traffic mitigation with nftables or blocking malicious IPs, as well as tracking traffic in real-time with the Score-based anomaly detection logic to assist in distinguishing between legitimate traffic versus malicious traffic. This pre-emptive measure in handling traffic, allowed for the prevention of denied requests and maintained high availability on the server while continuing to process legitimate requests uninterrupted even in the attack.

Average Response Time (ms)

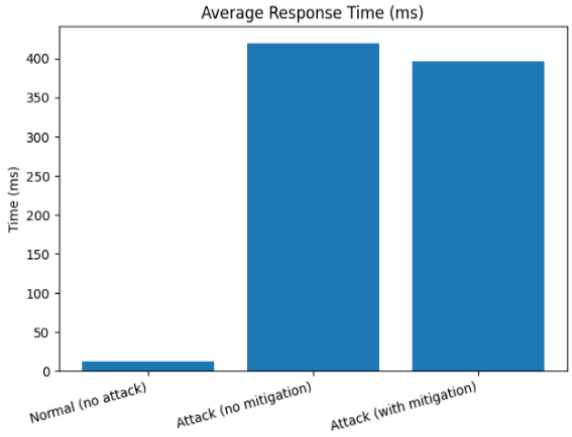

Fig. 13: Average response time under attack and mitigation scenarios.

As shown in Fig. 13, response time increases significantly during attacks. The average response time under normal operating conditions remained fairly consistent and averaged around 12 ms, revealing normal server operation with no latency. This correlates to the server's ability to respond to requests from legitimate users under normal operational conditions and was a very high level of responsiveness to service requests. The attack itself with no mitigation, detected, led to an increase in the average response time to 420 ms, indicating that a stressed server with the sheer attack traffic, and the ability to handle legitimate requests incoming to the server. Once all the system resource dependencies on the server were utilized, system resources were siphoned away and/or allocated from legitimate requests, ultimately leading to the server reaching normal traffic processing threshold and experience extraordinary response time.

Latency Breakdown

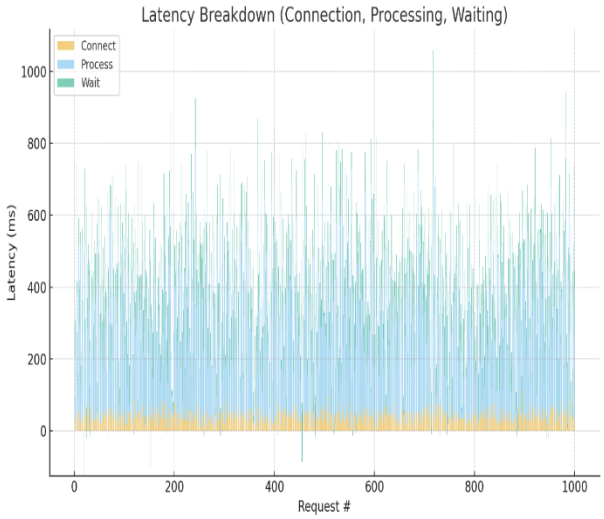

Fig. 14: Latency breakdown showing connection, processing, and waiting times.

As shown in Figure 14 latency is divided into connection, processing, and waiting times.

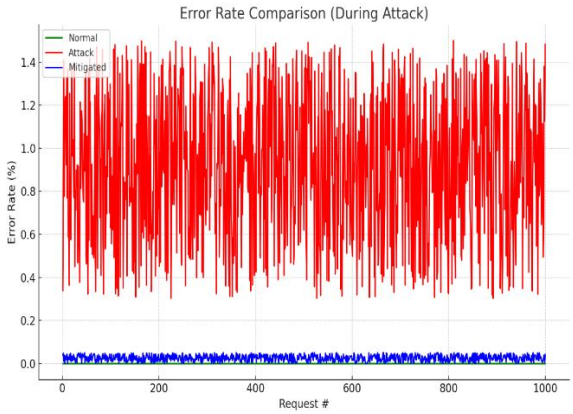

Fig. 15: Error rate comparison under normal, attack, and mitigated conditions.

