TPAAD: two-phase authentication system for denial of service attack detection and mitigation using machine learning in software-defined network.

NISA, N., KHAN, A.S., AHMAD, Z. and ABDULLAH, J.

2024

Received: 23 June 2023 | Revised: 2 November 2023 | Accepted: 11 December 2023

DOI: 10.1002/nem.2258

RESEARCH ARTICLE

WILEY

TPAAD: Two-phase authentication system for denial of service attack detection and mitigation using machine learning in software-defined network

Najmun Nisa[1,2] | Adnan Shahid Khan[1] | Zeeshan Ahmad[3] | Johari Abdullah[1]

[1]Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Kota Samarahan, Malaysia

[2]Computer Science Department, COMSATS University Islamabad, Islamabad, Pakistan

[3]National Subsea Centre, Robert Gordon University, Aberdeen, UK

Correspondence
Adnan Shahid Khan and Najmun Nisa, Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak, Kota Samarahan 94300, Malaysia.
Email: skadnan@unimas.my and najam_nisa@comsats.edu.pk

Funding information
Universiti Malaysia Sarawak, Kota Samarahan, Malaysia, Grant/Award Number: F08/PGRG/2058/2021

Abstract
Software-defined networking (SDN) has received considerable attention and adoption owing to its inherent advantages, such as enhanced scalability, increased adaptability, and the ability to exercise centralized control. However, the control plane of the system is vulnerable to denial-of-service (DoS) attacks, which are a primary focus for attackers. These attacks have the potential to result in substantial delays and packet loss. In this study, we present a novel system called Two-Phase Authentication for Attack Detection that aims to enhance the security of SDN by mitigating DoS attacks. The methodology utilized in our study involves the implementation of packet filtration and machine learning classification techniques, which are subsequently followed by the targeted restriction of malevolent network traffic. Instead of completely deactivating the host, the emphasis lies on preventing harmful communication. Support vector machine and K-nearest neighbours algorithms were utilized for efficient detection on the CICDoS 2017 dataset. The deployed model was utilized within an environment designed for the identification of threats in SDN. Based on the observations of the banned queue, our system allows a host to reconnect when it is no longer contributing to malicious traffic. The experiments were run on a VMware Ubuntu, and an SDN environment was created using Mininet and the RYU controller. The results of the tests demonstrated enhanced performance in various aspects, including the reduction of false positives, the minimization of central processing unit utilization and control channel bandwidth consumption, the improvement of packet delivery ratio, and the decrease in the number of flow requests submitted to the controller. These results confirm that our Two-Phase Authentication for Attack Detection architecture identifies and mitigates SDN DoS attacks with low overhead.

KEYWORDS
attack detection, attack mitigation, CICDoS 2017 dataset, denial-of-service attacks (DoS), distributed denial-of-service (DDoS) attacks, KNN, machine learning (ML), Mininet, open flow, RYU controller, software-defined networking (SDN), SVM, Two-Phase Authentication of Attack Detection

1 | INTRODUCTION

Due to the inherent complexity and limitations of the architecture, as well as the challenges of adapting to changing market demands, traditional network designs pose management challenges. The advent of the Internet has significantly transformed the landscape of communication and computing technology.[1,2] As the number of technologies incorporated into a typical network increase, the task of addressing emerging requirements such as scalability, security, flexibility, dependability, and reliability becomes increasingly complex.[3,4] The Internet of Things (IoT) refers to a network of interconnected devices, such as sensors, that are capable of collecting and transmitting data autonomously, without the need for human involvement, within the context of software-defined networking (SDN). The IoT has brought about significant transformations in communication. Its incorporation into computer networks, computing environments, smart gadgets, and healthcare technologies has resulted in enhanced personalized and efficient care. Data security and privacy are significant issues, and there is a lack of standardized data protocols in many IoT devices, highlighting the importance of ensuring security in SDN architecture.[5–7]

The challenges associated with administration and operation of conventional network technologies have prompted the emergence of SDNs. In traditional network architectures, distinct network components are responsible for managing the control plane and data plane functions. In contrast, SDN allocate separate network entities to individual operations.[8] The control plane, alternatively referred to as the controllers, has responsibility for all network operations, while the data plane only concentrates on the transmission of traffic based on instructions received from the control plane. Interfaces play a crucial role in facilitating the exchange of information and promoting effective communication across the realms of data, control, and application planes.[8–10] The SDN architecture has three interfaces: the eastbound/westbound interface, which establishes connections between distributed controllers; the southbound interface, which links the control plane to the data plane; and the northbound interface, which connects the control plane to the application plane.[3,11] Typically, the establishment of communication between the controller and switches is facilitated using a standardized protocol known as OpenFlow.[12]

The research group known as Cleanslate at Stanford University introduced OpenFlow, which has now become the prevailing industry standard for SDN. The widespread adoption of OpenFlow in both academic and industrial settings has contributed to the remarkable success of this SDN standard.[13] An OpenFlow Controller is a communication device that facilitates communication between underlying switches by adhering to the standards of the OpenFlow protocol. Typically, an OpenFlow Controller refers to a software application with elevated functionality that governs numerous OpenFlow logical switches.[12] An OpenFlow channel serves as a communication interface connecting an OpenFlow switch and an OpenFlow controller, enabling the controller to effectively manage the switches.[14] The utilization of the OpenFlow protocol has the potential to provide efficient communication between switches and controllers. The controller is responsible for making the decision of either discarding the packet or proceeding with its transmission to the intended destination. The task of processing incoming packets and updating the network switches with the most recent set of rules is undertaken by this entity.[4,15]

1.1 | Overview of SDN workflow

During communication, there is a lot of extra work involved in passing information between the control plane and the data plane, and it might slow down the whole network if it is not handled properly.[12,16] The controller, which is located in the control plane of an OpenFlow network, is responsible for governing the behaviour of the entire network. This is accomplished in one of two ways: by establishing flow rules on the data plane. These two primary ways to configure rules at switches are the proactive and reactive modes. During proactive network start-up, the controller first converts network policies into flow rules, which are then installed on the network's switches. When operating in the reactive mode, the controller does not compute or install new rules unless one of the switches makes an explicit request. So, it is abundantly clear that switches can respond rapidly to changes in the network by operating in reactive mode, which eliminates the need for massive flow tables.[17,18]

The installation of reactive rules, on the other hand, leaves SDN controllers and switches open to denial-of-service (DoS) attacks. An attacker can take control of compromised hosts and flood the network with transient faked flows, causing switches to repeatedly poll the controller.[17] In the reactive method, whenever an OpenFlow switch gets numerous packets, then it will queue these packets in an input queue and then proceeds to perform the various steps to process each packet in a First-In–First-Out way and this are done because the reactive method is considered to be the more efficient method.[18,19] In particular, if an Internet Protocol (IP) address in an incoming packet matches one in the local flow table, the packet is forwarded according to the corresponding flow forwarding rule. In this scenario, the entry is considered to be a table hit, and the packet is processed or forwarded in accordance with the rule that was installed. On the other hand, if the switch encounters a table miss, it will initiate a lookup request to the SDN controller, perhaps through an OpenFlow Packet-In message, if no match is discovered.[20–23] After that, the SDN controller will examine these search queries and install the proper forwarding flow rules among all networking devices/switches along the direction that this packet can flow, such as by using OpenFlow messages. It is important to keep in mind that the SDN controller has the capability of installing rules that cause these particular types of packets to be dropped as shown in Figure 1. The OpenFlow switch searches the flow table for flow rules matching the packet header. Based on the action field of the flow rule, the switch processes the packet. When this is not the case, the switch stores the packet in a buffer, and its header is encapsulated in a packet_in message and alerts the controller. Upon receiving a packet message, the controller processes it according to the control application's rules. The switch will get a packet-out message with the "action" when the packet is sent out. After that, the OpenFlow switch will process the buffered table-miss packet in accordance with the "action" received from the controller. In addition, the controller can set up flow rules with "match" and "action" attributes, telling the switch how to handle packets of the same flow upon their subsequent reception.[8,11,15,24–27]

1.2 | Problem statement

Despite its advantages over traditional networks, such as centralized monitoring and granular management, SDN is less secure due to its increased vulnerability to attacks and the introduction of new security concerns.[28] SDN can be compromised in a variety of ways, such as by spoofing,[29] tampering, repudiation, information leakage, and DoS.[30–32]

Attacks using DoS could also be launched against the SDN. DoS attacks take place whenever a host attempts to take control of one system by flooding it with excessive traffic. The primary objective of this attack is to reduce

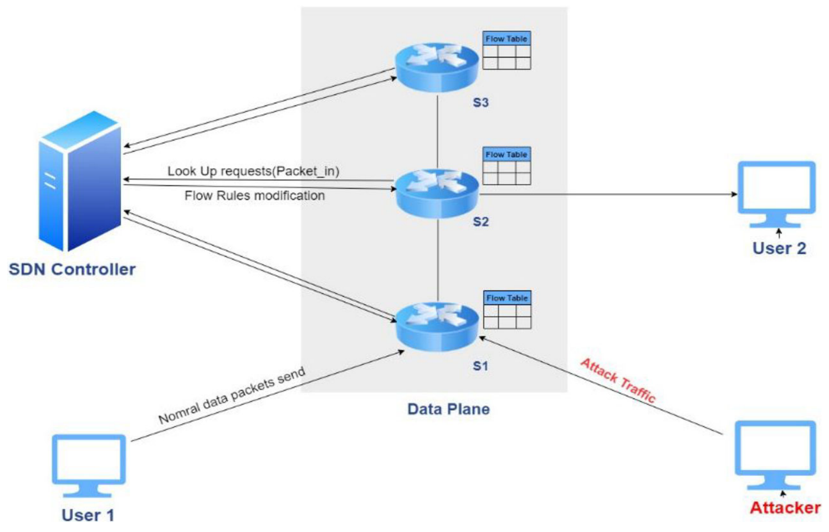

FIGURE 1 Software-defined networking (SDN) block diagram.

the amount of time that the system is available to users and to prevent those users from accessing any available services. Distributed DoS (DDoS) attacks are those in which the attackers use a large number of hosts rather than just one to attack a system that acts as the controller. And when an SDN controller is hit by a DoS/DDoS attack, it might cause serious problems.

