International Journal of Safety and Security Engineering
Vol. 16, No. 2, February, 2026, pp. 351-363
Journal homepage: http://iieta.org/journals/ijsse

A Novel STRIDE Model for Improved Security and DDoS Mitigation for Software-Defined Networking

Sugandhi Malhotra[1], Riya Sapra[2], Chaitanya Singla[1], Chander Prabha[3], Snehal Bhosale[4], Durgesh Nandan[5*]

[1] Department of Computer Science and Engineering, School of Engineering and Technology, CGC University, Mohali 140307, India
[2] Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Haryana, Gurgaon 122413, India
[3] Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab 140401, India
[4] E&TC Department, Symbiosis Institute of Technology, Pune Campus, Symbiosis International (Deemed University), Pune 412115, India
[5] School of CS & AI, SR University, Warangal 506371, India

Corresponding Author Email: durgeshnandano51@gmail.com

https://doi.org/10.18280/ijsse.160210

Received: 1 October 2025
Revised: 5 January 2026
Accepted: 13 January 2026
Available online: 28 February 2026

Keywords:
software-defined networking, network security, Distributed Denial-of-Service mitigation, OpenFlow, STRIDE, POX controller, entropy measures

ABSTRACT

Software-defined networking (SDN) is an innovative network architecture that supports the dynamic, high-bandwidth requirements of modern applications while enabling rapid adaptation to changing business needs. However, the centralized control paradigm of SDN also creates important security challenges, particularly under Distributed Denial-of-Service (DDoS) attacks. To address this issue, this study proposes P-STRIDE, a novel defense mechanism that strengthens security in SDN environments by combining statistical analysis with OpenFlow entries. P-STRIDE is inspired by the STRIDE security framework, which addresses Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege threats. By integrating OpenFlow table entries with a statistical controller based on POX, the proposed mechanism provides effective threat mitigation. In the reported results, P-STRIDE achieved an error detection rate of 98.76%, whereas the 6-tuple Self-Organizing Map (SOM) classification method achieved an accuracy of 67.08%. In addition, as the number of packets increased, P-STRIDE maintained better latency performance. At 200 packets, the latency was 10 ms, compared with 28 ms for conventional methods. These results indicate that P-STRIDE with a POX controller performs effectively under DDoS traffic, mitigates multiple security threats, and offers a practical approach to entropy handling in SDN networks.

1. INTRODUCTION

Software-defined networking (SDN) provides a solution by enabling network virtualization, dynamic policy enforcement, and improved control over network entities, all at a reduced operational cost [1, 2]. However, SDN's centralization of the control plane creates a significant burden for administrators to ensure the security and proper functioning of the entire network. Compromised network entities pose a threat, which can be exploited to exfiltrate sensitive information, launch targeted attacks, or even take down the entire network. Figure 1 depicts the SDN conceptual architecture [3]. The data plane stays on the network device, processing network packets according to flow rules. In contrast, the control plane is disconnected from the network device and resides on a centralized controller, deciding how network packets are routed.

Several controllers are occasionally necessary for replication or in bigger networks. These controllers connect with one another using an east/west-bound API, whereas the data plane communicates with the control plane via a south-bound API. The OpenFlow protocol is the most widely used south-bound API, defined by the Open Networking Foundation (ONF) [4]. A whole network may be controlled using SDN and OpenFlow controllers while keeping the data plane distinct from the control plane. This method merely allows switches to forward data packets while denying them control functionalities. Control over data is centralized and consistent, resulting in efficient network resource management and considerable gains in network controllability. Because SDN uses OpenFlow [5], communication between controllers and switches should follow OpenFlow standards. Flow tables are used by OpenFlow switches to match and transmit data packets depending on information supplied and updated by higher controllers.

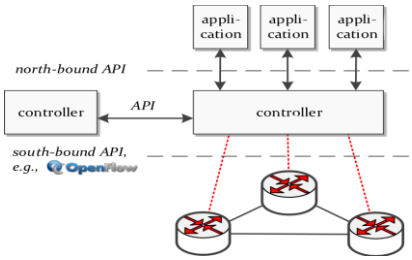


Figure 1. Architecture of software-defined networking (SDN), separation of the network (control) plane, and the forwarding plane

SDN affects network security in both positive and negative ways. On the one hand, it may perform security functions like traffic filtering and redirection based on packet content or application layer status without requiring additional security devices. On the other side, the separation of planes in SDN might introduce new attack vectors that risk network availability as well as the integrity, confidentiality, and authenticity of network data.

Despite research towards SDN-based Distributed Denial-of-Service (DDoS) attack detection, these attacks continue in frequency, complexity, and intensity. Because attackers are dispersed over the network, coordinated network-wide monitoring is critical for efficiently detecting DDoS attacks. A quick response to traffic irregularities is required for timely DDoS detection and mitigation. Since SDN's centralized controller can instantly alter measurement criteria on all switches in a coordinated fashion, it's perfect for DDoS detection. Furthermore, if recognized, the controller may quickly apply blocking rules to halt attack traffic [6].

The work known as P-STRIDE proposes a Mixed statistical defense mechanism and OpenFlow Entries-based approach for security in SDN networks [7]. This work is inspired by the STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege) [8] security framework. P-STRIDE achieves threat safety utilizing both OpenFlow table entries and a statistical controller based on POX [9]. The evaluation demonstrates that P-STRIDE can perform well under DDoS Traffic and can help mitigate SDN networks.

1.1 Research contributions

This paper aims to contribute to the field of SDN security by proposing a secure and efficient security model that effectively mitigates STRIDE security threats. Specifically, our proposed model, named P-STRIDE, addresses the following security concerns:

- Spoofing (S): By implementing measures to prevent IP spoofing and Address Resolution Protocol (ARP) spoofing, which can be leveraged for man-in-the-middle attacks against hosts on a computer network.
- Tampering (T): By ensuring that only authorized users can modify and make changes to the available information.
- Repudiation (R): By establishing mechanisms to ensure the authenticity and validity of information and data exchange between parties, thus preventing repudiation.
- Information Disclosure (I): By implementing measures to verify and validate information to the correct user, thus avoiding the risk of information disclosure.
- Denial of Service (D): Our proposed study also includes methods to detect, analyze, and prevent DDoS attacks by the controller and switches.
- Elevation of Privilege (E): In addition, our study assesses how machines on SDN can negatively impact host authentication and access privileges. We also examine various authentication protocols and propose methods to identify hackers.

In summary, this paper's main contribution is the P-STRIDE security model, which effectively addresses a range of security threats in SDN.

1.2 Organization

In conclusion, our research focuses on the obstacles and issues associated with installing authentication systems and presents secure deployment alternatives.

The remainder of the paper is organized as follows: Part 2 highlights the suggested security solutions and offers an overview of relevant studies in the field of SDN networks. The STRIDE model and its concept are explained in Section 3. Part 4 describes the P-STRIDE security paradigm that we propose. Section 5 describes the topology utilized to simulate our research. In Section 6, our study technique is given with performance measures such as network throughput, packet delivery ratio, and latency, as well as comparisons between utilizing the P-STRIDE model and not using the model. Sections 7 and 8 represent the culmination of our work and future work, respectively.

2. RELATED WORK

Substantial studies on security issues in SDN have resulted in a substantial body of literature. Schehlmann et al. [10] weighed the security benefits and downsides of SDN and determined that the benefits outweighed the risks, making SDN a security boon. Balarezo et al. [11] examined the novel security benefits offered by SDN and how it can address longstanding network security issues. The authors also identified SDN's new security threats and suggested ways to prevent and mitigate them. Ahmad et al. [12] examined the security vulnerabilities of SDN's application, control, and data planes, as well as the security platforms that protect each.

Numerous techniques have been proposed in the literature for enhancing security in SDN networks, but it is challenging to determine which of these mechanisms is superior. For instance, a lightweight method for detecting DDoS attacks using a low-overhead approach over the NOX platform was proposed in the study by Braga et al. [13], but the NOX platform is no longer being maintained and has been superseded by the POX controller. Moreover, the evaluation was conducted on the KDD-99 dataset, which is not ideal for real-time evaluations. Wang et al. [14] proposed an entropy-based lightweight DDoS flooding attack detection model running in the OpenFlow edge switch. However, using only entropy as a means of flow detection is insufficient in normal networks, and SDN is likely to face similar challenges.

