Hindawi
Security and Communication Networks
Volume 2018, Article ID 1308678, 14 pages
https://doi.org/10.1155/2018/1308678

Research Article

Validating User Flows to Protect Software Defined Network Environments

Ihsan H. Abdulqadder,[1] **Deqing Zou**,[1,2] **Israa T. Aziz**,[1,3] **and Bin Yuan**[1]

[1]*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China*
[3]*University of Mosul, Mosul 41002, Iraq*

Correspondence should be addressed to Ihsan H. Abdulqadder; ihsan@hust.edu.cn

Received 15 September 2017; Revised 13 November 2017; Accepted 10 December 2017; Published 19 February 2018

Academic Editor: Petros Nicopolitidis

Software Defined Network is a promising network paradigm which has led to several security threats in SDN applications that involve user flows, switches, and controllers in the network. Threats as spoofing, tampering, information disclosure, Denial of Service, flow table overloading, and so on have been addressed by many researchers. In this paper, we present novel SDN design to solve three security threats: flow table overloading is solved by constructing a star topology-based architecture, unsupervised hashing method mitigates link spoofing attack, and fuzzy classifier combined with L1-ELM running on a neural network for isolating anomaly packets from normal packets. For effective flow migration Discrete-Time Finite-State Markov Chain model is applied. Extensive simulations using OMNeT++ demonstrate the performance of our proposed approach, which is better at preserving holding time than are other state-of-the-art works from the literature.

1. Introduction

SDNs have gained worldwide attention from academia and industry and have been the subject of numerous research efforts. SDN architecture is decoupled into a data plane and a control plane [1–3]; this architecture comprises an application layer, a control layer, and an infrastructure layer [4]. However, an SDN is vulnerable without additional security, because it is subject to attacks such as control plane saturation, flow table overloading, Distributed Denial of Service (DDoS), ID spoofing, link spoofing, new flow, and man-in-the-middle attacks [5–10]. Security solutions for SDNs can be either network- or host-based. In a DDoS attack, the system is attacked by multiple scatter sources in a manner that denies service to valid users participating in the network [2].

DDoS attacks can be detected and prevented by IP blacklisting, connection success ratios, throttling connections, and aggregate analysis. A DDoS attack on an SDN affects the control layer, data layer, and data-control interfaces. The DDoS attack detection and Mitigation Framework (DDMF) was proposed by the authors of [9]. DDMF is a network design comprising an SDN router application, a capture server, and a detection server. These components work by analyzing the network data traffic. Invalid switches are identified by echo request messages; invalid routers are also predicted. The detection server is responsible for blocking DDoS attacks predicted by traffic flows. Simple sampling techniques are used to automatically detect different types of DDoS attacks by analyzing traffic in real-time in OpenFlow [11]. The open flow controller is configured with a Finite State Machine (FSM) mechanism to analyze policies. Vulnerabilities in SDNs are discussed using an open standardized semiqualitative, semiquantitative scoring system to determine the vulnerability severity levels [12]. The SDN features are integrated into a Common Vulnerability Scoring System (CVSS) with the support of an Analytical Hierarchy Process (AHP). Using the AHP, a hierarchy tree is constructed by estimating the SDN asset weights mathematically. Then, the CVSS computations are performed based on the weights.

Most researchers have focused on analyzing traffic to perform DDoS attack mitigation. The Software Defined Networking (DaMask) is involved with cloud architecture

[13], and attack detection and mitigation occur through a cloud provider rather than by a controller. Single point SDN failure vulnerabilities (i.e., failures due to a single controller) are solved by using a distributed multiple-controller design [14, 15]. In [14], a Distributed Rule Store (DRS) was proposed for use with multicontroller architectures in which flow rules are computed by applications cached across the controllers. Each controller is supplied with a set of rules that can be modified if a malicious flow is identified. A multicontroller cloud-supported design to protect an SDN against Byzantine-Resilient attacks was described in [15], in which the authors proposed the Requirement First Assignment (RQFA) algorithm. This algorithm assigns controllers to switches based on the requirements of a number of controllers (i.e., the required number of controllers changes dynamically with respect to the switches' fault-tolerances). Using multiple controllers allows confirmation of updates to flow table entries by switches. CAuth is used in SDNs to protect against spoofing attacks [10]. A collaborative approach is designed by manipulating OpenFlow messages.

In this paper, we design a novel SDN architecture to resist DDoS, flow table overloading, and link spoofing attacks in network flow traffic. In the traditional SDN architecture, each switch is controlled by a single controller; here, however, we propose using multiple controllers to effectively predict attacks and maintain security. In this work, we have proposed a hybrid classifier that guarantees efficient classification of normal and anomaly traffic. Our SDN architecture uses a star topology to mitigate the effects of flow table overloading attacks. To evaluate security, we have designed our proposed work by considering attacker behaviors in the network.

Major Contributions. The main contributions of this paper are as follows.

(1) We propose a secure SDN multicontroller architecture in which users are connected by a star topology to mitigate flow table overloading attacks.

(2) Switches evaluate each user in the flow using a trust value. The trust value is an additional field included along with the user's flow table, and its value depends on each user's data exchange.

(3) A Discrete-Time Finite-State Markov Chain (DTMC) model is utilized to periodically update the switch states regarding the flow table occupancy.