Because DoS attacks come in numerous shapes and sizes and may affect so many areas of an SDN's infrastructure, they are notoriously tricky to identify and counter. Because of this, they are widely recognized as among the most difficult forms of cyberattacks.[4,33] A network attack can take one of two forms: either the attacker sends a huge number of little packets to many hosts on the network, thereby consuming their bandwidth, or the attacker sends a flood of packets to many hosts, so consuming the memory and processing capacity of the switch and the controller. The attacker has a choice between the two. An authorized user may need to repeatedly scan the entire network or send a sizable volume of data to a large number of nodes.[16] Network activity includes both of these examples. This makes it challenging for a mitigation method to differentiate between an attacker and a legitimate user.[18,34–36]

An SDN-based system may readily be subjected to a DoS attack by flooding the OpenFlow switch with packets having random destinations. In search of a new flow rule, every packet is now transmitted from the switch to the controller. By using their resources, such an attack might affect several parts of the SDN architecture, such as switch storage, control channel bandwidth, and central processing unit (CPU) usage. This resource utilization may lead to performance concerns, increased latency, packet loss, and reduced accuracy in detecting attacks. When SDN is utilized, DoS attack may not completely halt the functioning of the network or its components, but it has the potential to slow down data transmission due to an overabundance of network resources.

In this research, we proposed a system called Two-Phase Authentication of Attack Detection (TPAAD) that mitigates the impact of DoS attacks on SDN while conserving network resources including bandwidth, computation, and flow table space.

The subsequent sections of the paper are structured in the following manner: Section 2 provides a comprehensive analysis of the strategies employed to mitigate DoS attacks within the context of SDN. Section 3 provides a thorough explanation of the proposed TPAAD system. In Section 4, an evaluation of the performance of the suggested system will be conducted by a comparative analysis with certain benchmarks that are now in existence. In Section 5 of our article, we present our concluding remarks and provide suggestions for further research.

2 | RELATED WORK

This section provides a comprehensive overview of several techniques employed over the past few years to protect SDN from DoS attacks. For SDN DoS attack detection, machine learning algorithms provide convincing performance. The attack on the SDN controller's control and data planes can be efficiently detected using machine learning approaches.

FloodDefender[18] is a scalable and protocol-independent system for OpenFlow networks. It is located in the centre of the controller platform and other controller programs, ensuring that both comply with the OpenFlow rules without the need for any additional hardware. New frequency features are utilized by FloodDefender's detection module to enable accurate identification of SDN-targeted DoS attacks. The mitigation module employs three strategies to efficiently mitigate attack traffic: table-miss engineering to avoid exhausting the communication overhead, packet filtration to isolate the malicious requests and conserve the control plane's computation power, and flow rule strategic planning to get rid of most of the redundant incoming packets in the flow table. Additionally, a prototype is implemented to assess FloodDefender's functionality in both software and hardware settings, and a queueing delay model is employed to determine the optimal number of neighbour switches to be used in the table-miss engineering. However, when the attack rate is high, FloodDefender may discard benign packets, and communication between two planes may slow down the overall network.

Mousavi and St-Hilaire[37] introduce an entropy-based method for preventing DoS attacks in SDN with centralized management. The procedure is explained in depth below. Monitoring for entropy changes at an IP address can help detect DoS attacks. Variation in the incoming data packet request is one way to detect DoS attempts. Entropy rises as uncertainty rises and vice versa. Both the window's size and the threshold's setting control how much entropy is produced. The barrier point will always be set at 50; however, the height of the barrier will vary depending on the dimensions of the window. Using the packet's IP address, the entropy of an incoming packet can be calculated. When the entropy value of such an attack reaches a specified requirement or threshold, it is considered a DoS attack. On the other hand, looking at a temporal range can improve precision and yield better outcomes. However, this approach can only be used on centralized SDNs and a single host at a time.

Avant-Guard[38] uses the Transmission Control Protocol (TCP) handshake method to confirm where the attack is coming from. This makes DoS attacks on open flow (OF) switches less effective. Once validation is complete, the controller will set up a flow rule to begin the data transmission. However, in this approach, only TCP data may be processed using this method, and the OF switches will need to be updated. To avoid both planes' saturation, FloodGuard[39] employs a proactive flow rule analyser and packet migration. When FloodGuard detects an overloading attack, it will send any packets that failed to be processed because of an OpenFlow table miss to the data plane cache. The Analyser module helps to keep an eye on the values of sensitive variables in programs or applications that are currently running. These values are then turned into path conditions, and the flow rules are set up as proactive ones in OF switches. Then, Packet-In message notification will be sent to the controller by the data plane cache. However, both of the above methods focus on protecting against DDoS attacks directed against SDN. When compared with attack prevention, the importance of detection is often overlooked. Furthermore, both methods employ extra hardware, like in Avant-Guard (TCP-Proxy) and in FloodGuard (data plane cache), that is incompatible with the standard OpenFlow protocol.

In order to safeguard the controller from DoS attacks, Kandoi and Antikainen[40] implemented a mechanism to restrict the pace at which packets are transmitted towards the controller. The authors proposed the utilization of flow aggregation as a viable solution to mitigate the issue of flow table overflow. Additionally, they suggested the implementation of an optimal timeout value for flow rules to effectively address this concern. While this method provides assistance, it is important to acknowledge the potential for loss of routine traffic packets. Belyaev[41] introduced a load-balancing strategy to enhance the overall survival of SDN by dividing the congested lines with other routes between switches. This was done to maximize the amount of traffic that may pass through the network. To put this strategy into action, there needs to be an alternative route that can lighten the load. Oktian et al.[42] developed a program that keeps the media access control and IP addresses of connecting hosts in a binding table so that he can become completely aware of their location. When a person leaves or joins a host, the information about that host is updated accordingly. For DoS protection, the program gathers three OpenFlow messages. Each procedure will separately and jointly generate a solution based on different attributes, all of which will be used to identify and react to the messages. At this time, many security techniques, such as firewalls, authentication systems, various encryption methods, and various antiviruses, are utilized in order to shield sensitive data from the possibility of being compromised by a security breach.[10,43–45]

The network was protected from DoS attacks by Shoeb and Chithralekha[46] using a trust-based method. For each new packet, an IP-based trust value was applied based on the packet's communication history. A parallel flow installation paradigm was suggested by Imran et al.[47] as a way to make DoS attacks on the controller and the control channel less harmful. Upon receiving the flow request, this model will install flow rules on all switches between the source and the destination. For preventing DDoS attacks, Cui et al.[48] created SD-Anti-DDoS, which has four parts (attack trigger, detection, traceback, and mitigation). By using a backpropagation neural network, we can identify the attack, track it to its origin, and then prevent it by setting up flow rules. However, this mechanism will affect performance while using different OF protocols. Packet-In messages entering from several switches are queued to prevent DoS attacks via multilayer queuing.[17] Requests are retrieved from the queues using a weighted round-robin algorithm. Depending on the number of ports, a switch's queue can be broken down into separate queues. It is also possible to combine the queues of several switches into a single queue.

Peng et al.[49] demonstrated a DDoS attack detection system for SDN that relies on centralized management. Preprocessing and anomaly detection modules are what make up this component. The flow feature vectors are standardized and normalized by the preprocessing function, and the anomaly detection method then labels whether they are legitimate or normal. An approach that is based on statistical analysis and machine learning is proposed in Banitalebi Dehkordi et al.[50] K-means and K-nearest neighbours (KNN) are merged into a single machine learning technique to detect malicious flows by relying on their asymmetrical nature, and high rates of change are suggested in Tan et al.[51] A study of DDoS detection using a support vector machine (SVM) in a SDN can be found in Ye et al.[52] They used the SVM technique to evaluate the traffic and detect DDoS attacks based on the derived six-tuple feature values. Accurately classifying DDoS attacks based on incomplete information is essential for detection. Locating the source of the controller attack is essential. The SVM classifier is popular because of its high accuracy and low false-positive rate for detecting DDoS attacks.[53] Solutions for early detection of DDoS attacks utilizing entropy to ascertain the degree of unpredictability in flow data are provided in the literature.[54,55] But because threshold values do not consider the potential variances, Revathi et al.[56] propose a solution with principal component analysis, which, from the collected information, provides new models that allow predicting the attack.

Cybersecurity is a constantly developing topic, and the collective future orientation of scholars, programmers, and security professionals needs to be in line with the novel proposals put forth by major authors in the field. The integration of artificial intelligence and machine learning with cyber threat analysis has been emphasized by various writers, who argue that this convergence will have a significant impact on the identification of attacks and will greatly influence the future of this field. Our ambition involves leveraging artificial intelligence-powered algorithms to independently identify and categorize risks with exceptional precision and efficiency. These algorithms will continue to improve themselves through a process called reinforcement learning, thereby responding to new types of attacks and machine learning strategies.[5–7,57–59]

All of the aforementioned studies are connected in some way, and they all offer partial answers to the problem of detecting and preventing DoS attacks. We find that DoS attacks in SDN may be identified and mitigated by different methods, including the inclusion of hardware requirements, packet drop rate, the blocking of malicious traffic at the port and switch levels, false alerts, detection accuracy, and computational overhead. In the event of serious attacks, however, there will be a significant delay for flows that are benign. In addition, each method relies on extra network resources while ignoring the significance of attack detection and mitigation.

3 | PROPOSED METHODOLOGY

Our proposed solution is based on detection and mitigation modules as shown in Figure 2, which will be on a control layer that can simulate the impact of DoS attacks on networks that are controlled by SDN.