Researchers have investigated multiple approaches for leveraging SDN to effectively block large-scale DDoS attacks that appear legitimate. One such method proposed in the study by Lim et al. [15] is to enhance the functionality of the SDN controller using flow statistics to detect anomalies and help allocate network resources. However, this approach may not be practical for high-traffic scenarios. Another approach

proposed in the study by Giotis et al. [16] is a distributed collaborative framework that allows customers to request DDoS mitigation services from their ISPs. While this alternative provides a way to detect and mitigate DDoS attacks, its scalability and operational costs in large-scale deployments are questionable.

In addition, Dayal and Srivastava [17] identified various possible DDoS attacks in the SDN environment using an attack tree and model. They suggested using an SVM-based architecture for the SDN network to defend against and detect the attack source, which is known as the FLGUARD system [18]. However, this method may generate a large number of data packets sent to the controller, which can impact network security and risk range.

Authentication is also a significant concern in SDN networks. One study proposed by Guodong et al. [19] used cipher identification information during packet flow to verify their source. Nevertheless, because this technique does not verify users when packets enter or exit the network, network security and risk range are limited. Aladaileh et al. [20] presented an early detection technique to allow remedial steps to be taken at the right moment. Nevertheless, this strategy necessitates strong coordination among multiple network components, making it difficult to adjust networking device behavior and achieve coordination in traditional networks.

Hadi et al. [21] developed a security system that is both simple and fully customizable, making use of SDN to achieve a high level of security with minimal latency costs. However, the implementation of strict policies may lead to an increase in controller rejections, potentially impacting performance. Ilyas and Khondoker [22] conducted a Security Analysis of various SDN controllers, including Beacon, ZeroSDN FloodLight, and POX, concluding that SE-FloodLight is the most secure controller, while the POX controller is the most configurable. This paper did not focus on attack detection models but rather on the selection of an appropriate controller for SDN, prompting us to adopt the POX controller for our STRIDE framework.

Several approaches have been proposed to address DDoS attacks in SDN networks. However, most of these approaches have limitations that make them less practical in real-world scenarios. To this end, we present a statistical defense mechanism against DDoS attacks for SDN networks in this work. Our mechanism relies on statistical analysis of network traffic to identify abnormal patterns indicative of DDoS attacks. Specifically, we use the entropy-based approach [14, 15] as the foundation of our mechanism, extending it to incorporate packet header information and flow size, allowing us to detect DDoS attacks more accurately while minimizing false positives. Unlike entropy-based schemes [14, 15] and FLGUARD, which primarily focus on detecting Denial-of-Service (DoS/DDoS) attacks, P-STRIDE provides comprehensive protection across all STRIDE threat categories, including Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privileges. This holistic security coverage is achieved through a tight integration of OpenFlow table rules and controller-level logic, rather than relying solely on traffic statistics or machine learning classifiers.

3. METHODOLOGY: P-STRIDE THREAT MODEL

The POX controller, which is highly configurable and ideal for our purposes, was utilized to implement our mechanism. Additionally, we use machine learning algorithms to classify network traffic as either normal or anomalous, enabling us to adapt to new and evolving attack patterns. We evaluate our mechanism's effectiveness by conducting experiments on real-world traffic traces and comparing our results to existing approaches. Our results indicate that our mechanism has higher detection rates and lower false positives than other methods. Furthermore, our mechanism is scalable and can be implemented in large-scale SDN networks without causing significant performance overhead.

In summary, our proposed statistical defense mechanism provides an effective and practical solution for mitigating DDoS attacks in SDN networks. Our mechanism overcomes the limitations of existing approaches while leveraging their strengths, making it a promising solution for real-world deployments.

As the proposed P-STRIDE has been inspired by the STRIDE model. The STRIDE attribute of the proposed model is explained in subsections 3.1-3.3. The STRIDE attributes are implemented using various OpenFlow approaches, some including flow table entries and some using the POX controller. The STRIDE model is as follows.

3.1 S: Spoofing (Authentication)

Spoofing is a technique that allows attackers to deceive a system and hide or falsify their identity. The P-STRIDE model enforces mandatory authentication procedures to ensure the trustworthiness of remote or local application commands. The Spoofing in P-STRIDE is handled by OpenFlow entries to disable ARP attacks in the network, as shown in Table 1. As seen in the table, the ARP entries identified by ARP (nw_proto = 0×806) in L1 matching handle the spoofing attack efficiently.

3.2 T: Tampering (Integrity)

Tampering with a network can result in compromising the integrity of data that is being transported or stored. In this work, we have implemented the Message-Digest Algorithm 5 (MD5) to prevent packet tampering by verifying the integrity of packets during their transmission. The MD5 algorithm employs hash checks to prevent man-in-the-middle attacks.

3.3 R: Repudiation (Non-repudiation)

The absence of monitoring capabilities in switches and control software can lead to repudiation, enabling system users to deny their actions or shift the blame. To address this issue, we have utilized OpenFlow table mappings with L1 and L2 rules to enable non-repudiation, taking advantage of the unique identification of SDN devices. With the automatic and secure logging and tracking mechanisms in the OpenFlow tables used in Table 1, repudiation can be prevented.

3.4 Information disclosure (Confidentiality)

By abusing any vulnerability, a node in the network might reveal sensitive data or passwords to an attacker. Using the centralized SDN controller and information storage improves network performance. Redirecting illegitimate traffic and sending packets to mask the sender's identity can further improve confidentiality.

The P-STRIDE uses OpenFlow's flow modification (mod_nw_tos) to send information so that the receiver can't identify the actual source of the packets. As mentioned in Table 1, the flow entries are modified in such a way that the receiver finds the IP address of the source. For example, the IP address of 8.8.8.8 was used to mask the actual IP address (10.0.0.4) using flow modification to increase confidentiality in the network.

Table 1. Flow entries used for enabling STRIDE functionality in software-defined networking (SDN)

Attack / STRIDE Component	Layer 1	Layer 2				Layer 3			Action	Remarks
	IN Port	Src	Dst	Type	flow-id	Source	Destination	Protocol	Out Port	-
Spoofing (S)	*	*	*	0×806	-	10.0.0.0/25	10.0.0.3	ARP	Drop	ARP requests
Tampering (T)	*	*	*	0×806	-	*	*	TCP	Controller	Check MD5 hash
Repudiation (R)	80	*	*	0×800	-	10.0.0.0/25	10.0.0.2	TCP	tp_dst = 80,output:3	Redirect traffic
Information disclosure (I)	21	*	*	*	-	8.8.8.8	*	TCP/UDP	mod_nw_tos	Flow modification
Denial of service (D)	*	*	*	*	-	*	*	TCP/UDP	Controller	DDoS controller
Elevation of privileges (E)	*	*	*	*	6	10.0.0.0/25	10.0.0.2	TCP	Priority = 100	-

Note: Address Resolution Protocol (ARP); Transmission Control Protocol (TCP); User Datagram Protocol (UDP); Distributed Denial-of-Service (DDoS); Message-Digest Algorithm 5 (MD5).

3.5 Denial of service (Availability)

SDN networks are vulnerable to DDoS attacks, which can cause significant financial losses and disrupt legitimate user access to network resources. To address these challenges, a custom controller has been developed that includes intrusion detection mechanisms, statistical analysis of the attack, and redundancy features for effective mitigation of DDoS attacks. This custom POX controller is discussed further in the following section.

3.6 Elevation of privileges (Authorization)

One reason for the vulnerability of some systems is their lack of access control, which may allow users to increase their authority and gain access to restricted or classified assets. To mitigate this risk, P-STRIDE verifies the priority of packets that attempt to access sensitive information. When the priority is missing, P-STRIDE reduces it to a lower value (such as 100) by modifying the flow.

4. DISTRIBUTED DENIAL-OF-SERVICE FLOODING ATTACK DETECTION

This section presents our proposed DDoS detection model during attacks in the SDN network. In DDoS attacks, the attacker forges its IP address or uses zombies to attack the target server it also changes port numbers to hide where the attack is coming from. Thus, from the point of view of the target network, there can be multiple input flows to the SDN Controller switch. We need to perform the flow aggregation from the IP Address to detect the attack. Flow aggregation helps identify the sources from which the expected attack might come.

4.1 Software-defined networking flow aggregation

During an attack, the attacker targets the victim's servers within the SDN network. The incoming traffic originates from various IPsrc, and to manage multiple flows directed toward the same target address (IPdst), the controller initiates a flow aggregation process. Figure 2 illustrates this aggregation mechanism, where the destination is overwhelmed with packets, simulating an attack scenario. To mitigate this, the proposed SDN controller employs a hashing table to efficiently aggregate Source-Destination IP pairs, as shown in Table 2.