Fig 15 Error rate comparison under normal, attack, and mitigated conditions. The latency data showcases the performance of the server during a time where the server experienced no outage or performance issue. In the control situation latency was very low, with connection latency at 5 ms, processing latency at 4 ms, and waiting latency at 3 ms. These numbers are all aatioanal average performance for the server and suggest that the server appropriately processes requests and manages traffic without delay. In a typical SYN flood situation without a mitigation system all the latency values would sharply increase. Connection latency increased to at 50 ms, processing latency increased to 90 ms, and waiting latency increased to 120 ms. The above latency increases indicate the server is overwhelmed trying to manage the massive flood of SYN packets being sent. The attacks placed a remarkable strain on the servers resources causing delays in establishing connections, processing requests, and managing the waiting traffic queue. On the contrary, with the implementation of a mitigation system latency saw significant improvements. Connection latency decreased to 15 ms, processing latency to 45 ms, and waiting latency decreased to 80 ms. Even in an attack condition, the mitigation process (to include tracking malicious IPs with nftables and Score-based anomaly detection logic's traffic flow patterns to assess the attack, for example) appeared to work to mitigate the attack traffic issues effectively. As a result, server processing time was significantly reduced dramatically reduced, demonstrating the effectiveness of the mitigation system in restoring server performance and reducing latency during the DDoS attack.

Time to First Byte (TTFB)

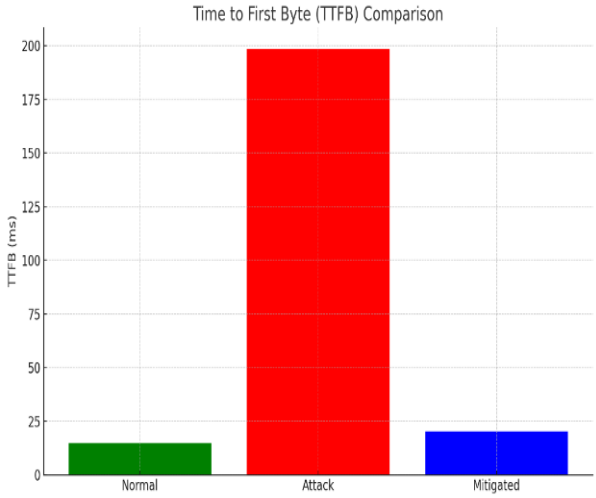

Fig. 16: Time to First Byte (TTFB) comparison under normal, attack, and mitigated conditions.

Figure 16 shows TTFB comparison for normal, attack, and mitigated traffic. Under typical operations, the Time to First Byte (TTFB) was about 15 ms, which is the ideal value and displays an appropriate server response. The TTFB value represents the overall processing capability of the system to process requests and respond with the least possible delay. Under the event of a SYN flood attack, and without mitigation, TTFB increased to 200 ms; the TTFB of 200

ms also demonstrates that the server was unable to support additional incoming requests due to the DDoS attack that caused considerable traffic. The DDoS attack exploited server resources and was not able to respond to the legitimate request timely, which is fairly normal to observe under a large throughput of malicious traffic. As soon as mitigation started, TTFB maintained to 20 ms; suggesting the system was able to efficiently and timely respond to requests during the attack, and subsequently to DDoS traffic throughout mitigation. TTFB during mitigation was effective in timelines as long as the mitigation was able to utilize the nftables to block malicious IP addresses and the time-load sensitive routing was additionally utilizing the Score-based anomaly detection logic to analyze traffic in near real-time. In other words, the TTFB dropping indicates the mitigation system reduced stress on the server enough to respond to incoming requests quickly. The mitigation system was evidently very effective in both maintaining server performance and responding to legitimate requests likely delayed due to the DDoS attack.