Initially, the attack detection module of TPAAD will constantly observe the network's current state to identify any signs of impending attacks. It employs a two-phase authentication strategy for detecting attacks. The identification of attacks in progress is required for the mitigation module to be activated.

3.1 | Detection module

In the default configuration, a single First-In–First-Out buffer is used, and it does not differentiate between legitimate and malicious requests that are transmitted from the data plane switches. An efficient method for configuring the SDN controller and improving operational efficiency is developed to solve this. As shown in Figure 2, this is accomplished by integrating the detection module with the SDN RYU controller, where our detection program is used to detect DoS attacks. TPAAD is the proposed methodology that is utilized by the detection module. To identify abnormal traffic by analysing packet flows, it first applies the packet filtration technique. These discovered abnormal traffic flows are then sent to classifier models powered by machine learning for effective identification and management of harmful attacks. To reduce the load on the control plane and maintain the available communication bandwidth, the system employs SVM and KNN models that are very accurate at identifying between malicious and benign packet flows (Figure 3).

Our detection module comprises a TPAAD as part of its functionality. The detection module operates as follows:

- The sender launches a login procedure to enter the network and then sends a packet of data to the SDN switch. The switch determines if the packet is new.
- To control which data packets are accepted or rejected at a network interface, packet filtration is used to examine their source and destination addresses and associated ports. This procedure involves inspecting the flow table.
- A switch will check the IP address of an incoming data packet against the entries in its flow table. With the use of a set of predetermined flow rules, this comparison aims to determine if a match exists.
- If the switch detects a match (a “table hit”), it will process or route the packet in accordance with the associated rule.
- If, on the other hand, the switch experiences a “table miss,” where no matching entry is located in the flow table, the packet is instead kept in a buffer until the issue can be resolved.

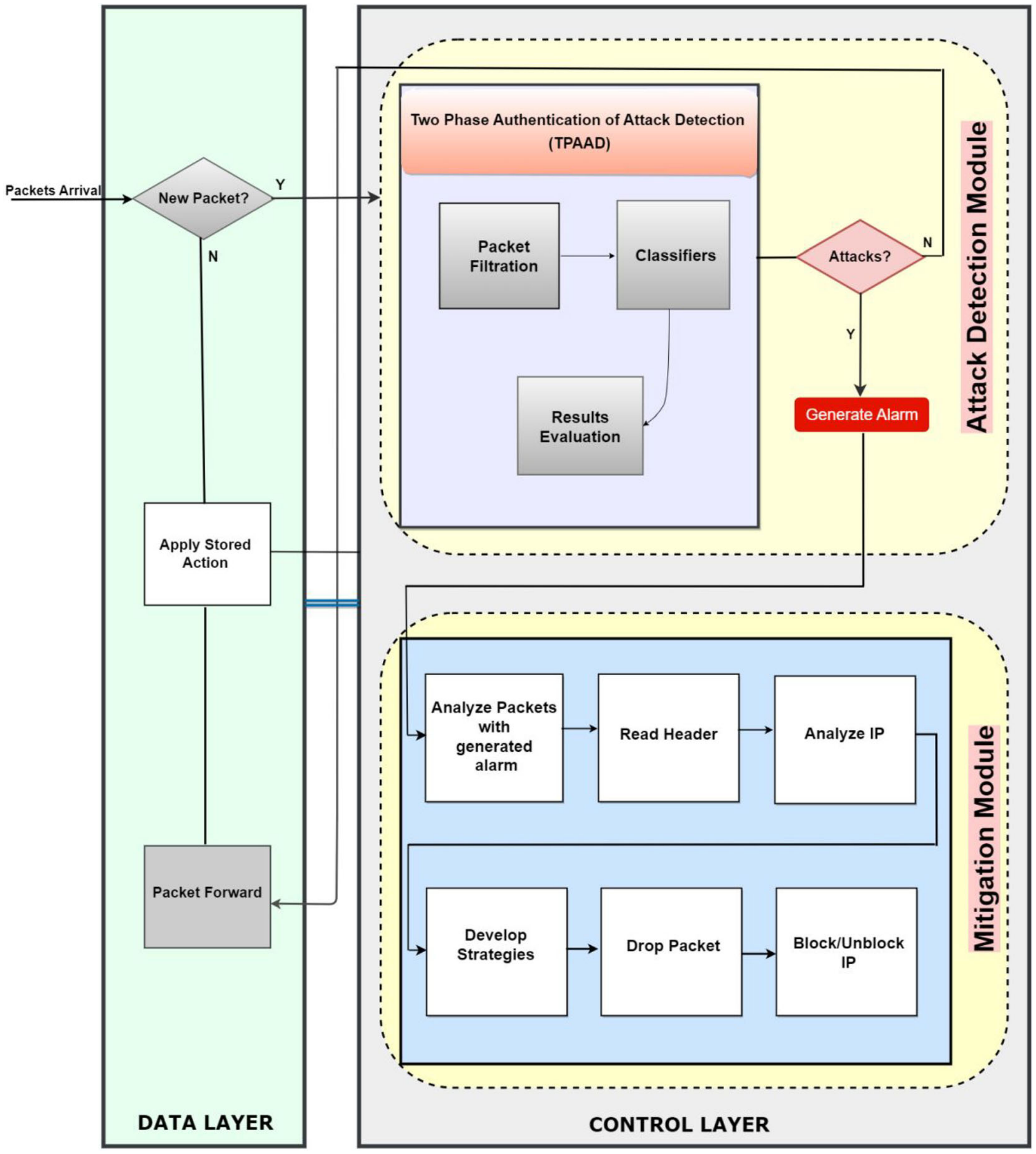

FIGURE 2 Proposed solution for detection and mitigation of denial-of-service attack in software-defined networking. IP, Internet Protocol.

- If the packet's source IP address is found to match a value in the banned list (its list contains IP addresses from where harmful packets coming again and again), the packet is dropped and the source IP address and other data are recorded for auditing reasons.
- Each request will only be executed up to a maximum of four times from the same IP address.
- Attack traffic is identified based on the rate of flow entries. The rate at which a flow enters the switch is proportional to the number of packets it sends and receives. Attack traffic will have a low frequency because its volume is intended to overwhelm the target. The opposite is true for normal flows, which are predicted to occur regularly.
- If a packet from the same IP address exceeds the threshold values for frequency and duration, the counter is increased. The packet is then sent to the harmful queue, where a classifier model is used to determine whether or not the anomaly is indeed malicious. In this way, the best possible assessment outcomes may be implemented.
- We use the CICDoS 2017 dataset to help with attack identification and categorization. Unfortunately, there are many types of information in this dataset that cannot be used in a machine learning algorithm. Preprocessing

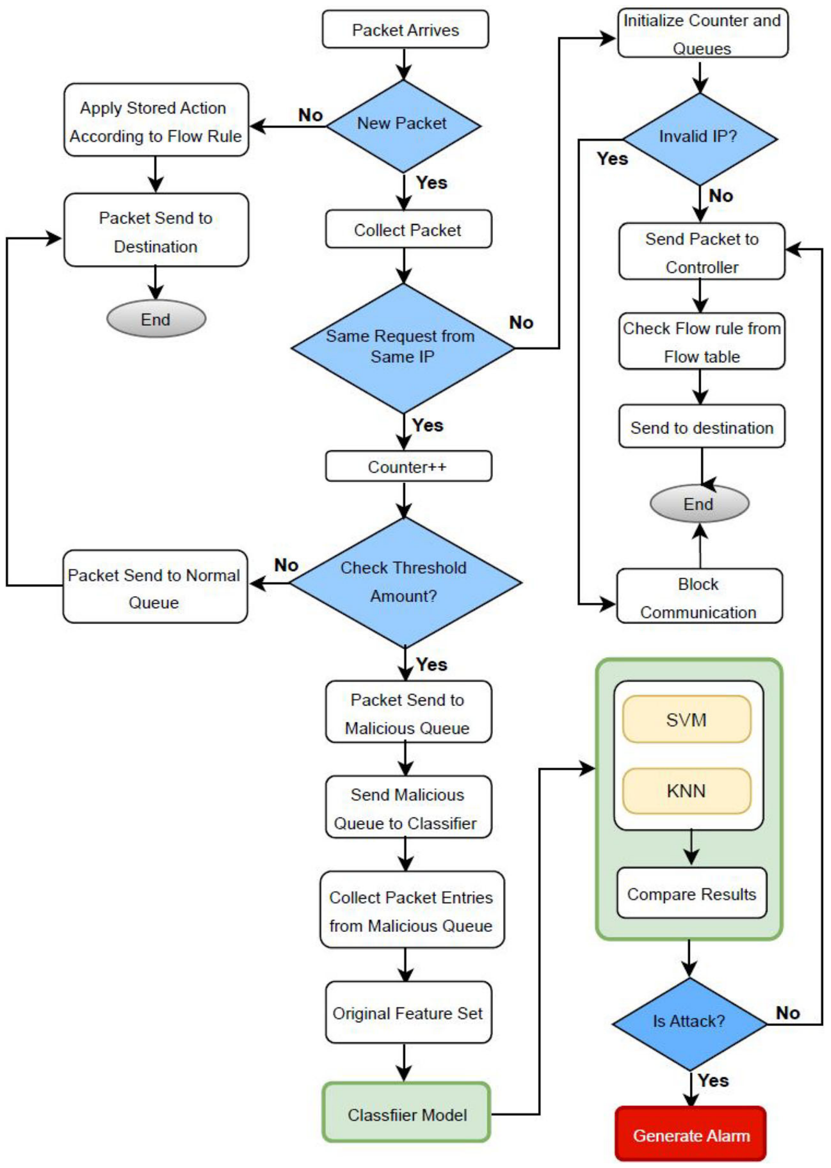
Packet Arrives
New Packet
No
Apply Stored Action According to Flow Rule
Packet Send to Destination
End
Yes
Collect Packet
Same Request from Same IP
No
Yes
Counter++
Check Threshold Amount?
No
Packet Send to Normal Queue
Yes
Packet Send to Malicious Queue
Send Malicious Queue to Classifier
Collect Packet Entries from Malicious Queue
Original Feature Set
Classfiier Model
Initialize Counter and Queues
Invalid IP?
Yes
No
Send Packet to Controller
Check Flow rule from Flow table
Send to destination
End
Block Communication
SVM
KNN
Compare Results
Is Attack?
No
Yes
Generate Alarm

FIGURE 3 Flow chart of Two-Phase Authentication of Attack Detection. IP, Internet Protocol; KNN, K-nearest neighbours; SVM, support vector machine.

the CICDoS 2017 dataset is required to remove these superfluous features before the machine learning model can be trained.