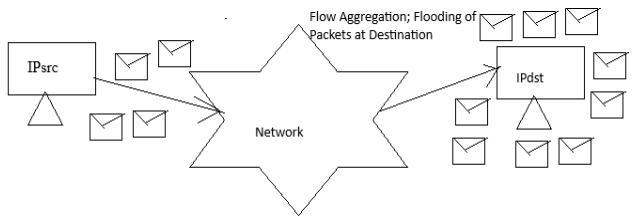


Figure 2. Flow aggregation

Table 2. Flow entries used for enabling STRIDE functionality in software-defined networking (SDN)

Source IP Address, Port	Destination IP Address	Packet Count
192.162.1.2,45	10.0.0.3	4651
192.162.1.5,41	10.0.0.1	3212
…	…	…
192.162.1.15,80	10.0.0.48	1326

Applying statistical evaluations, such as counting total active IP addresses with packet count during the specified time window. We can isolate IP Addresses such that the packet count of the incoming packets sent by the source IP addresses in each timeframe T exceeds the specified threshold ∂.

$$H(IP_j, C) = \sum_{j=i}^{N} \sum_{t=1}^{T} PacketCount(IP_j) \quad (1)$$

where, H is the hash table with size N.

IP_j represents any active IP address during the time frame $t \in T$.

$H(IP_j, C)$ represents the hash table with IP address.

C is the packet count.

't' represents the time span 't'.

For all the 'N' nodes, an entry about incoming traffic is maintained in the hash table to detect the attack.

The Proposed SDN controller then identifies the IP address which exceeds the predefined threshold ∂. The threshold ∂ can be identified by knowing the network bandwidth. The IP addresses which exceed the threshold ∂ are put in a blocking basket such that if $H(IP_j, C) > \partial$, block the IP address IP_j using OpenvSwitch (OVS) flow entries. The detection method checks packet count per IP. If an attack is detected, then traffic is diverted to a black hole. To reduce false positives, P-STRIDE distinguishes between legitimate high-traffic flows and malicious traffic. Instead of using a fixed value, thresholds are dynamically adjusted based on real-time network conditions. The proposed algorithm was implemented in POX Controller. The Controller works in independent mode, the SDN switch works in parallel, doing the forwarding for normal packets and retaining the normal flow of the SDN mechanism, as shown in Figure 3.

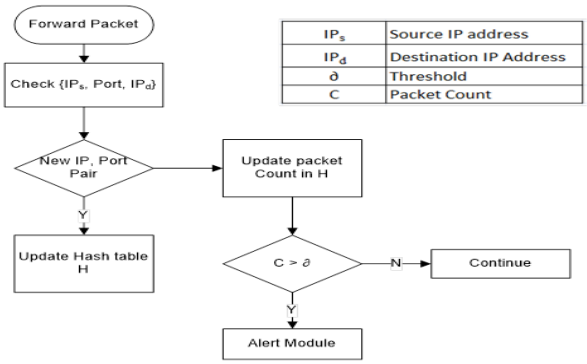


Figure 3. Working procedure of the P-STRIDE controller

5. TOPOLOGY USED

The OpenFlow-based SDN topology shown in Figure 4, with the help of miniedit, has been considered in this work. The topology, as depicted in Figure 4, consists of various hosts h1-h9, OVS-based switch s1, and POX-based custom controller c0. We have used 50mb/s of network bandwidth between the hosts and the switch and a 10 ms delay for evaluating the controller's performance [23].

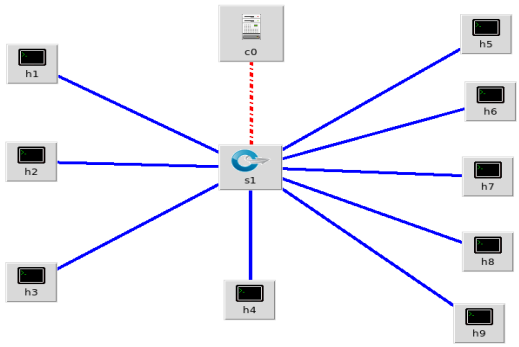


Figure 4. Network topology used to evaluate the P-STRIDE framework

6. SIMULATION RESULTS AND DISCUSSION

In the first part of our experiment, we selected a controller to use. We chose POX [24] because it is popular for experiments, is fast and lightweight, and is designed as a platform for building custom controllers. POX is an improved version of NOX [25], written in Python and used in many SDN papers mentioned in our paper. However, NOX is no longer being developed [26], which is why we opted to use POX instead. We developed a custom POX controller that uses statistical evaluation on the flows to identify DDoS attacks from incoming packets in the OpenFlow network.

Existing entropy-based methods [14, 15] rely mainly on statistical measures (e.g., entropy of flow distributions) computed at switches or the controller, which can suffer from high false positives under legitimate traffic surges. FLGUARD uses a centralized SVM-based classifier, resulting in increased control-plane communication overhead. In contrast, P-STRIDE introduces a hybrid co-design, where:

- Lightweight threat prevention (e.g., ARP spoofing, repudiation, privilege escalation) is enforced directly at OpenFlow tables,
- Computationally intensive DDoS detection is handled by a custom statistical POX controller, minimizing unnecessary packet-in events and reducing controller overload.

The experiments were conducted using Iperf [27] and the custom DDoS Scripts written in Python for simulating the DDoS attack. Traffic flows were generated during the normal and attack periods. The POX implementation, P-SRIDE, is compared with the Native OpenFlow architecture for comparison. During the testing, 50% of flows were collected during the attack period, and the other half were collected during normal traffic periods. Attacks were performed at the rate of 200 packets per second (PPS) on a 50Mb/s base network bandwidth. It is clear from Figure 5 that the P-STRIDE framework reduces network traffic, i.e., unwanted network traffic, significantly.

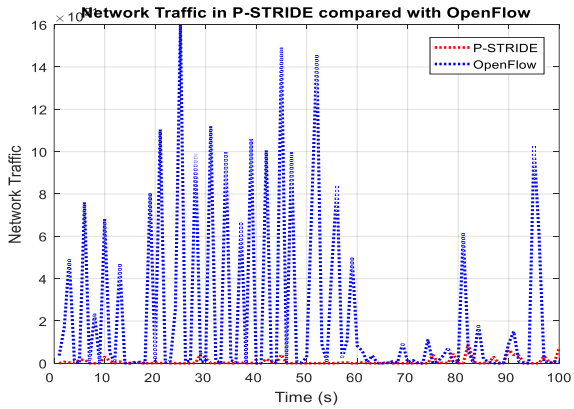
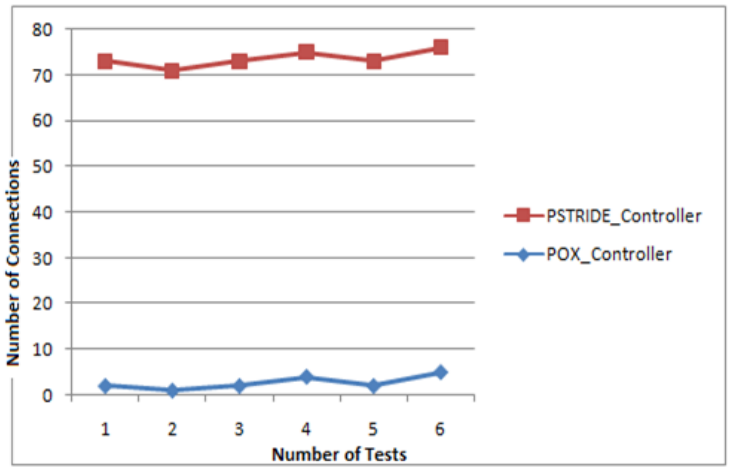


Figure 5. Traffic during the 200 PPS attack traffic

P-STRIDE possesses the potential to discover the abrupt and abnormal flow of traffic, as the modified POX controller has been trained to compare the traffic flowing at a point from a particular host with the predefined traffic flow. If it is not below the specified threshold, the modified POX Controller detects and notifies of the attack [28, 29]. Normal traffic, as depicted in red, is allowed to pass to the network hosts being the normal traffic flow. The Traffic peaks are visible for the case of OpenFlow in blue.