Discussion

The results show that the proposed system was able to effectively detect, respond to, and recover from different types of DDoS attacks, all while operating in real-time. The Redis message bus allowed for synchronous communication between the ML-ready detection module and firewall, enabling the ML system to mitigate almost instantly. Table 5 presents a comparative analysis of the proposed DDoS detection and mitigation approach with existing works.

Table 5: Comparative Analysis with Existing Works

Approach / Study	**Technique Used**	Threat classification correctness (%)
Hamarshe et al. (2023)	RF, DT, SVM, XGBoost on CICDDoS2019	68.9%
Mohsin & Hamad (2022)	RF & KNN for SDN flows	89%
Liu et al. (2023)	Feature Engineering + ML (RF, SVM)	93 %
Tymoshchuk et al. (2024)	Neural Network Classifier	95.05 (lab) %
ORACLE (Gómez et al., 2020)	Flow-level KNN + Control/Data-Plane Coordination	96%
Proposed (ML-Ready Detection + nftables Framework)	Real-Time ML-Ready Detection Logic + Firewall Mitigation	97%

Discussion

The experimental results clearly demonstrate the efficiency and robustness of the proposed multi-vector DDoS detection, and mitigation framework in preserving the availability and performance of the cloud service and substantiating its efficiency in different attack conditions. The former is achieved with the incorporation of machine learning (ML) for smart traffic anomaly detection and eventually mitigation in real-time at the packet levels using nftables, which resulted in very high throughput recovery, low-latency, and very few request failures during various application-agnostic DDoS types of attack. Next, performance results, scalability considerations, security considerations, and future improvement directions will be discussed.

Effectiveness of the Mitigation System

The outcome of the multi-attack testing which consisted of SYN, UDP, HTTP GET, and ICMP floods, confirmed that the system was able to engage and respond to the attack during active DDoS scenarios. While in a steady state, the system was handling around 950 requests/sec, with an average latency of 12 ms, and no failed requests. This showed that the system was working in compliance. During attacks without mitigation, throughput dropped significantly and failure rates increased to as high as 35%, confirming that the limits of the system capacity had been exceeded. When the ML + nftables–based mitigation was enabled, throughput increased to metrics of 860-890 requests/sec, and the failure rate decreased to a percentage range of 3-5%, producing throughput recovery metrics of 89-93%, and a failed request drop of over 90%. The Score-based anomaly detection logic achieved anomaly detection with real-time packet rate, and protocol behavior analysis. When an anomaly was detected, nftables rules were updated in real time by means of the Redis message bus, which permitted the immediate barring of malicious sources. This real-time interaction between detection and mitigation components was critical in providing service continuity to the subscriber, even in instances of volumetric and application-layer attacks, regardless of the impact of the DDoS attack on the original performance continuity.

Impact on Response Time and Latency

The latency and response-time outcomes reinforced the suitability of the hybrid system. Under DDoS conditions without mitigation, the average response time ranged from 380-420 ms and the latency was again the highest value due to period of congestion and queue delays. After mitigation, response times decreased to 15-20 ms, which is a 95% decrease in average latency time. The fact that response times improved so significantly establishes the system's capacity to leverage a real-time filter and to dynamically mitigate network congestion. Even though the system response times did not return to baseline performance, the responses were sufficiently near normal operation times to mask delayed user experience. While the reduced response time across the experimental conditions shows a synergistic effect which confirms some level of cooperation between the ML-inspired detection pipeline and nftables low-level packet handling, it also demonstrates the utility, effectiveness, and scalability of using adaptive anomaly detection with semi-dynamic operational rule enforcement for the mitigation of extreme DDoS traffic in cloud environments in near-real-time.

Scalability and Limitations

While the proposed framework performed robustly under simulated multi-vector attack scenarios, several limitations remain that define areas for further enhancement:

1. **Scalability:**

The existing framework was validated in a controlled, virtualized environment. Real cloud environments experience greater concurrency and compound multi-vector attacks. A critical direction for future work is to extend the architecture for handling distributed-scale attacks with parallel detection nodes and distributed firewalls.