- The CICDoS 2017 dataset undergoes a data cleaning/preprocessing step to remove inconsistencies and duplicates. To improve the efficiency of the machine learning model, it is necessary to delete columns that contain null or empty values. If you want to get reliable results from your analysis, you must deal with the missing and incomplete data in your dataset. The CICDoS 2017 dataset was found to have several characteristics (columns) with missing values, which might have an undesirable effect on the machine learning model.
- Model accuracy, training speed, overfitting, and data visualization may all be improved with careful feature selection. A filter selection strategy is used to make the most of available features and cut down on unnecessary steps. This technique not only reduces the size of the dataset but also finds the best feature sets to use in detecting attacks.
- DoS attacks in the malicious queue are identified using SVM and KNN models. To effectively detect DoS attacks, these models require both the malicious queue and the dataset.
- The best attack identification results are chosen after comparing the two models' outputs. Once an issue has been identified, the controller will issue an alarm to alert the mitigation module, which will then take whatever steps are necessary.

- Alternatively, packets in the malicious queue can be sent to the controller, where they can be sent to their destination if they fit the rules in the flow table.

3.2 | TPAAD functions

3.2.1 | Packet filtration in attack detection

The filtration function sets up a number of different queues and variables. It accepts incoming packets containing switch IDs and IP addresses as input. The method retrieves packets from the buffer and determines if they have the same IP address as the packet that came before them. If they do, a counter is increased, and the packet's timing and frequency are noted. Communication with the IP is blocked, it is placed on the banned list, and the packet and source IP are noted if the packet originates from an invalid IP.

When an IP packet is legitimate, it is delivered to the normal queue and then to the controller for further processing. The packet is added to the malicious queue and the controller is alerted of a potential attack from that IP if the timing and frequency of the packet exceed a certain threshold. In any other case, a typical packet from that IP is reported to the controller. The classifier's approach is used to handle the malicious queue of abnormal packets that the function outputs. Packet filtration process is detailed in Algorithm 1.

Algorithm 1: Packet Filtration in Attack Detection (Anomaly Detection in TPAAD)

```
Input:  Incoming new packet, switch Id, IP address
Output:  Abnormal packets in the malicious queue
1.  while true do
2.  Initialize counter, time, frequency, threshold amount, malicious queue, normal queue, Banned list
3.  Collect packet from buffer
4.  If packet from same IP then
5.      Increment counter
6.      Record time and frequency
7.  else
8.      If invalid IP then
9.          Block communication
10.         Add that IP to Banned list
11.         Log packet and source IP
12.     else
13.         Send packet to normal queue and then to controller for action.
14.         Packet sends to destination.
15.     end if
16. end if
17. If packet time, frequency > time, frequency threshold amount then
18.         Send this abnormal packet to malicious queue
19.         return malicious queue
20. else
21.         Send packet to normal queue and then to controller for action.
22.         Packet sends to destination.
23. end if
24. end while
```

3.2.2 | Classifiers for attack detection in TPAAD

The function accepts as inputs malicious queue packets and a dataset. The process begins by importing and preparing the dataset. Next, a method for selecting the most pertinent features for attack detection is applied. Using the selected feature set, packet headers and network traffic are calculated to extract features. These extracted features are then fed to classifier models, and the model with the highest performance is selected. If the model detects an attack, the corresponding IP address is added to a banned list, the packet and source IP are logged, an alarm is generated, and the packet is sent for mitigation. Alternatively, if the model determines that the packet is normal, it is sent to the normal queue and then to the controller for further processing. Algorithm 2 provides the main operations performed for classifiers for attack detection process.

Algorithm 2: Attack Detection Module (Classifiers in TPAAD)

```
Input:    Malicious queue packets with features, Dataset
Output:   DoS Attack Detected
1.  Provide dataset
2.  Pre-process dataset
3.  while true do     // Apply Feature selection method
4.     for each feature set(i) do
5.        obtain features using packet headers and network traffic by performing calculations
6.     end for
7.     Input features set to classifier models
8.     Choose best results
9.     If Model detects Attack then
10.        Add that IP to Banned list
11.        Log packet and source IP
12.        Generate Alarm
13.        Send packet for mitigation module
14.    else
15.        Send packet to normal queue and than to controller for action
16.        Packet send to destination
17. end while
```

3.3 | Mitigation module

To protect the SDN against DoS attacks, we will use a mitigation module. The mitigation unit uses a cutting-edge tunnelling method to effectively manage attack traffic, relieving the controller of the burden of meeting high computational bandwidth requirements. This is what we will accomplish using a tunnel. Consider the following vital actions for preventing a malicious attack:

- The malicious queue will first send packets to the tunnel marked with an alert, signifying their presence. The usual queue will forward packets to the appropriate destination addresses.
- For additional processing, the controller will examine the header fields of the alarm-triggering packets, including the source and destination addresses, source port, destination port, and virtual LAN (VLAN) priority. The controller can decide whether to discard the packet or route it to alternative ports using the results of this analysis.
- Highly dangerous packets will be discarded, and the relevant IP addresses will be banned listed for a predetermined amount of time. Because these packets come from the same source repeatedly, this action is conducted.
- The source IP address will be permanently banned, prohibiting the reception of any more packets from that source, if the aforementioned behaviour continues for a certain period.

- However, the controller can unlock the source to allow regular traffic to continue once the packet flooding lowers below a certain level in the queue.

Attack mitigation Algorithm 3 is mentioned as follows.

```
Algorithm 3: Attack Mitigation Module
Input:   Packet With Generated Alarm
Output:  Mitigates DoS Attack
1.  Investigate packet
2.  Read header, initialize threshold time , flow req=0, threshold time
3.  Analyse IP
4.  Develop strategies for attack handling
5.    If Src IP from Banned List do
6.        if flow request > 4  do
7.            Block IP
8.            Drop packet
9.            Add new flow rule to block communication
10.       else
11.           Increment Flow req
12.           Drop Packet
13.           Warn IP
14.       end if
15.   else
16.       Increment Flow req
17.       Drop packet
18.   end if
19.   if No more packets coming from same IP at threshold time then
20.       Unblock IP
21.       Remove IP from Banned List
```

4 | RESULTS AND ANALYSIS

4.1 | Dataset

The CICDoS 2017 dataset was used for both training and testing the classification models. This dataset addresses many of the problems and restrictions of previous efforts. Twenty network traffic features were retrieved and computed for all benign and DoS flows, and these features are used to designate CICDoS 2017. The dataset provides the results of the network traffic analysis together with labelled flows that are based on the following like time stamp, source and destination IPs, source and destination ports, protocols, and attack. Outliers and values that were either infinite or empty were removed from the data before it was shuffled. In order to perform binary classification, the category labels were converted to integer format, where 1 indicates safety and 0 indicates danger. Misclassification was eliminated by indexing the string values and then standardizing the data. The classification record from dataset is shown in Table 1.

TABLE 1 Classification record.

Number of classes	2
Normal records	2,153,72 (50.02%)
Malicious records	2,153,72 (49.98%)
Total features	79
Features selected	20

4.2 | Experimental setup

For the evaluation to be conducted, the following installations are required:

- Install the Ubuntu operating system on a virtual machine. Install the SDN network and switches on Ubuntu within this virtual machine.
- Install the Python-based Ryu controller program. The Ryu controller will be executed using an integrated development environment like PyCharm. By connecting Ryu to the switches, incoming packets can be forwarded to Ryu for further processing. The Ryu controller then makes decisions based on the packets that have been received.
- Utilize Ryu controller to implement the essential network protocols for the assessment.

When discussing SDN, the term “OpenFlow switch” or “SDN switch” is used to describe either a computer program or a physical device that handles packet forwarding. OpenFlow is used by the SDN controller to program these devices. The use of the SDN controller simplifies the programming of the switches, which is an advantage.

4.2.1 | SDN RYU controller

Ryu controller is an open, SDN controller designed to increase the agility of the network by making it easy to manage and adapt how traffic is handled. In general, the SDN controller is the brain of the SDN environment, communicating information down to the switches and routers with southbound application programming interface (API) and up to the applications and business logic with northbound APIs.

4.2.2 | SDN without the proposed solution

An attacker might quickly flood an SDN-enabled network through their host. It will be difficult to differentiate between regular and this kind of traffic because they will be combined. If a switch cannot determine what to do with a newly arriving packet, it will store it in its Flow Buffer and subsequently send a Packet-In message to the controller, as per the OpenFlow protocol as shown in Figure 1. As a result, if a DoS attack were to occur, the controller would need to process a massive amount of Packet-In messages caused by the flooding traffic, which may potentially overwhelm the system and prevent it from handling legitimate traffic. It is also possible that the attacking traffic will use up all available bandwidth between the controller and the switches, thus slowing down the entire network.

4.2.3 | SDN with the proposed solution

An attack begins when the attacker floods the switches with packets, as shown in Figure 4. The open-source SDN Ryu controller recognizes this suspicious behaviour and, after checking its behaviour, blocks the offending IP address. The required rules are then given to the switches, telling them to prevent incoming packets from the specified IP address.