	POX_Controller	PSTRIDE_Controller
1	2	71
2	1	70
3	2	71
4	4	71
5	2	71
6	5	71

Figure 6. Network connectivity test 200 PPS attack traffic

The results depict that the native OpenFlow controller is unsuitable for handling DDoS traffic. Results for 6 test cases, as depicted in Figure 6, have been generated at different amounts of time to check the connectivity and reach through the "ping" command.

At heavy load time, at an interval of every 0.025 seconds, a new packet is sent, and results are generated after every 120 seconds. To ensure that the hosts in the network are connected and data can be transferred between them, the "pingall" statement is used to check the connectivity. For transferring files between reachable hosts, the Linux "wget" feature is used. The experiments showed that the modified P-STRIDE Controller was able to handle attacks more effectively compared to the OpenFlow POX Controller, which failed to detect the attack and resulted in the SDN Controller going down. It controls legitimate users from connecting. The entire network service is hampered due to the attack on the SDN Controller.

In the case of P-STRIDE, timely detection and handling avoid such problems. Legitimate users can connect and continue with the services.

Assessing bandwidth usage is a common step in determining the quality of service (QoS) in a network. A network's maximum data transmission rate is referred to as its bandwidth, and excessive use might result in alarms. An additional advantage of P-STRIDE is that it can measure bandwidth use. We can clearly understand the network's bandwidth consumption by calculating the ratio of regular traffic to the network's overall capacity, and it can be calculated using Eq. (2).

$$BWUtil = \frac{Normal\ Traffic\ Bandwidth}{Total\ Capacity} \quad (2)$$

where,

BWUtil represents Bandwidth Utilization Ratio, which represents the proportion of the available network bandwidth currently consumed by legitimate (normal) traffic. It is a dimensionless value ranging from 0 to 1.

Normal Traffic Bandwidth represents the bandwidth consumed by normal traffic, denoting the amount of network bandwidth used by legitimate, non-malicious flows during the observation interval. This parameter is typically measured in bits per second (bps) or megabits per second (Mbps) and is derived from flow-level statistics collected at the SDN switches.

Total Capacity represents total available network bandwidth, which refers to the maximum bandwidth capacity of the network link or aggregate links under consideration, expressed in the same units as Normal Traffic Bandwidth.

The bandwidth utilization depicts how much normal traffic is flooded in the network. During the DDoS attack, as seen clearly from Figure 7, the bandwidth utilization of the OpenFlow implementation keeps on decreasing. However, as P-STRIDE works by eliminating illegitimate network traffic, the bandwidth utilization remains close to normal. Bandwidth usage at a particular time has been calculated using "Iperf".

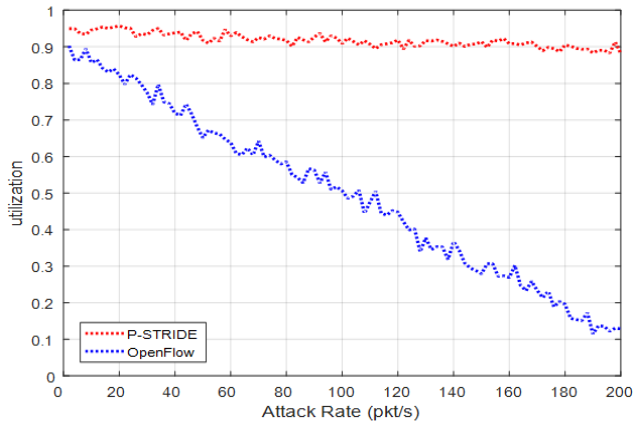


Figure 7. Bandwidth utilization during different traffic rates

OpenFlow struggles to handle adaptive attack patterns in the environment, resulting in excessive consumption of network resources. P-STRIDE is dynamically adapting to filter the attack traffic, which is preventing disruptions in bandwidth usage.

The number of flow entries made in the OpenFlow tables acts as a great measures that affect the network's bandwidth usage. This fact can also be realized by studying the No. of flow entries in the Open Flow Controller table, as shown in Figure 8. During an attack and even in normal traffic mode, the P-STRIDE helps in removing unwanted entries in Open Flow tables as much as 19%. Due to a large number of flow entries that need to be written at the time of the attack in the OpenFlow POX, the performance of the OpenFlow in terms of bandwidth utilization and latency is badly affected.

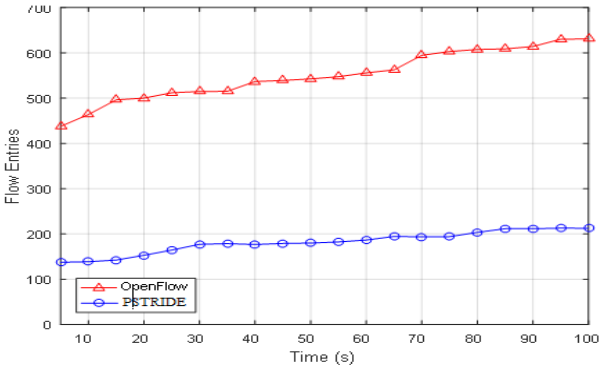


Figure 8. Number of flow entries during the attack at different times

Delay time refers to the time elapsed between when a request is generated and when the data is delivered. An increase in delay time can indicate significant performance issues. Figure 9 shows that the presence of many flow tables in the OpenFlow implementation can degrade the network's latency.

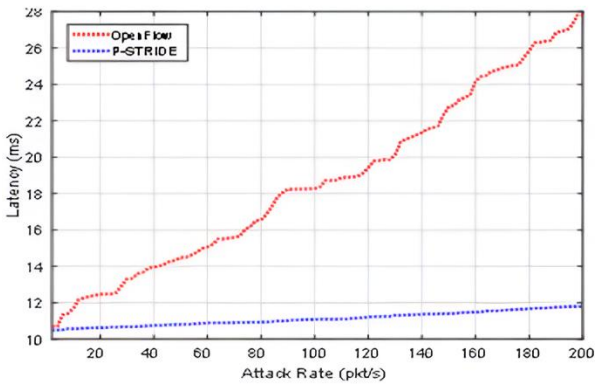


Figure 9. Expected latency in the network with P-STRIDE vs. OpenFlow – given the normal mode delay at 10 ms

We use the ping command to measure packet loss and latency, which checks the connectivity between virtual machines and measures their performance. The Latency can be calculated using Eqs. (3) and (4). In real-time traffic, high jitter can degrade performance even if average latency remains within acceptable limits. P-STRIDE actively reduces jitter by prioritizing critical traffic and uses traffic shaping.

$$Latency = Round\ TripTime\ (RTT) \tag{3}$$

$$Latency = Message\ Sent\ Time_{Host1 \rightarrow Host2} + Response\ Time_{Host2 \rightarrow Host1} \tag{4}$$

where,

Latency is the total time delay experienced by a data packet to travel from the source host to the destination host and back. It reflects end-to-end communication delay in the network and is typically measured in milliseconds (ms).

Round Trip Time (RTT) is the time interval between sending a request packet from the source host and receiving the corresponding response packet. RTT captures both forward and return path delays, including transmission, propagation, processing, and queuing delays.

$Message\ Sent\ Time_{Host1 \rightarrow Host2}$ is the time taken for a message or data packet to travel from Host 1 (source) to Host 2 (destination). This includes transmission delay, propagation delay, and any intermediate processing or queuing delays along the forward path.

$Response\ Time_{Host2 \rightarrow Host1}$ is the time required for the response packet to travel from Host 2 back to Host 1, encompassing response generation time at Host 2 as well as transmission, propagation, and queuing delays along the return path.

Ping sends "echo request" and "echo reply" packets over ICMP. The user considers high latency to be an undesirable quality. In an ideal world, latency is limited by the speed of light in that fiber. P-STRIDE supports real-time traffic. The suggested P-STRIDE architecture has a delay of 11.7 ms when the nodes in the network have a typical traffic latency of 10 ms, when there is no assault.

Measuring the packet drop ratio is crucial for both network operators and application users as it indicates the performance of the network. The ratio shows the number of packets sent by the source in relation to the number of packets received by the destination. Analyzing such measures can offer valuable information for enhancing the network's performance. It can be calculated using Eqs. (5) and (6).

$$PDR(\%) = \frac{No.of\ Packets\ Received\ by\ Receiver}{No.of\ Packets\ Sent\ by\ Receiver} \tag{5}$$

where,

Packets Received is the number of data packets successfully delivered to the receiver.

Packets Sent is the total number of data packets transmitted by the sender.