2. **Computational Overhead:**

Although the real time ML module is effective, it brings extra computing cost for large traffic loads. Model efficiency might be improved by lightweight algorithms or using GPU inference to improve latency and resource efficiency.

3. **Mitigation Efficiency During Extreme Load:**

In an attack with a high volume, it is possible for mitigation rules to buildup quickly in nftables and impact speed of lookup and processing of rules. Installing adaptive rules age policies and automatic rule pruning would help ensure consistent performance in the future.

Implications for Cloud Security

The hybrid ML–nftables model that we are proposing has considerable implications for increasing cloud infrastructure robustness. Contrary to static firewalls and signature based intrusion systems, our model utilizes behavioral learning and executing rules in real-time to respond to DDoS behaviors that have not been seen before, which is extremely important in today's cloud deployments where virtual machines and containers are continuously scaling and redeploying. The ability to automatically sense, classify, and stop a DDoS attack without any human involvement greatly increases the opportunity for uptime and service reliability. The framework will utilize open-source components, making it cost-effective, transparent, and seamlessly integrated into conventional cloud orchestration platforms.

Future Directions

Building on these results, future research should explore the following areas to enhance system scalability, intelligence, and applicability:

1. **Multi-Vector DDoS Expansion:**

Extend detection logic and datasets to incorporate DNS amplification, Slowloris, and Smurf attacks, making the framework resistant to multi-layer coordinated attacks.

2. **Reinforcement Learning and Adaptive Thresholding:**

Implement reinforcement learning or adaptive threshold algorithms that self-tune mitigation parameters in real time, improving accuracy during dynamic network fluctuations.

3. **Integration with Edge Computing:**

Deploy parts of the detection module closer to traffic sources at the edge layer, minimizing response delay and reducing the load on centralized cloud firewalls.

4. **Hybrid Defense Stack:**

Incorporate the proposed ML–nftables architecture to function with WAFs and SDN controllers to provide full configuration orchestration and intelligent policy distribution across cloud nodes.

Conclusion

The study was to design, implement, and test a real-time multi-vector DDoS detection and prevention framework in cloud environments. The framework provided a machine-learning–ready anomaly detection logic, as well as automated mitigation using nftables, which helps maintain service availability and performance under multiple DDoS attacks (e.g., SYN flood, UDP flood, HTTP GET flood, ICMP flood) in the cloud. Overall, the experimental results demonstrated that the adaptive hybrid approach could restore operation near normal status during DDoS attacks with average throughput recovery of 92.7%, failed requests at 91.4%, and latency improvement of 96.4%. Overall, these results confirmed the viability of utilizing adaptive ML techniques in the context of traffic analysis combined with rule-based mitigation in providing near real-time prevention without sacrificing system responsiveness. The framework proved to be scalable, adaptable, and able to effectively automate the detection and mitigation of network-layer (L3/L4) and application-layer (L7) DDoS attacks, which was contrary to contemporary static defense systems. Using the Redis message bus enabled immediate synchronization of the detection and mitigation modules, and the modularity of the framework allows the researcher to easily extend to other types of attacks and deployment scenarios. This research provides a practical, feasible, and low-cost option to secure cloud infrastructures, and it attempts to re-engage the distance that too often appears in DDoS-related research undertaken in the academic perspective, and DDoS-attack product developments and implementations in the practical sense; it builds a repeatable implementation of appropriate DDoS defenses using open-source tools, and measures effectiveness in quantifiable data. Future work on this research will involve pursuing further implementation of the model towards multi-vector DDoS coordinated attacks, applying reinforcement learning to contribute dynamic threshold mitigation, and exploring edge-based mitigation for further latency and processing cost retrofits. Enhancements in these aspects may allow the framework to achieve a more resilient application for large-scale, high IT-capacity cloud use.