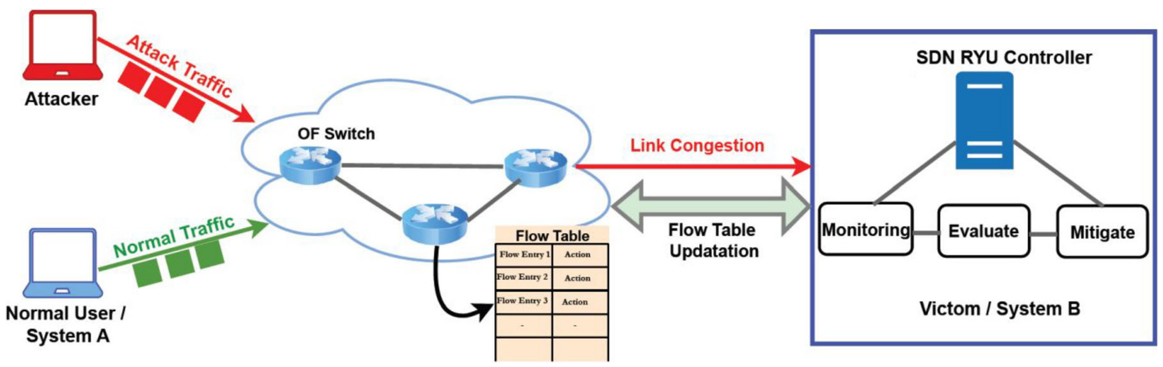

FIGURE 4 Proposed system visualization diagram. SDN, software-defined networking.

4.3 | Results discussion

The proposed system's performance has been analysed and compared with existing systems, in particular references.[12,18,56,60] The comparison's findings are listed below.

4.3.1 | CPU utilization

The level of CPU utilization during normal traffic that our system and benchmarks offer with comparison is shown in Figure 5. During normal traffic, the average CPU utilization of Imran et al., Gao et al., Revathi et al., Wang et al., and our proposed system are almost equal and 7.50%, 7.63%, 7.52%, 7.75%, and 7.62%, respectively, as shown in Figure 5.

On the other hand, during the DoS attack, the average CPU utilization of our proposed system with benchmarks is compared and shown in Figure 6. During the DoS attack, the controller's CPU resources are diverted to the endless installation of meaningless flow rules for flow tables, leaving less processing power for actual flow requests coming from the source. When an attack occurs, the percentage of time that the CPU is being used immediately increases to its

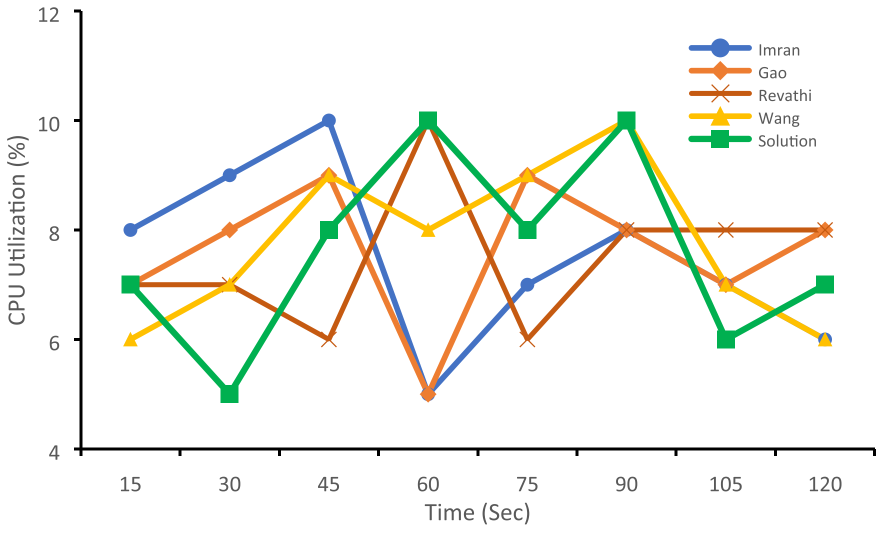

FIGURE 5 Central processing unit (CPU) utilization without denial-of-service attack.

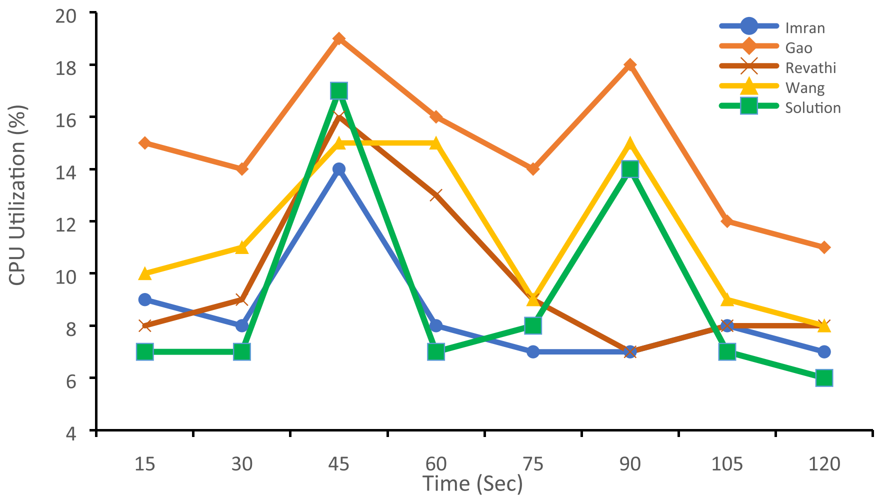

FIGURE 6 Central processing unit (CPU) utilization with denial-of-service attack.

high capacity, which is somewhere around 9% and takes less than 2 s to reach. Then, it goes down slowly because the packet filter starts to pass this in the malicious queue.

Following a delay of around 2 s, the level of CPU use never changes. However, at point, the packet in the buffer is capable of storing the packet-in messages in such an effective manner while only consuming approximately 1% of the CPU's utilization. After approximately 8 s, there is a brief increase in CPU utilization, which peaks at approximately 4% before dropping down precipitously in 2 s. This is due to the two-phase filtering that occurs in the packet filter. During DoS attacks, the average CPU utilization of Imran et al., Gao et al., Revathi et al., Wang et al., and our proposed system are almost equal and 8.50%, 14.87%, 9.75%, 11.50%, and 9.13%, respectively, as shown in Figure 6. The finding demonstrates that our system is able to conserve the computational power of the control layer effectively; in addition to this, it also demonstrates that the latency of the packet filter is relatively low. The level of CPU utilization during a DoS attack that benchmarks offer with comparison is shown in Figure 6.

4.3.2 | Control channel bandwidth

The percentages of control channel bandwidth during normal traffic with our system and benchmarks are compared and shown in Figure 7. During normal traffic, the average control channel bandwidth of Imran et al., Gao et al., Revathi et al., Wang et al., and our proposed system are almost equal and 8.63%, 8.25%, 8.13%, 8.34%, and 8.13%, respectively, as shown in Figure 7.

However, it is difficult to isolate attacked traffic from the regular traffic because they are mixed together. OpenFlow specifies that a switch will store a new packet in its Flow Buffer before sending a Packet-In message to the controller for further direction if the switch has no prior knowledge of what to do with the packet. In order to check bandwidth, we send multiple packets from the same IP in 5 s, and we see that when packets flood came in, bandwidth increases by a maximum of 30 kbps but it suddenly falls down and again increases after the 30 s as shown in Figure 8. The controller may run out of resources needed to process normal traffic if a DoS attack occurs and a large number of Packet-In messages are generated by the flooded traffic in a short period. Additionally, the attacking traffic may use up all available bandwidth between the controller and the switches, drastically slowing down the entire network. Requests for new flows must pass through the control channel, which might become jammed during DoS attacks. Our proposed solution not only blocks attacked traffic at switches because if the same flow request will come from the same IP four times which is from the banned list, then that packet will be dropped, and the IP will be blocked at switches. In this way, the proposed solution also provides ease on the control channel by reducing the bandwidth by 14.13% as compared with benchmarks whose bandwidth increases by 15.38%, 15.13%, 15.54%, and 15.25%.

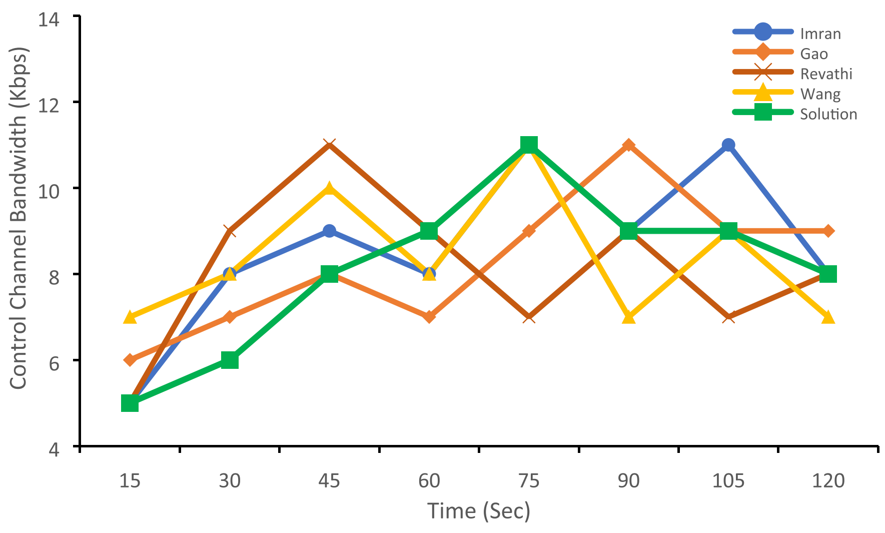

FIGURE 7 Control channel bandwidth without denial-of-service attack.