It also measures the number of packets dropped by nodes due to various reasons.

$$Packet\ Loss = No.of\ Packets\ Sent - No.of\ Packets\ Received \tag{6}$$

where,

Packets Sent is the total number of packets transmitted by the sender.

Packets Received is the number of packets successfully received at the destination.

It has been observed that in the proposed P-STRIDE data, in the case of a normal connection, which is near 100%, the initial 2% loss is due to path selection and connection delay, and Time-To-Live (TTL) parameters common in all the networks; it is an ideal state. As the traffic is increasing, OpenFlow POX fails to generate the ideal results as depicted in Figure 10, and due to late or poor detection of attacks, it falls at a rapid rate.

Throughput (T) is the count of data packets successfully delivered from source to destination per unit of time, as visualized in Figure 11, and it is calculated using Eq. (7).

$$T = \frac{1}{n}\sum_{i=1}^{n}\frac{b_i}{t_i} \tag{7}$$

where,

b_i = Total amount of data.

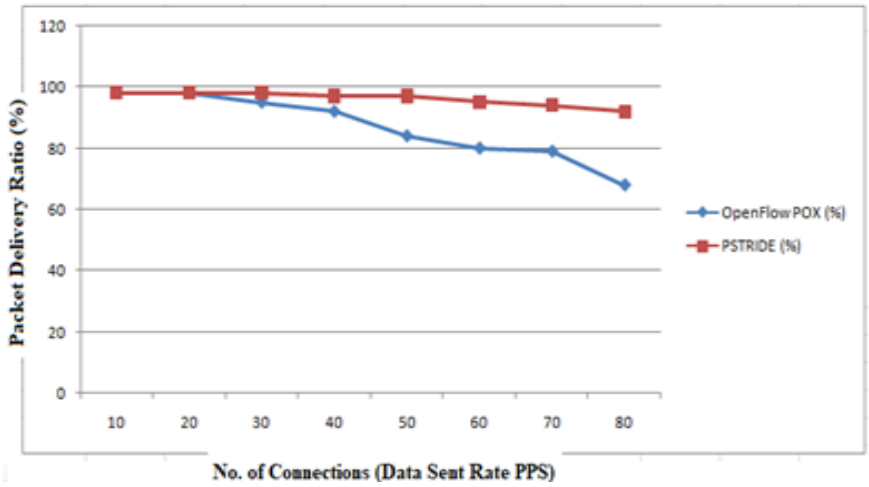
t_i = time taken for a destination to get the final packet.

n = total no of application traffic.

The equation calculates the average throughput across

multiple application traffic flows, where each flow may have different data sizes and transmission times. It provides an overall measure of how efficiently data is being transmitted in a system with multiple applications.

A high ratio of unsuccessful message delivery in the existing method also degrades performance in terms of throughput. Figure 12 represents the throughput at different times with varying network traffic rates. As the packet sending rate increases, the traffic rises, and throughput on account of P-STRIDE has minimal impact on the attack.



No. of Connection	OpenFlow POX (%)	PSTRIDE (%)
10	98	98
20	98	98
30	95	98
40	92	97
50	84	97
60	80	95
70	79	94
80	68	92

Figure 10. Packet delivery ratio (%) m PSTRIDE with OpenFlow POX

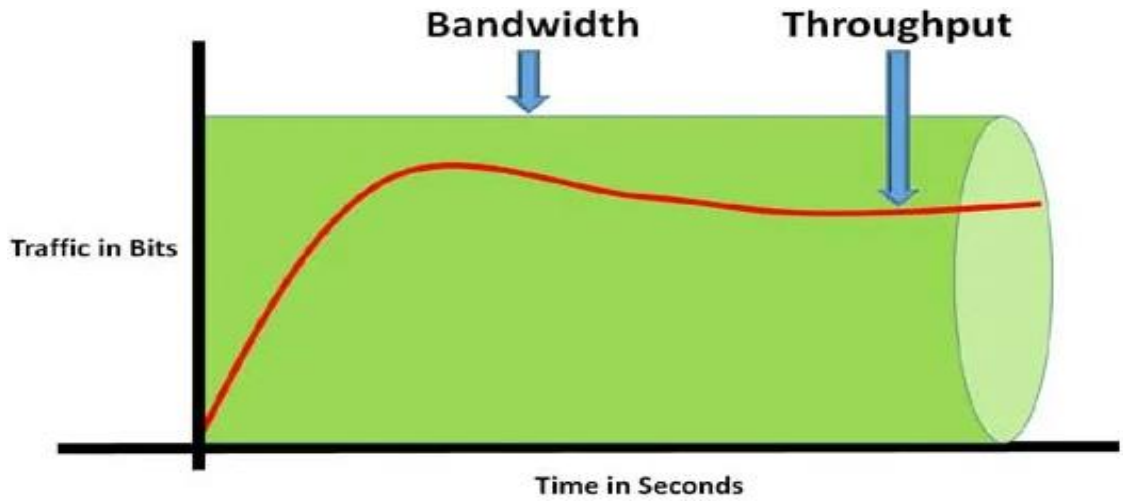
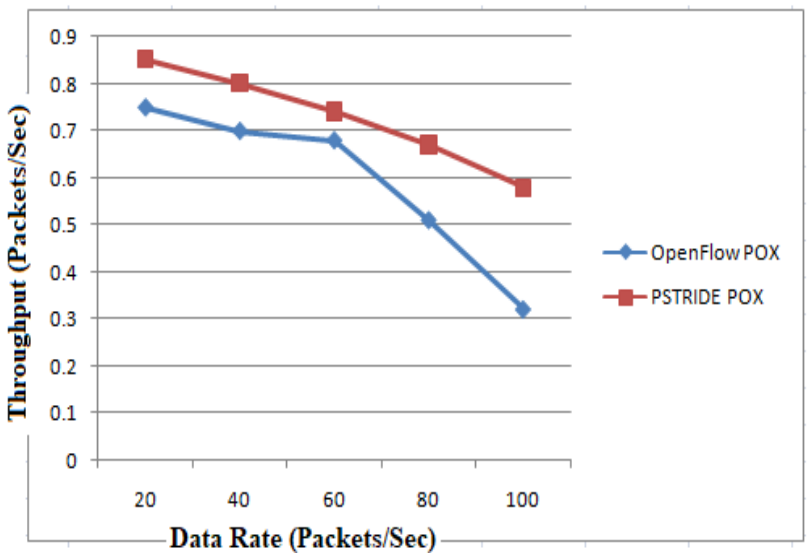


Figure 11. Throughput



Data Rate (Packets/Sec)	OpenFlow POX	PSTRIDE POX
20	0.75	0.85
40	0.7	0.8
60	0.68	0.74
80	0.51	0.67
100	0.32	0.58

Figure 12. Throughput rate in PSTRIDE with OpenFlow POX

Our maximum achievable data transfer rate, even under network congestion conditions with an increased packet load, exhibited only a marginal decrease in throughput from 0.67 to 0.58 when the data rate was increased from 80 to 100. However, under the same conditions, OpenFlow POX experienced a substantial throughput drop from 0.51 to 0.32. Additionally, in scenarios with higher packet loss due to congestion, the necessity for retransmissions consumes extra bandwidth, leading to a significant degradation in throughput.

It is defined as the average time taken by data packets to propagate from source to destination across the network and is calculated using Eq. (8).

$$End\ End\ Delay = \sum (Arrived\ Time - Sent\ Time) \quad (8)$$

where, *Arrived Time* is the time at which a packet reaches the destination.

Sent Time is the time at which the packet is transmitted by the source.

The average delay, as shown in Figure 13 are calculated based on the following Eq. (9).

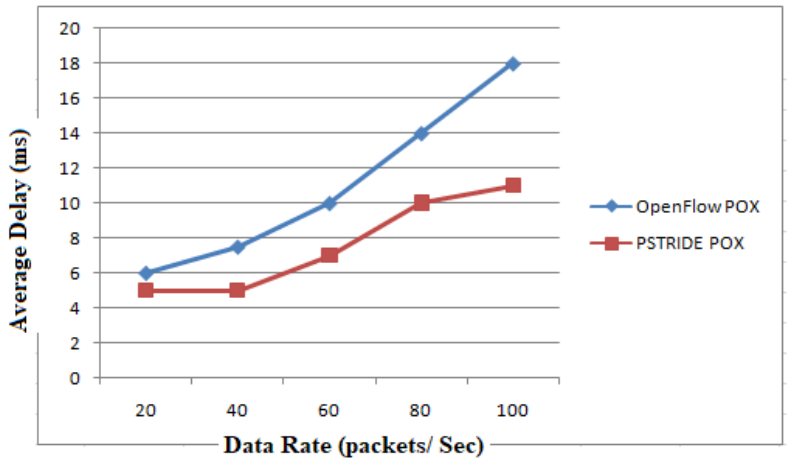
$$Average\ Delay = \frac{1}{(\mu - \delta)} \quad (9)$$

where,

μ: Packet service rate of the system (packets per unit time).