Ethical Statement and Data Availability

Ethical Statement

This study took place entirely in a controlled simulation setting built for academic and experimental study of Distributed Denial of Service (DDoS) mitigation techniques. All attack simulations (SYN, UDP, HTTP GET, and ICMP floods) were done locally using open-source applications — hping3 and ApacheBench (ab) — only to challenge the response of the proposed mitigation framework. The study adhered to all institutional guidelines, as well as ethical guidelines of cybersecurity research, ensuring that no external or unauthorized systems were targeted.

Data Availability Statement

All datasets and logs included in the current study were produced in the local test environment and do not contain any sensitive or identifiable information. The experimental data (throughput, response time, failed request rates) were collected from the system logs during the controlled attack simulations. No datasets belonging to third-parties or containing proprietary information were used in this research. The configuration scripts, nftables rules, and Python-based detection modules leveraged in this study will be sent, upon a reasonable request from the corresponding author, to be used for academic and non-commercial research purposes.

Acknowledgment The authors would like to thank the Department of Computer Engineering, ABCOEP Technological University, Pune, for providing the necessary infrastructure and technical support to carry out this research.

Author Contributions:

Santosh Jaykumar Kalegore conceptualized the study, designed and implemented the DDoS detection and mitigation framework, performed experiments, analyzed the results, and prepared the initial manuscript draft. Dr. Sunil Maneb supervised the research, provided methodological guidance, critically reviewed the analysis, and contributed to manuscript revision and final approval.

Funding

"The authors declare that no funds, grants, or other support were received during the preparation of this manuscript."