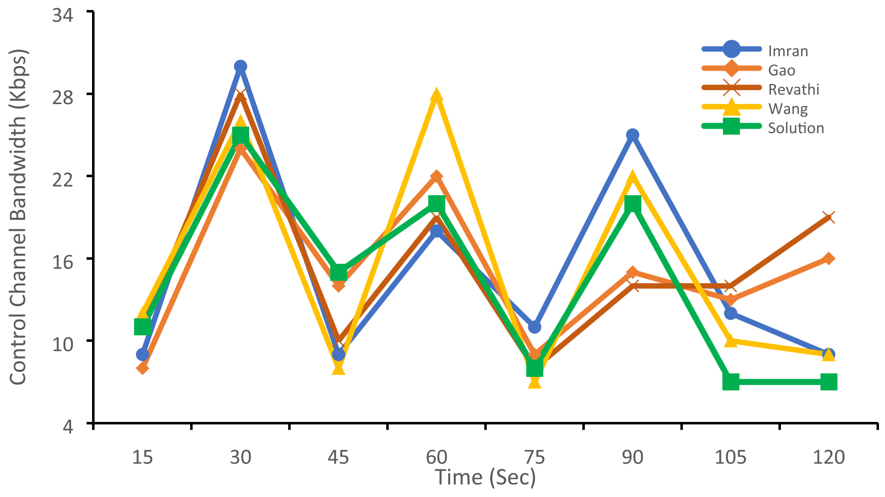

FIGURE 8 Control channel bandwidth with denial-of-service attack.

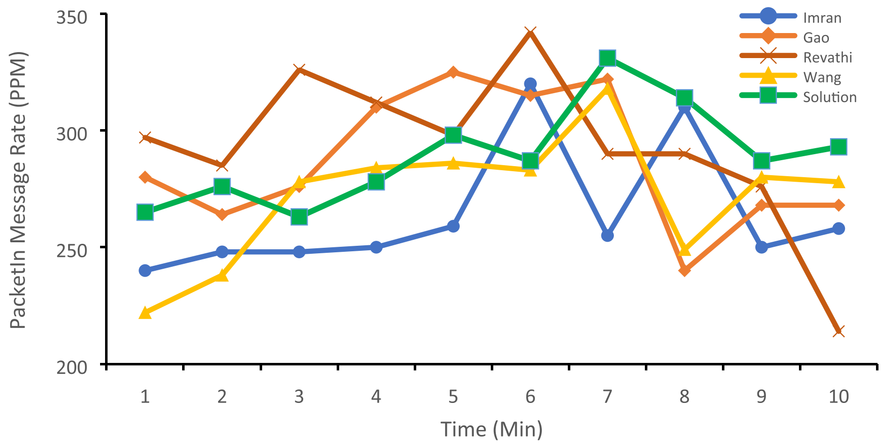

FIGURE 9 Packet-in messages without denial-of-service attack.

4.3.3 | Packet-in messages

In the OpenFlow network, switches utilize flow rules to handle messages received from the SDN. OpenFlow switches match packet headers to flow tables. Each match triggers a flow table action and table misses indicate no match as this activity is very important during SDN traffic flow. Figure 9 shows that the flow rate of packets during normal traffic sent to the controller is between 200 to 400 packets per minute which is approximately the same range for all the benchmarks.

While during DoS attacks, the number of flows entering a switch increases, new rules must be added to the device's flow table; for this, the controller compares incoming packets to its database. The controller first checks it if that packet comes from the banned list in packet filtration module; if that happens, then that packet will be dropped, and if it is not, so then it creates a new flow table if there is no match. Figure 10 shows that during DoS attack, the average flow rate requests per minute increases for the benchmarks as 5565, 5209, 5874, and 5272, and again, it is less for proposed solution which is 5197 because of packet filtration module.

4.3.4 | Evaluation of latency

Average response times during normal traffic for benchmarks and proposed solution are depicted in Figure 11. Here, we can observe that our proposed solution provides better response time in milliseconds during normal traffic as an

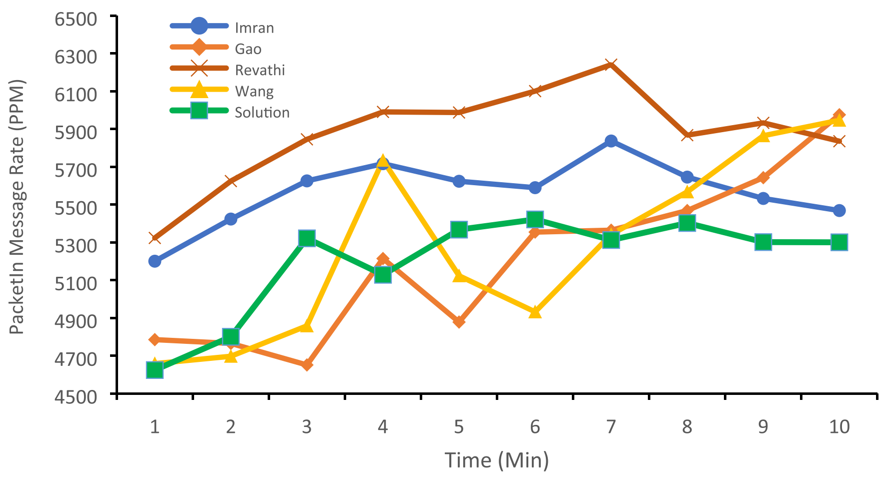

FIGURE 10 Packet-in messages with denial-of-service attack.

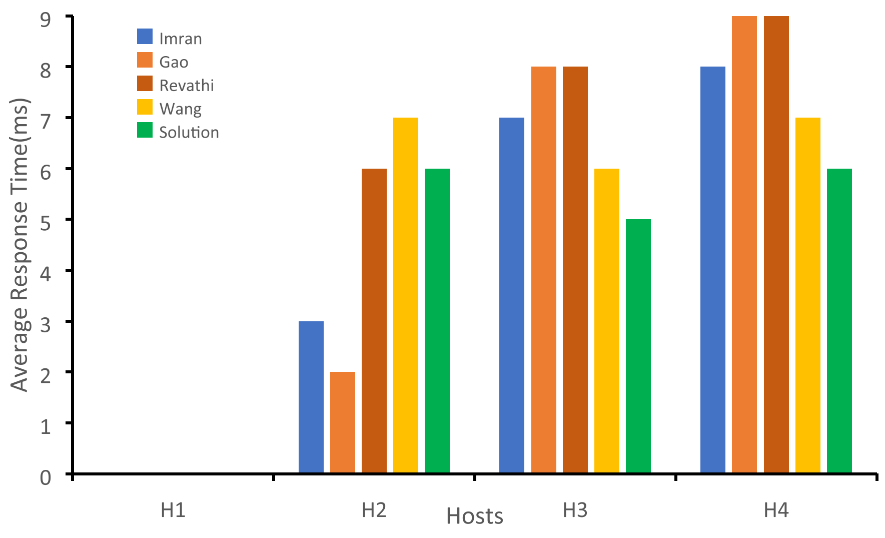

FIGURE 11 Response time without denial-of-service attack.

average of 4.25% because of its queue system where normal packets and malicious packets are separated in two different queues while benchmarks have higher average response time as 4.50%, 4.75%, 5.75%, and 5.0%.

While during DoS attacks, the average response time of our proposed solution is approximately (6.50%) the same as some of the benchmarks because of two-phase authentication filter where packets are placed in two separate queues after filtration, and it rejects malicious traffic and frees up the controller to serve legitimate hosts. The average response times for the benchmarks as depicted in Figure 12 are 6.0%, 6.5%, 6.5%, and 5.5%.

4.3.5 | Attack identification false rate

In Figure 13, we can see how well the TPAAD detects attacks. As the frequency of attacks increases, so does the proportion of false positives. This is due to the fact that a higher attack rate will result in a greater occurrence of the same flow within a given time window. So, in order to filter out attack traffic, packet filtration will employ a higher threshold and move the attack traffic to malicious queue. As the attack rate rises, a greater number of attack packets are falsely labelled as normal flow, but the fraction of false positives remains constant, making for a more consistent recall rate. Overall, the TPAAD filtering method has a false-positive rate of less than 5% and can accurately identify over 99% of attack traffic.

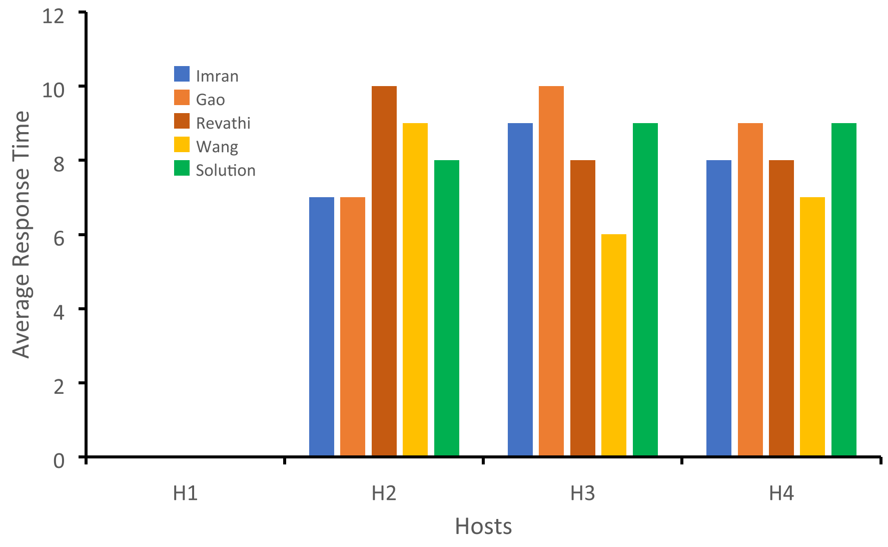

FIGURE 12 Response time with denial-of-service attack.