δ: Packet arrival rate to the system (packets per unit time).

where, μ is the number of PPS and δ is the average at which packets are arriving. End-to-end delay is observed to be initially the same with a marginal difference in the case of P-STRIDE and existing, but with the increase in traffic and attack rate, delay margins in both increase. P-STRIDE has given better results in terms of reduced delay time, even at heavy load timings.



Data Rate (Packets/ Sec)	OpenFlow POX	PSTRIDE POX
20	6	5
40	7.5	5
60	10	7
80	14	10
100	18	11

Figure 13. Average delay in PSTRIDE with OpenFlow POX

OpenFlow POX works well when there is no attack, and normal average traffic, too, is comparable to P-STRIDE, as shown in Figure 14. The average Normal flow is less in P-STRIDE due to flow rules modifications in the OpenFlow Table Flow entries. TTL and flood policies control the flow entries in P-STRIDE.

OpenFlow works well when there is no attack; however, during an attack, flow table entries in OpenFlow rise significantly. As the P-STRIDE POX controller protects network hosts during the attack, the traffic in the network utilizes 25% flow tables, as depicted in Figure 15. Timely action on the part of the P-STRIDE Controller controls the network's unnecessary traffic flow and congestion.

CPU usage refers to the business of the CPU in performing the network traffic-related issues. The more CPU Usage by any process, the lower the performance. The CPU will be busy solving the process-related task and will have less time for other processes. In the case of P-STRIDE, due to the low number of flow entries in the OpenFlow table during normal traffic, CPU usage is low, which in turn means better utilization of the CPU as reflected in Figure 16.

CPU utilization is effective in P-STRIDE even in the case of attacks. P-STRIDE does not consume all the CPU time for its operations. The CPU gets free and can execute several other processes. On the other hand, OpenFlow POX consumes almost 59% of CPU time as depicted in Figure 17 in detecting the attack and maintaining flow table entries, which badly affects the CPU's overall performance.

Network usage refers to the business of Network resources, such as paths, firewall, router, etc., in performing network traffic-related issues. The more network resources usage by any process, the lower the performance is. Network resources will be busy solving the attack-related task and have less time for other processes. In the case of P-STRIDE, due to early detection of an attack and a low number of flow entries in the OpenFlow table during normal traffic, Network usage is low, which in turn means better utilization of Network resources as reflected in Figure 18.

Network Resource utilization is effective in P-STRIDE even in the case of attacks. P-STRIDE does not consume all the network resources for its operations and activities. Network resources are easily available and can work on correct and authenticated traffic. On the other hand, OpenFlow POX consumes almost 56% of Network resources as depicted in Figure 19 in detecting the attack and routing flow table entries, which badly affects the network's overall performance.

A comparative analysis of average delay under different types of flooding attacks (TCP/SYN, UDP, and ICMP) has been conducted between OpenFlow POX and P-STRIDE. The results, presented in Table 3, demonstrate that P-STRIDE

operates in real-time detection mode, significantly reducing the average delay compared to OpenFlow POX, which incurs noticeable delays, thereby impacting network performance. Furthermore, OpenFlow POX does not address certain STRIDE threats, such as Spoofing, Repudiation, Information Disclosure, and Elevation of Privileges, whereas P-STRIDE efficiently mitigates these threats using OpenFlow table entries.

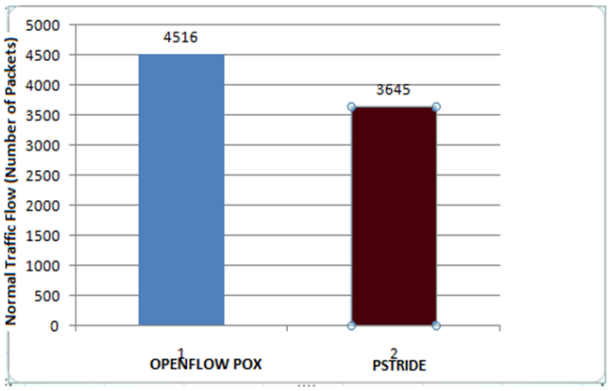


Figure 14. Average traffic flows during normal flows

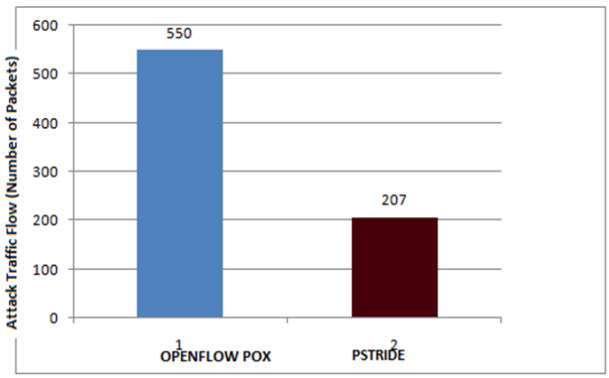


Figure 15. Average traffic flows during the attack

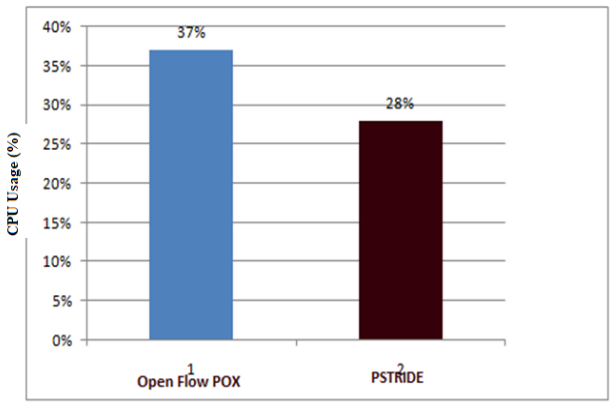


Figure 16. CPU usage for average traffic flows during normal traffic

Further, we have compared P-STRIDE with Self-Organizing Map (SOM), a widely used approach for DDoS detection that classifies attack patterns using 4-tuple and 6-tuple feature extraction from OpenFlow table entries. A dynamic adaptive behaviour to handle the adversaries is noted for cross-validation techniques in terms of 6-tuple SOM with parameters source IP, destination IP, source Port, destination port, protocol, and port size, and 4-tuple SOM with parameters source IP, destination IP, source Port, and destination port. As shown in Table 4, P-STRIDE achieves the fastest detection time (135 ms), significantly outperforming 4-tuple SOM (271 ms) and 6-tuple SOM (352 ms), demonstrating its efficiency in detecting and mitigating DDoS attacks in SDN environments.