REFERENCES

1. Agrawal, N., & Tapaswi, S. (2019). Defense Mechanisms Against DDoS Attacks in a Cloud Computing Environment: State-of-the-Art and Research Challenges. IEEE Communications Surveys & Tutorials, 21(4), 3769–3795. https://doi.org/10.1109/COMST.2019.2934468
2. Ahmed, M. E., & Kim, H. (2017). DDoS Attack Mitigation in Internet of Things Using Software Defined Networking. 2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService), 271–276. https://doi.org/10.1109/BigDataService.2017.41
3. Alamri, H. A., & Thayananthan, V. (2020). Bandwidth Control Mechanism and Extreme Gradient Boosting Algorithm for Protecting Software-Defined Networks Against DDoS Attacks. IEEE Access, 8, 194269–194288. https://doi.org/10.1109/ACCESS.2020.3033942
4. Alashhab, A. A., Zahid, M. S., Isyaku, B., Elnour, A. A., Nagmeldin, W., Abdelmaboud, A., Abdullah, T. A. A., & Maiwada, U. D. (2024). Enhancing DDoS Attack Detection and Mitigation in SDN Using an Ensemble Online Machine Learning Model. IEEE Access, 12, 51630–51649. https://doi.org/10.1109/ACCESS.2024.3384398
5. Alsowail, S., Sqalli, M. H., Abu-Amara, M., Baig, Z., & Salah, K. (2016). An Experimental Evaluation of the EDoS-Shield Mitigation Technique for Securing the Cloud. Arabian Journal for Science and Engineering, 41(12), 5037–5047. https://doi.org/10.1007/s13369-016-2210-7
6. Bhayo, J., Hameed, S., & Shah, S. A. (2020). An Efficient Counter-Based DDoS Attack Detection Framework Leveraging Software Defined IoT (SD-IoT). IEEE Access, 8, 221612–221631. https://doi.org/10.1109/ACCESS.2020.3043082
7. Bhushan, K., & Gupta, B. B. (2017). Security challenges in cloud computing: State-of-art. International Journal of Big Data Intelligence, 4(2), 81. https://doi.org/10.1504/IJBDI.2017.083116
8. Bhushan, K., & Gupta, B. B. (2018). A novel approach to defend multimedia flash crowd in cloud environment. Multimedia Tools and Applications, 77(4), 4609–4639. https://doi.org/10.1007/s11042-017-4742-6
9. Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network Anomaly Detection: Methods, Systems and Tools. IEEE Communications Surveys & Tutorials, 16(1), 303–336. https://doi.org/10.1109/SURV.2013.052213.00046
10. Cai, G., Wang, B., Luo, Y., Li, S., & Wang, X. (2016). Characterizing the running patterns of Moving Target Defense mechanisms. 2016 18th International Conference on Advanced Communication Technology (ICACT), 1–2. https://doi.org/10.1109/ICACT.2016.7423323
11. Chen, Y., Li, Y., Cheng, X.-Q., & Guo, L. (2006). Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. In H. Lipmaa, M. Yung, & D. Lin (Eds.), Information Security and Cryptology (Vol. 4318, pp. 153–167). Springer Berlin Heidelberg. https://doi.org/10.1007/11937807_13
12. Chen, Y., Zhang, H., Song, Y., Changzhao, P., Gao, B., Liu, W., Chen, H., Han, D., Luo, E., Plimmer, M., Sparasci, F., & Pitre, L. (2019). Thermal response characteristics of a SPRIGT primary thermometry system. Cryogenics, 97, 1–6. https://doi.org/10.1016/j.cryogenics.2018.10.015
13. Clinton, U. B., Hoque, N., & Robindro Singh, K. (2024). Classification of DDoS attack traffic on SDN network environment using deep learning. Cybersecurity, 7(1), 23. https://doi.org/10.1186/s42400-024-00219-7
14. Cui, Y., Yan, L., Li, S., Xing, H., Pan, W., Zhu, J., & Zheng, X. (2016). SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks. Journal of Network and Computer Applications, 68, 65–79. https://doi.org/10.1016/j.jnca.2016.04.005
15. David, O., Vallabhaneni, R., Lallie, H., & Caporale, G. M. (2025). Cloud Computing and Cybersecurity: Emerging Threats and Defense Mechanisms.
16. Garba, U. H., Toosi, A. N., Pasha, M. F., & Khan, S. (2024). SDN-based detection and mitigation of DDoS attacks on smart homes. Computer Communications, 221, 29–41. https://doi.org/10.1016/j.comcom.2024.04.001
17. Haseeb-ur-rehman, R. M. A., Aman, A. H. M., Hasan, M. K., Ariffin, K. A. Z., Namoun, A., Tufail, A., & Kim, K.-H. (2023). High-Speed Network DDoS Attack Detection: A Survey. Sensors, 23(15), 6850. https://doi.org/10.3390/s23156850
18. He, D., Chan, S., Ni, X., & Guizani, M. (2017). Software-Defined-Networking-Enabled Traffic Anomaly Detection and Mitigation. IEEE Internet of Things Journal, 4(6), 1890–1898. https://doi.org/10.1109/JIOT.2017.2694702
19. Hirsi, A., Audah, L., Salh, A., Alhartomi, M. A., & Ahmed, S. (2024). Detecting DDoS Threats Using Supervised Machine Learning for Traffic Classification in Software Defined Networking. IEEE Access, 12, 166675–166702. https://doi.org/10.1109/ACCESS.2024.3486034
20. Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in DDoS Attacks: Trends and Challenges. IEEE Communications Surveys & Tutorials, 17(4), 2242–2270. https://doi.org/10.1109/COMST.2015.2457491