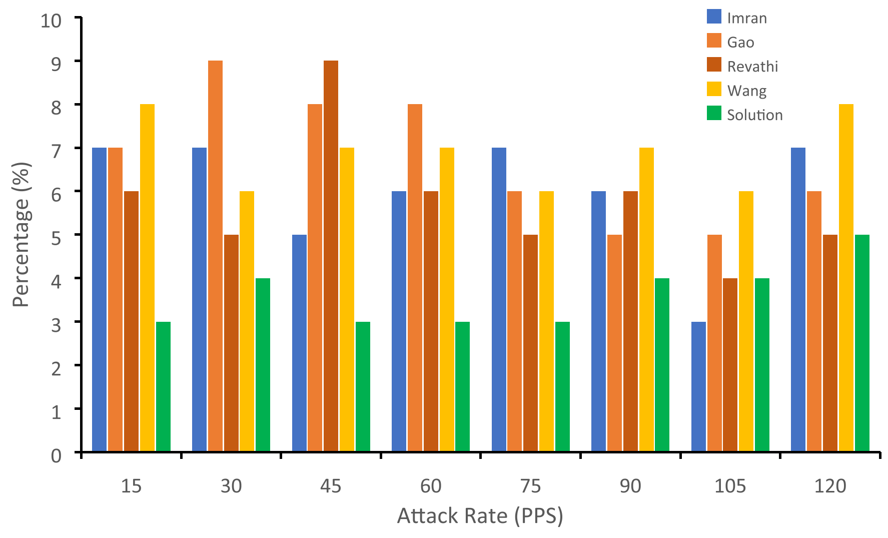

FIGURE 13 Attack identification and false-positive rate.

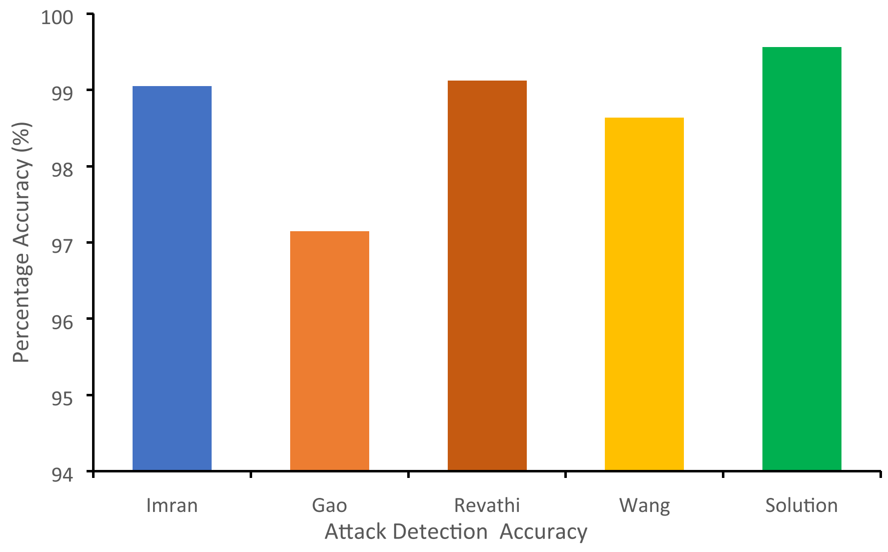

FIGURE 14 Attack detection accuracy.

4.3.6 | Detection accuracy

Due to TPAAD, we evaluate our system with both real-time data and CICDoS 2017 dataset. During evaluation, we analysed that some of the benchmarks provide false alarm generation. On the other hand, proposed solution not only blocks the IP. Our proposed system provides higher detection accuracy as 99.56% than the other benchmarks 99.05%, 97.15%, 99.12%, and 98.64% as depicted in Figure 14.

5 | CONCLUSIONS

This study presents a proposed technique, called the TPAAD and mitigation, which aims to protect SDN systems from DoS attacks. The initial step of the proposed approach involves the identification of malevolent network traffic, which arises from DoS attacks. This identification is accomplished through the utilization of TPAAD detection. Subsequently, the identified malicious traffic is redirected to a mitigation mechanism, wherein the focus is on terminating the malevolent communication rather than disrupting the entire host. In order to enhance precision, we employed SVM and KNN, two widely utilized machine learning methodologies that are currently prevalent for efficient identification. The comprehensive methodology was applied to evaluate the efficiency of the strategy using the CICDoS 2017 dataset within the context of an SDN environment. The unblocking of a host takes place subsequent to the verification that the host is no longer involved in the transmission or receiving of harmful network traffic. The verification process occurs subsequent to the placement of the host in the banned queue. The experiment was performed using the Ubuntu operating system within a VMware virtualized environment. Based on our examination of benchmarks, it has been determined that the implementation of SDN with our proposed solution yields enhanced precision by mitigating the occurrence of erroneous notifications, minimizing CPU utilization, optimizing control channel bandwidth, and reducing the number of flow requests directed towards the controller, all without necessitating the incorporation of supplementary hardware. Based on our conducted tests, it can be inferred that the proposed approach is effective in detecting and mitigating SDN-based DoS attacks while incurring minimal additional resource utilization. In the foreseeable future, we aim to implement our proposed method on an actual network environment and evaluate its performance.

ACKNOWLEDGEMENTS
This research is fully funded by Universiti Malaysia Sarawak, Kota Samarahan, Malaysia under grant number F08/PGRG/2058/2021.

CONFLICT OF INTEREST STATEMENT
All the authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

DATA AVAILABILITY STATEMENT
The analysed data for this research article is available upon request from the corresponding author.

ORCID
Zeeshan Ahmad https://orcid.org/0000-0002-8530-864X

REFERENCES

1. Khan AS, Ahmad Z, Abdullah J, Ahmad F. A spectrogram image-based network anomaly detection system using deep convolutional neural network. *IEEE Access*. 2021;9:87079-87093. doi:10.1109/ACCESS.2021.3088149
2. Jan SU, Xi A, Abbasi FA, Khan AS. A verifiably secure ECC based authentication scheme for securing IoD using FANET. *IEEE Access*. 2022;10:95321-95343. doi:10.1109/ACCESS.2022.3204271
3. Jimenez MB, Fernandez D, Rivadeneira JE, Bellido L, Cardenas A. A survey of the main security issues and solutions for the SDN architecture. *IEEE Access*. 2021;9:122016-122038. doi:10.1109/ACCESS.2021.3109564
4. Al-Mafrachi BHA. Detection of DDoS attacks against the SDN controller using statistical approaches. 2017.
5. Alzubi OA, Qiqieh I, Alzubi JA. Fusion of deep learning based cyberattack detection and classification model for intelligent systems. *Cluster Comput*. 2023;26(2):1363-1374. doi:10.1007/s10586-022-03686-0
6. Alzubi OA, Alzubi JA, Al-Zoubi AM, Hassonah MA, Kose U. An efficient malware detection approach with feature weighting based on Harris Hawks optimization. *Cluster Comput*. 2022;25(4):2369-2387. doi:10.1007/s10586-021-03459-1