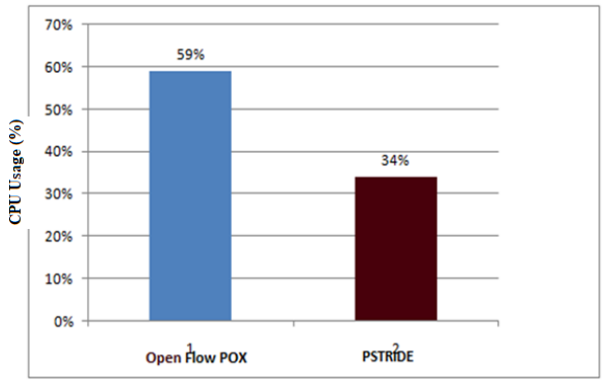


Figure 17. CPU usage for average traffic flows during the attack

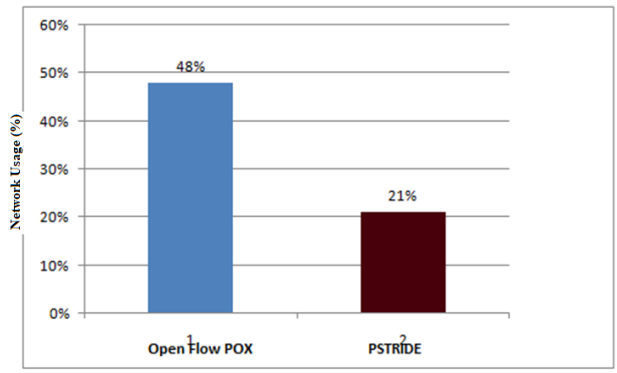


Figure 18. Network usage for average traffic flows during normal traffic

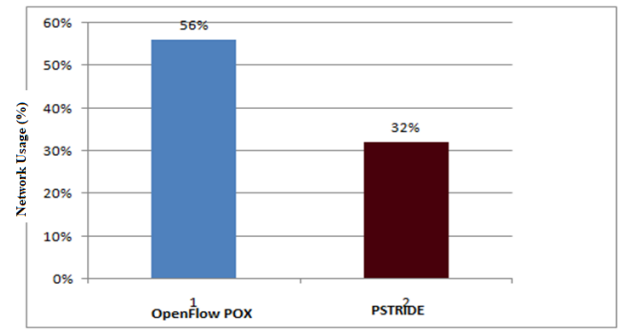


Figure 19. Network usage for average traffic flows during the attack

The more time a method takes to detect the attack, the poorer the performance in terms of utilization of network resources, CPU usage, network traffic, latency, etc. Renowned 4-tuple and 6-tuple SOM classification takes more time, i.e., 271 ms and 352 ms, respectively, compared to 135 ms in P-STRIDE as depicted in Figure 20. This detection time has improved other quality measures and has proved its suitability in terms of various STRIDE attacks.

Table 3. STRIDE threats used in existing and proposed work

Attack Types	OpenFlow	P-STRIDE
TCP/SYN flood	137612 ms	372 ms – Real-time Detection
UDP flood	52733 ms	171 ms – Real-time Detection
ICMP flood	57973 ms	182 ms – Real-time Detection
Spoofing	NA	OF Tables
Repudiation	NA	OF Tables
Information disclosure	NA	OF Tables
Elevation of privileges	NA	OF Tables

Note: Transmission Control Protocol (TCP); Synchronization (SYN); User Datagram Protocol (UDP); Internet Control Message Protocol (ICMP).

Table 4. DDOS detection time of P-STRIDE with 4-tuple and 6-tuple SOM classifications

Algorithm	Detection Time
4-tuple SOM classification	271 ms
6-tuple SOM classification	352 ms
P-STRIDE	135 ms

Note: Distributed Denial-of-Service (DDoS); Self-Organizing Map (SOM).

Table 5 defines the complete detection rate for P-STRIDE and various SOM classifiers with 3 different OpenFlow switches. The Detection Rate is a measure of the accuracy parameter, how well the method identifies the attack if it prevails. False positives give a measure when the attack was not prevailing, and it was detected, which is an undesirable property.

The performance variations observed in Table 5 are attributed to dynamic traffic conditions and attack intensity differences across experimental scenarios. Specifically, higher packet loss and delay values occur during peak attack injection intervals, reflecting transient controller congestion before mitigation rules are fully enforced. These behaviors are consistent with SDN control-plane dynamics reported in prior studies.

An important measure in choosing is how well a method can detect the attack and how well on time. It accounts for the reliability and accuracy of a method. Various OpenFlow switch levels have been created to identify the rate at which a model can identify the attack. P-STRIDE in any situation with any number of switch levels has proven to be the best choice with a 98.76% detection rate in comparison to a detection rate of 67.08% and 66.86% with 6-tuple and 4-tuple, respectively, as depicted in Figure 21.

False positives are a measure that defines the detection of an attack in the absence of an attack. The accuracy of a method. 3 OpenFlow switch levels have been created to measure the correctness of the method. P-STRIDE in any situation with any number of switch levels has proven to be the best choice, with at least 0.34% false positives compared to false positives of 0.39% and 0.41% with 6-tuple and 4-tuple, respectively, as depicted in Figure 22.

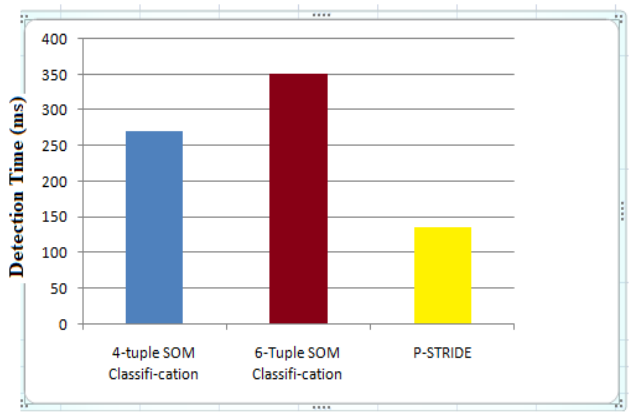


Figure 20. Distributed Denial-of-Service (DDoS) detection time

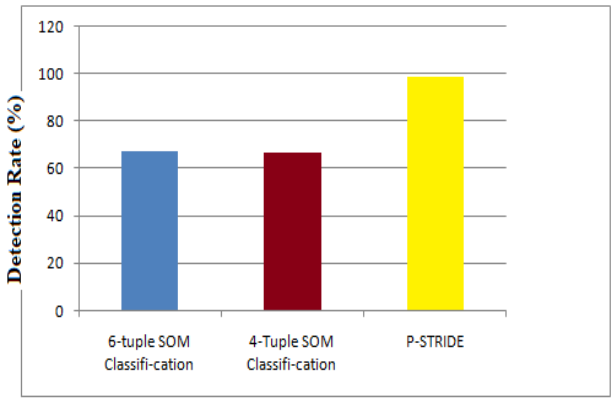


Figure 21. Distributed Denial-of-Service (DDoS) detection rate

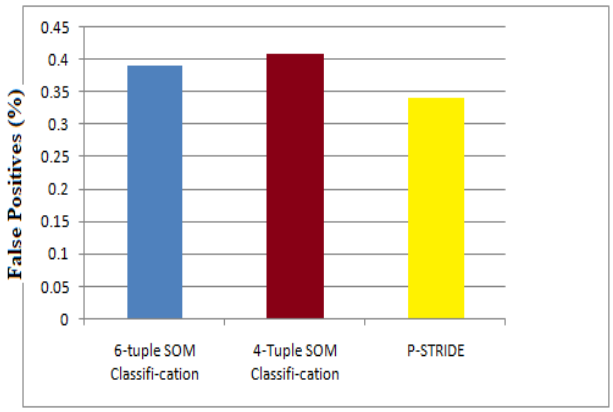


Figure 22. Distributed Denial-of-Service (DDoS) accuracy false positives

Table 5. DDoS detection time of P-STRIDE with 4-tuple and 6-tuple SOM classifications

Method	6-Tuple		4-Tuple		P-STRIDE	
	Detection Rate	False Positives	Detection Rate	False Positives	Detection Rate	False Positives
OF switch 1	98.61%	0.59%	98.57%	0.48%		
OF switch 2	99.11%	0.46%	98.73%	0.62%	98.76%	0.34%
OF switch 3	3.52%	0.12%	3.27%	0.13%		
Average	67.08%	0.39%	66.86%	0.41%	98.76%	0.34%

Note: Distributed Denial-of-Service (DDoS); Self-Organizing Map (SOM).

7. CONCLUSION AND FUTURE SCOPE

This study presents a comprehensive security framework for SDN by addressing key vulnerabilities and mitigating STRIDE threats. The proposed P-STRIDE framework effectively enhances SDN security by detecting and mitigating both known and unknown attacks, overcoming the limitations of OpenFlow-based implementations. Our experimental results demonstrate that P-STRIDE significantly reduces the impact of attacks, optimizes flow table entries and CPU usage, and maintains a minimal latency footprint on the SDN controller. Additionally, the extended TLS security framework, incorporating entropy parameters into OpenFlow, enhances network integrity, prevents unauthorized access, and detects data tampering using MD5.

Despite its advantages, P-STRIDE faces challenges in highly dynamic networks where frequent topology changes impact its adaptability. Moreover, as network complexity increases, scalability and computational overhead become critical concerns, requiring continuous monitoring and threat analysis. To address these limitations, future research will focus on integrating SDN with NFV to dynamically provision security services and reduce computational overhead. The SDN-NFV architecture is expected to enhance scalability, automation, and security in large-scale networks. Additionally, container-based virtualization using Docker will be explored to improve SDN flexibility and deployment efficiency, allowing seamless security updates and resource allocation.