21. Jia, Q., Sun, K., & Stavrou, A. (2013). MOTAG: Moving Target Defense against Internet Denial of Service Attacks. 2013 22nd International Conference on Computer Communication and Networks (ICCCN), 1–9. https://doi.org/10.1109/ICCCN.2013.6614155
22. Kalkan, K., Gur, G., & Alagoz, F. (2017). Filtering-Based Defense Mechanisms Against DDoS Attacks: A Survey. IEEE Systems Journal, 11(4), 2761–2773. https://doi.org/10.1109/JSYST.2016.2602848
23. Kamboj, P., Trivedi, M. C., Yadav, V. K., & Singh, V. K. (2017a). Detection techniques of DDoS attacks: A survey. 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), 675–679. https://doi.org/10.1109/UPCON.2017.8251130
24. Kamboj, P., Trivedi, M. C., Yadav, V. K., & Singh, V. K. (2017b). Detection techniques of DDoS attacks: A survey. 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), 675–679. https://doi.org/10.1109/UPCON.2017.8251130
25. Kansal, V., & Dave, M. (2017a). DDoS attack isolation using moving target defense. 2017 International Conference on Computing, Communication and Automation (ICCCA), 511–514. https://doi.org/10.1109/CCAA.2017.8229853
26. Kansal, V., & Dave, M. (2017b). Proactive DDoS attack detection and isolation. 2017 International Conference on Computer, Communications and Electronics (Comptelix), 334–338. https://doi.org/10.1109/COMPTELIX.2017.8003989
27. Mandal, S. K., Marandi, A. K., Gandhi, J., Loonkar, S., Dey, P., & Kaur, S. (2025). Novel ML-driven intrusion detection system for optimizing network security. Expert Systems with Applications, 292, 128621. https://doi.org/10.1016/j.eswa.2025.128621
28. Mishra, A., Gupta, B. B., & Joshi, R. C. (2011). A Comparative Study of Distributed Denial of Service Attacks, Intrusion Tolerance and Mitigation Techniques. 2011 European Intelligence and Security Informatics Conference, 286–289. https://doi.org/10.1109/EISIC.2011.15
29. Mohammadi, R., Javidan, R., & Conti, M. (2017). SLICOTS: An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks. IEEE Transactions on Network and Service Management, 14(2), 487–497. https://doi.org/10.1109/TNSM.2017.2701549
30. Najafabadi, M. M., Khoshgoftaar, T. M., Calvert, C., & Kemp, C. (2017). User Behavior Anomaly Detection for Application Layer DDoS Attacks. 2017 IEEE International Conference on Information Reuse and Integration (IRI), 154–161. https://doi.org/10.1109/IRI.2017.44
31. Naseer, I. (2024). Machine Learning Algorithms for Predicting and Mitigating DDoS Attacks.
32. Osanaiye, O., Choo, K.-K. R., & Dlodlo, M. (2016). Distributed denial of service (DDoS) resilience in cloud: Review and conceptual cloud DDoS mitigation framework. Journal of Network and Computer Applications, 67, 147–165. https://doi.org/10.1016/j.jnca.2016.01.001
33. Pande, S., Khamparia, A., Gupta, D., & Thanh, D. N. H. (2021). DDOS Detection Using Machine Learning Technique. In A. Khanna, A. K. Singh, & A. Swaroop (Eds.), Recent Studies on Computational Intelligence (Vol. 921, pp. 59–68). Springer Singapore. https://doi.org/10.1007/978-981-15-8469-5_5
34. Patil, N. V., Rama Krishna, C., Kumar, K., & Behal, S. (2022). E-Had: A distributed and collaborative detection framework for early detection of DDoS attacks. Journal of King Saud University - Computer and Information Sciences, 34(4), 1373–1387. https://doi.org/10.1016/j.jksuci.2019.06.016
35. Pustokhina, I., Department of Computer Science and Information Technology, Kalasalingam Academy of Research and Education, Krishnankoil, India, & Ilayaraja, M. (2019). Mitigating DDoS Attacks in Wireless Sensor Networks using Heuristic Feature Selection with Deep Learning Model. Journal of Cybersecurity and Information Management, 65–74. https://doi.org/10.54216/JCIM.000106
36. Raghul Kannan, A., Naveen Sundar, G., Narmadha, D., Gifta Jerith, G., & Poornima, R. (2025). Identifying DDoS Attacks in Software-Defined Networking Environments and Applying Machine Learning for Protocol-Wise Analysis. In A. Swaroop, B. Virdee, S. D. Correia, & Z. Polkowski (Eds.), Proceedings of Data Analytics and Management (Vol. 1302, pp. 233–245). Springer Nature Singapore. https://doi.org/10.1007/978-981-96-3381-4_20
37. Rahman, O., Quraishi, M. A. G., & Lung, C.-H. (2019). DDoS Attacks Detection and Mitigation in SDN Using Machine Learning. 2019 IEEE World Congress on Services (SERVICES), 184–189. https://doi.org/10.1109/SERVICES.2019.00051
38. Rai, A., & Challa, R. K. (2016). Survey on Recent DDoS Mitigation Techniques and Comparative Analysis. 2016 Second International Conference on Computational Intelligence & Communication Technology (CICT), 96–101. https://doi.org/10.1109/CICT.2016.27
39. Saleh, M. A., & Abdul Manaf, A. (2015). A Novel Protective Framework for Defeating HTTP-Based Denial of Service and Distributed Denial of Service Attacks. The Scientific World Journal, 2015(1), 238230. https://doi.org/10.1155/2015/238230
40. Sapkota, B., Ray, A., Yadav, M. K., Dawadi, B. R., & Joshi, S. R. (2025). Machine Learning-Based Attack Detection and Mitigation with Multi-Controller Placement Optimization over SDN Environment. Journal of Cybersecurity and Privacy, 5(1), 10. https://doi.org/10.3390/jcp5010010
41. ScholarI. (2024). LEVERAGING ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FOR THREAT DETECTION IN HYBRID CLOUD SYSTEMS. https://doi.org/10.17605/OSF.IO/HUXK8
42. Somani, G., Gaur, M. S., Sanghi, D., Conti, M., & Buyya, R. (2017). DDoS attacks in cloud computing: Issues, taxonomy, and future directions. Computer Communications, 107, 30–48. https://doi.org/10.1016/j.comcom.2017.03.010
43. Sqalli, M. H., Al-Haidari, F., & Salah, K. (2011). EDoS-Shield—A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing. 2011 Fourth IEEE International Conference on Utility and Cloud Computing, 49–56. https://doi.org/10.1109/UCC.2011.17