7. Alzubi JA. Blockchain-based Lamport Merkle digital signature: authentication tool in IoT healthcare. *Comput Commun*. 2021;170:200-208. doi:10.1016/j.comcom.2021.02.002
8. Esch J. Software-defined networking: a comprehensive survey. *Proc IEEE*. 2015;103(1):10-13. doi:10.1109/JPROC.2014.2374752
9. Kaljic E, Maric A, Njemcevic P DoS attack mitigation in SDN networks using a deeply programmable packet-switching node based on a hybrid FPGA/CPU data plane architecture. ICAT 2019-27th International Conference on Information, Communication and Automation Technologies, Proceedings. 2019. doi:10.1109/icat47117.2019.8938862
10. Nisa N, Khan AS, Ahmad Z, Aqeel S, Asim J, Afzal S. Conceptual review of DoS attacks in software defined networks. In: *Proceedings - AiIC 2022: 2022 Applied Informatics International Conference: Digital Innovation in Applied Informatics During the Pandemic*. Institute of Electrical and Electronics Engineers Inc.; 2022:154-158. doi:10.1109/AiIC54368.2022.9914598
11. Mohammadi R, Javidan R, Conti M, Member S. SLICOTS: an SDN-based lightweight countermeasure for TCP SYN flooding attacks. *IEEE Trans Netw*. 2017;14(2):487-497.
12. Imran M, Durad MH, Khan FA, Abbas H. DAISY: a detection and mitigation system against denial-of-service attacks in software-defined networks. *IEEE Syst J*. 2020;14(2):1933-1944. doi:10.1109/JSYST.2019.2927223
13. Shu Z, Wan J, Li D, Lin J, Vasilakos AV, Imran M. Security in software-defined networking: threats and countermeasures. *Mobile Netw Appl*. 2016;21(5):764-776. doi:10.1007/s11036-016-0676-x
14. Wang T, Chen H. SGuard: a lightweight SDN safe-guard architecture for DoS attacks. *China Commun*. 2017;14(6):113-125. doi:10.1109/CC.2017.7961368
15. Gray N, Zinner T, Tran-gia P. Enhancing SDN security by device fingerprinting. pp. 879-880, 2017.
16. Khan AS, Yahya MIB, Zen KB, et al. Blockchain-based lightweight multifactor authentication for cell-free in ultra-dense 6G-based (6-CMAS) cellular network. *IEEE Access*. 2023;11:20524-20541. doi:10.1109/ACCESS.2023.3249969
17. Zhang P, Wang H, Hu C, Lin C. On denial of service attacks in software defined networks; on denial of service attacks in software defined networks. *IEEE Netw*. 2016;30(6):28-33. doi:10.1109/MNET.2016.1600109NM
18. Gao S, Peng Z, Xiao B, Hu A, Song Y, Ren K. Detection and mitigation of DoS attacks in software defined networks. *IEEE/ACM Trans Netw*. 2020;28(3):1419-1433. doi:10.1109/TNET.2020.2983976
19. Bera P, Saha A, Setua SK. "Denial of service attack in software defined network," Proceedings of 2016 5th International Conference on Computer Science and Network Technology, ICCSNT 2016, no. December, pp. 497-501, 2017. doi:10.1109/ICCSNT.2016.8070208
20. Kumar N Cross layer design in software defined networking (SDN): challenges and solutions. no. December. 2018. doi:10.13140/RG.2.2.34256.15361
21. Masoudi R, Ghaffari A. Software defined networks: a survey. *J Netw Comput Appl*. 2016;67(December):1-25. doi:10.1016/j.jnca.2016.03.016
22. Kaur H. Cross-layer design in software defined networks (SDNs): issues and possible solutions. *Researchgate Net*. 2021; April:0-9. [Online]. Available: https://www.researchgate.net/profile/Jashanpreet_Kaur2/publication/322114928_Cross-layer_design_in_Software_Defined_Networks_SDNs_issues_and_possible_solutions/links/5a45b33a0f7e9ba868a94225/Cross-layer-design-in-Software-Defined-Networks-SDNs-issues-and
23. Srivastava V, Motani M. Cross-layer design: a survey and the road ahead. *IEEE Commun Magol*. 2005;43(12):112-119. doi:10.1109/MCOM.2005.1561928
24. Lara A, Ramamurthy B. OpenSec: policy-based security using software-defined networking. *IEEE Trans Netw Service Manage*. 2016;13(1):30-42.
25. Rawat DB, Member S, Reddy SR. Software defined networking architecture, security and energy efficiency: a survey. *IEEE Commun Surv Tutor*. 2017;19(1):325-346.
26. El-mougy A, Ibnkahla M, Hegazy L. "Software-defined wireless network architectures for the Internet-of-Things." 2015:804-811.
27. Dargahi T, Caponi A, Ambrosin M, Member S, Bianchi G. A survey on the security of stateful SDN data planes. 2017;19(3):1701-1725.
28. Moazzeni S, Khayyambashi MR, Movahhedinia N, Callegati F. On reliability improvement of software-defined networks. *Computer Networks*. 2018;133:195-211. doi:10.1016/j.comnet.2018.01.023
29. Khan AS, Sattar MA, Nisar K, et al. A survey on 6G enabled light weight authentication protocol for UAVs, security, open research issues and future directions. *Appl Sci (Switzerland)*. 2023;13(1):277. doi:10.3390/app13010277
30. Alsmadi I, Xu D. Security of software defined networks: a survey. *Comput Secur*. 2015;53:79-108. doi:10.1016/j.cose.2015.05.006
31. Gao S, Li Z, Xiao B, Wei G. Security threats in the data plane of software-defined networks. *IEEE Netw*. 2018;32(4):108-113. doi:10.1109/MNET.2018.1700283
32. Open Networking Foundations (ONF). "Threat analysis for the SDN architecture," no. July, pp. 1-21, 2016.
33. Aqeel S, Shahid Khan A, Ahmad Z, Abdullah J. A comprehensive study on DNA based security scheme using deep learning in healthcare. *EDPACS*. 2022;66(3):1-17. doi:10.1080/07366981.2021.1958742
34. Khan AS, Javed Y, Saqib RM, et al. Lightweight multifactor authentication scheme for NextGen cellular networks. *IEEE Access*. 2022;10:31273-31288. doi:10.1109/ACCESS.2022.3159686
35. Ahmad Z, Khan AS, Aqeel S, et al. S-ADS: spectrogram image-based anomaly detection system for IoT networks. In: *Proceedings - AiIC 2022: 2022 Applied Informatics International Conference: Digital Innovation in Applied Informatics During the Pandemic*. Institute of Electrical and Electronics Engineers Inc.; 2022:105-110. doi:10.1109/AiIC54368.2022.9914599
36. Ahmad Z, Shahid Khan A, Wai Shiang C, Abdullah J, Ahmad F. Network intrusion detection system: a systematic study of machine learning and deep learning approaches. *Trans Emerg Telecommun Technol*. 2021;32(1):e4150. doi:10.1002/ett.4150

37. Mousavi SM, St-Hilaire M. Early detection of DDoS attacks against software defined network controllers. *J Netw Syst Manag*. 2018;26(3): 573-591. doi:10.1007/s10922-017-9432-1
38. Shin S, Yegneswaran V, Porras P, Gu G. Avant-guard: scalable and vigilant switch flow management in software-defined networks. *Proceed ACM Conf Comput Commun Security*. 2013;413-424. doi:10.1145/2508859.2516684
39. Wang H, Xu L, Gu G. FloodGuard: a DoS attack prevention extension in software-defined networks.
40. Kandoi R, Antikainen M. Denial-of-service attacks in OpenFlow SDN networks. Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015, pp. 1322-1326, 2015. doi:10.1109/INM.2015.7140489
41. Khan AS, Abdullah J, Zen K, Tarmizi S. Secure and scalable group rekeying for mobile multihop relay network. *Adv Sci Lett*. 2017;23(6): 5242-5245. doi:10.1166/asl.2017.7350
42. Oktian YE, Lee S, Lee H. Mitigating denial of service (DoS) attacks in OpenFlow networks. *Int Conf ICT Convergence*. 2014;325-330. doi: 10.1109/ICTC.2014.6983147
43. Khan AS, Javed Y, Abdullah J, Zen K. Trust-based lightweight security protocol for device to device multihop cellular communication (TLwS). *J Ambient Intell Humaniz Comput*. 2021. doi:10.1007/s12652-021-02968-6
44. Ahmad Z, Shahid Khan A, Nisar K, et al. Anomaly detection using deep neural network for IoT architecture. *Appl Sci (Switzerland)*. 2021;11(15):7050. doi:10.3390/app11157050
45. Asim J, Khan AS, Saqib RM, et al. Blockchain-based multifactor authentication for future 6G cellular networks: a systematic review. *Appl Sci (Switzerland)*. 2022;12(7):3551. doi:10.3390/app12073551
46. Shoeb A, Chithralekha T. Resource management of switches and controller during saturation time to avoid DDoS in SDN. Proceedings of 2nd IEEE International Conference on Engineering and Technology, ICETECH 2016, no. March, pp. 152-157, 2016. doi:10.1109/ICETECH.2016.7569231
47. Imran M, Durad MH, Khan FA, Derhab A. Reducing the effects of DoS attacks in software defined networks using parallel flow installation. *HCIS*. 2019;9(1):16. doi:10.1186/s13673-019-0176-7
48. Cui Y, Yan L, Li S, et al. SD-Anti-DDoS: fast and efficient DDoS defense in software-defined networks. *J Netw Comput Appl*. 2016;68:65-79. doi:10.1016/j.jnca.2016.04.005
49. Peng H, Sun Z, Zhao X, Tan S, Sun Z. A detection method for anomaly flow in software defined network. *IEEE Access*. 2018;6:27809-27817. doi:10.1109/ACCESS.2018.2839684
50. Banitalebi Dehkordi A, Soltanaghaei MR, Boroujeni FZ. The DDoS attacks detection through machine learning and statistical methods in SDN. *J Supercomput*. 2021;77(3):2383-2415. doi:10.1007/s11227-020-03323-w
51. Tan L, Pan Y, Wu J, Zhou J, Jiang H, Deng Y. A new framework for DDoS attack detection and defense in SDN environment. *IEEE Access*. 2020;8:161908-161919. doi:10.1109/ACCESS.2020.3021435
52. Ye J, Cheng X, Zhu J, Feng L, Song L. A DDoS attack detection method based on SVM in software defined network. *Sec Commun Netw*. 2018;2018:1-8. doi:10.1155/2018/9804061
53. Tuan NN, Hung PH, Nghia ND, Van Tho N, Van Phan T, Thanh NH. A DDoS attack mitigation scheme in ISP networks using machine learning based on SDN. *Electronics (Switzerland)*. 2020;9(3):413. doi:10.3390/electronics9030413
54. Kumar P, Tripathi M, Nehra A, Conti M, Lal C. SAFETY: early detection and mitigation of TCP SYN flood utilizing entropy in SDN. *IEEE Tran Netw Service Manag*. 2018;15(4):1545-1559. doi:10.1109/TNSM.2018.2861741
55. Sahoo KS, Puthal D, Tiwary M, Rodrigues JJPC, Sahoo B, Dash R. An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics. *Future Gen Comput Syst*. 2018;89:685-697. doi:10.1016/j.future.2018.07.017
56. Revathi M, Ramalingam VV, Amutha B. A machine learning based detection and mitigation of the DDOS attack by using SDN controller framework. *Wirel Pers Commun*. 2022;127(3):2417-2441. doi:10.1007/s11277-021-09071-1
57. Khan AS, Balan K, Javed Y, Tarmizi S, Abdullah J. Secure trust-based blockchain architecture to prevent attacks in VANET. *Sensors*. 2019;19(22):4954. doi:10.3390/s19224954
58. Alzubi JA. Bipolar fully recurrent deep structured neural learning based attack detection for securing industrial sensor networks. *Trans Emerg Telecommun Technol*. 2021;32(7):e4069. doi:10.1002/ett.4069
59. Movassagh AA, Alzubi JA, Gheisari M, et al. Artificial neural networks training algorithm integrating invasive weed optimization with differential evolutionary model. *J Ambient Intell Humaniz Comput*. 2023;14(5):6017-6025. doi:10.1007/s12652-020-02623-6
60. Ahmad Z, Khan AS, Zen K, Ahmad F. MS-ADS: multistage spectrogram image-based anomaly detection system for IoT security. *Trans Emerging Tel Tech*. 2023;34(8):e4810. doi:10.1002/ett.4810

How to cite this article: Nisa N, Khan AS, Ahmad Z, Abdullah J. TPAAD: Two-phase authentication system for denial of service attack detection and mitigation using machine learning in software-defined network. *Int J Network Mgmt*. 2024;34(3):e2258. doi:10.1002/nem.2258