Further, AI-driven threat detection techniques, including machine learning and deep learning, will be implemented to enable real-time attack detection, anomaly detection, and adaptive threat mitigation. Security challenges in agile SDN-NFV deployments, particularly in access control, security bottlenecks, and policy enforcement, will also be addressed to ensure a robust security framework. Lastly, efforts will be made to refine the P-STRIDE model to enhance its scalability, adaptability, and processing efficiency, especially in highly dynamic network environments. By addressing these challenges, future research will further strengthen SDN security, scalability, and adaptability, ensuring a more resilient and intelligent networking infrastructure.

REFERENCES

[1] Sezer, S., Scott-Hayward, S., Chouhan, P.K., Fraser, B., et al. (2013). Are we ready for SDN? Implementation challenges for software-defined networks. IEEE Communications Magazine, 51(7): 36-43. https://doi.org/10.1109/MCOM.2013.6553676

[2] Kim, H., Feamster, N. (2013). Improving network management with software defined networking. IEEE Communications Magazine, 51(2): 114-119. https://doi.org/10.1109/MCOM.2013.6461195

[3] Bhuiyan, Z.A., Islam, S., Islam, M.M., Ullah, A.A., Naz, F., Rahman, M.S. (2023). On the (in) security of the control plane of SDN architecture: A survey. IEEE Access, 11: 91550-91582. https://doi.org/10.1109/ACCESS.2023.3307467

[4] Goth, G. (2011). Software-defined networking could shake up more than packets. IEEE Internet Computing, 15(4): 6-9. https://doi.org/10.1109/MIC.2011.96

[5] Hussein, N.H., Koh, S.P., Yaw, C.T., Tiong, S.K., et al. (2024). SDN-based VANET routing: A comprehensive survey on architectures, protocols, analysis, and future challenges. IEEE Access, 13: 126801-126861. https://doi.org/10.1109/ACCESS.2024.3355313

[6] Hnamte, V., Najar, A.A., Nhung-Nguyen, H., Hussain, J., Sugali, M.N. (2024). DDoS attack detection and mitigation using deep neural network in SDN environment. Computers & Security, 138: 103661. https://doi.org/10.1016/j.cose.2023.103661

[7] Yue, M., Yan, Q., Lu, Z., Wu, Z. (2024). CCS: A cross-plane collaboration strategy to defend against ldos attacks in SDN. IEEE Transactions on Network and Service Management, 21(3): 3522-3536. https://doi.org/10.1109/TNSM.2024.3363490

[8] Maleh, Y., Qasmaoui, Y., El Gholami, K., Sadqi, Y., Mounir, S. (2023). A comprehensive survey on SDN security: Threats, mitigations, and future directions. Journal of Reliable Intelligent Environments, 9(2): 201-239. https://doi.org/10.1007/s40860-022-00171-8

[9] Kaur, S., Singh, J., Ghumman, N.S. (2014). Network programmability using POX controller. In ICCCS International Conference on Communication, Computing & Systems, pp. 134-138. https://sbsstc.ac.in/icccs2014/Papers/Paper28.pdf.

[10] Schehlmann, L., Abt, S., Baier, H. (2014). Blessing or curse? Revisiting security aspects of Software-Defined Networking. In 10th International Conference on Network and Service Management (CNSM) and Workshop, Janeiro, Brazil, pp. 382-387. https://doi.org/10.1109/CNSM.2014.7014199

[11] Balarezo, J.F., Wang, S., Chavez, K.G., Al-Hourani, A., Kandeepan, S. (2022). A survey on DoS/DDoS attacks mathematical modelling for traditional, SDN and virtual networks. Engineering Science and Technology, an International Journal, 31: 101065. https://doi.org/10.1016/j.jestch.2021.09.011

[12] Ahmad, I., Namal, S., Ylianttila, M., Gurtov, A. (2015). Security in software defined networks: A survey. IEEE Communications Surveys & Tutorials, 17(4): 2317-2346. https://doi.org/10.1109/COMST.2015.2474118

[13] Braga, R., Mota, E., Passito, A. (2010). Lightweight DDoS flooding attack detection using NOX/OpenFlow. In IEEE Local Computer Network Conference, Denver, CO, USA, pp. 408-415. https://doi.org/10.1109/LCN.2010.5735752

[14] Wang, R., Jia, Z., Ju, L. (2015). An entropy-based distributed DDoS detection mechanism in software-defined networking. In 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, pp. 310-317. https://doi.org/10.1109/Trustcom.2015.389

[15] Lim, S., Ha, J., Kim, H., Kim, Y., Yang, S. (2014). A SDN-oriented DDoS blocking scheme for botnet-based attacks. In 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), Shanghai, China, pp. 63-68. https://doi.org/10.1109/ICUFN.2014.6876752

[16] Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., Maglaris, V. (2014). Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. Computer Networks, 62: 122-136. https://doi.org/10.1016/j.bjp.2013.10.014

[17] Dayal, N., Srivastava, S. (2017). Analyzing behavior of DDoS attacks to identify DDoS detection features in

SDN. In 2017 9th International Conference on Communication Systems and Networks (COMSNETS), Bengaluru, India, pp. 274-281. https://doi.org/10.1109/COMSNETS.2017.7945387

[18] Liu, J., Lai, Y., Zhang, S. (2017). FL-GUARD: A detection and defense system for DDoS attack in SDN. In Proceedings of the 2017 International Conference on Cryptography, Security and Privacy, Wuhan, China, pp. 107-111. https://doi.org/10.1145/3058060.3058074

[19] Guodong, T., Xi, Q., Chaowen, C. (2017). A SDN security control forwarding mechanism based on cipher identification. In 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, China, pp. 1419-1425. https://doi.org/10.1109/ICCSN.2017.8230343

[20] Aladaileh, M.A., Anbar, M., Hasbullah, I.H., Chong, Y.W., Sanjalawe, Y.K. (2020). Detection techniques of distributed denial of service attacks on software-defined networking controller–A review. IEEE Access, 8: 143985-143995. https://doi.org/10.1109/ACCESS.2020.3013998

[21] Hadi, F., Imran, M., Durad, M.H., Waris, M. (2018). A simple security policy enforcement system for an institution using SDN controller. In 2018 15th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, pp. 489-494. https://doi.org/10.1109/IBCAST.2018.8312269

[22] Ilyas, Q., Khondoker, R. (2018). Security analysis of floodlight, zeroSDN, beacon and POX SDN controllers. In SDN and NFV Security: Security Analysis of Software-Defined Networking and Network Function Virtualization, pp. 85-98. https://doi.org/10.1007/978-3-319-71761-6_6

[23] Salau, A.O., Beyene, M.M. (2024). Software defined networking based network traffic classification using machine learning techniques. Scientific Reports, 14(1): 20060. https://doi.org/10.1038/s41598-024-70983-6

[24] Shalimov, A., Zuikov, D., Zimarina, D., Pashkov, V., Smeliansky, R. (2013). Advanced study of SDN/OpenFlow controllers. In Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia, Moscow, Russia, pp. 1-6. https://doi.org/10.1145/2556610.2556621

[25] Tirumala, A., Dunigan, T., Cottrell, L. (2003). Measuring end-to-end bandwidth with Iperf using Web100. In Presented at Passive and Active Monitoring Workshop (PAM 2003), San Diego, CA, USA. https://www.slac.stanford.edu/pubs/slacpubs/9500/slac-pub-9733.pdf.

[26] De Oliveira, R.L.S., Schweitzer, C.M., Shinoda, A.A., Prete, L.R. (2014). Using mininet for emulation and prototyping software-defined networks. In 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, Colombia, pp. 1-6. https://doi.org/10.1109/ColComCon.2014.6860404

[27] Midha, S., Tripathi, K. (2019). Extended security in heterogeneous distributed SDN architecture. In International Conference on Advanced Communication and Computational Technology, pp. 991-1002. https://doi.org/10.1007/978-981-15-5341-7_75

[28] Salau, A.O., Marriwala, N., Athaee, M. (2020). Data security in wireless sensor networks: Attacks and countermeasures. In Mobile Radio Communications and 5G Networks: Proceedings of MRCN 2020, pp. 173-186. https://doi.org/10.1007/978-981-15-7130-5_13

[29] Salau, A.O., Assegie, T.A., Akindadelo, A.T., Eneh, J.N. (2023). Evaluation of Bernoulli Naive Bayes model for detection of distributed denial of service attacks. Bulletin of Electrical Engineering and Informatics, 12(2): 1203-1208. https://doi.org/10.11591/eei.v12i2.4020