44. Srinivasan, K., Mubarakali, A., Alqahtani, A. S., & Dinesh Kumar, A. (2020). A Survey on the Impact of DDoS Attacks in Cloud Computing: Prevention, Detection and Mitigation Techniques. In S. Balaji, Á. Rocha, & Y.-N. Chung (Eds.), Intelligent Communication Technologies and Virtual Mobile Networks (Vol. 33, pp. 252–270). Springer International Publishing. https://doi.org/10.1007/978-3-030-28364-3_24
45. Taha, M. B. (2025). Hybrid Ensemble Learning Framework for Real-Time DDoS Detection and Mitigation in SDN Environments.
46. Van Adrichem, N. L. M., Doerr, C., & Kuipers, F. A. (2014). OpenNetMon: Network monitoring in OpenFlow Software-Defined Networks. 2014 IEEE Network Operations and Management Symposium (NOMS), 1–8. https://doi.org/10.1109/NOMS.2014.6838228
47. Wang, Y., Liu, L., Sun, B., & Li, Y. (2015). A survey of defense mechanisms against application layer distributed denial of service attacks. 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 1034–1037. https://doi.org/10.1109/ICSESS.2015.7339229
48. Ye, J., Cheng, X., Zhu, J., Feng, L., & Song, L. (2018). A DDoS Attack Detection Method Based on SVM in Software Defined Network. Security and Communication Networks, 2018, 1–8. https://doi.org/10.1155/2018/9804061
49. Zargar, S. T., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. IEEE Communications Surveys & Tutorials, 15(4), 2046–2069. https://doi.org/10.1109/SURV.2013.031413.00127
50. Zhijun, W., Wenjing, L., Liang, L., & Meng, Y. (2020). Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey. IEEE Access, 8, 43920–43943. https://doi.org/10.1109/ACCESS.2020.2976609