(4) Switches are verified by the controllers based on an unsupervised hashing method, which solves the problem of link spoofing attacks that occur between switches.

(5) We apply a hybrid classifier (i.e., a combination of a fuzzy classifier with the L1-Extreme Learning Machine (L1-ELM)). To achieve faster processing, this hybrid classifier is executed over a neural network.

The rest of this paper is organized as follows. Section 2 provides background information about SDNs and their security. A review of the related works on SDN is presented in Section 3. Section 4 presents the problem formulation and system design. Section 5 describes all the contributions of this paper in detail and provides the design methodology. Section 6 reports the simulation results and provides a comparative analysis. Finally, Section 7 concludes the paper.

2. Preliminary Knowledge

2.1. Software Defined Network. SDN architecture consists broadly of three functional planes: application, control, and data [16].

(i) The application plane consists of intelligent SDN applications that are aware of the network's capabilities and that can efficiently switch among the available network resources.

(ii) The control plane is logically centralized and is responsible for translating application requests into low-level rules.

(iii) The data plane is also called the "infrastructure layer," because it consists of ready-state SDN devices.

An SDN has a dynamic networking architecture that uses the standardized open protocol called OpenFlow [4]. OpenFlow switches provide consolidated intelligence for performance optimization, coarse policy management, acceleration of novel features, and so on. OpenFlow-enabled SDN switches contain flow and group tables to identify user flows. These flow entries are responsible for controlling traffic flows arriving from large numbers of users. In SDN different OpenFlow controllers are involved such as NOX, POX, Beacon, Floodlight, and RYu [17]. The significant characteristic of OpenFlow controller is that it is enabled to achieve maximum throughput and minimum latency. This performance of OpenFlow in SDN is evaluated by measuring the packet-in messages based on a queuing model [18]. Measurement of network traffic arrived over the networks supports with effective evaluation of SDN environment

As discussed earlier, SDNs are vulnerable to several attacks that can be minimized by validating the flow table entries shown in Figure 1. In some cases, the OpenFlow switches can be compromised by malevolent attackers [16]. In an SDN, security is provided not only by a trusted authority but also by other features. For example, AVISPA was a security analysis system that guaranteed security between trusted domains [19]. The trusted domain authentication process consisted of two parts: controller platform certification and controller software certification. Using these certifications, replay attacks and intermediate attacks could be predicted because attackers were identified not only by valid certificates but also by packet aggregation [20]. OpenFlow switches monitor traffic; each traffic flow comprises a set of ten features. By analyzing the flow features from each user, aggregated flows are classified as benign or suspicious using preformulated rules. Security is a significant requirement in SDNs that must be implemented in such a way as to enhance the users' experiences while maintaining other network performance metrics.

2.2. Threats and Vulnerabilities in SDNs. SDNs are vulnerable to several threats because of the separation of their control and data planes. In an SDN, common security challenges include man-in-the-middle attacks, DoS attacks, DDoS attacks, and saturation attacks [21, 22]. A man-in-the-middle attack often occurs between a switch and a controller,

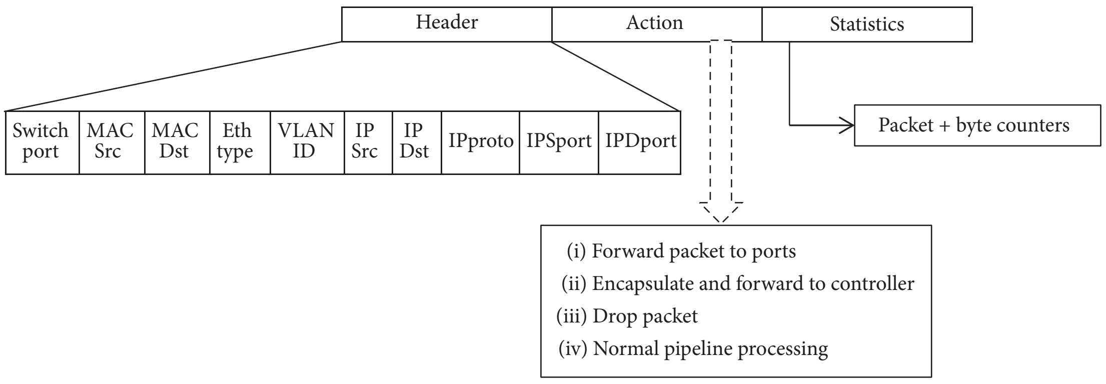


Figure 1: Example flow table entries of an OpenFlow switch.

targeting the data forwarding layer, whereas DoS and DDoS attacks can occur in either the data forwarding layer or the control layer and are intended to interfere with controller execution. DDoS attacks occur frequently in applications such as cloud computing and mobile infrastructure [2] and cause damage such as denial of service, controller flooding, and flow entry flooding in switches. Spoofing attacks modify switch rules, cause data leakage, and allow attackers to create malicious applications [10]. To overcome vulnerabilities from new flows, a Smart Security Mechanism (SSM) was proposed in [7] that checks each new flow arrival from a user; in this study, flow entries were verified with hit-rates. However, although this approach can identify an attacker in the data plane, other attack types target the control and application planes. Having several attackers in the network can cause the entire network to cease functioning. The authors of [23] focused entirely on enhancing security in SDNs. Security can be enhanced by network separation, flow monitoring, defining new policies, deploying security appliances, improving the control of flow entries, and adopting prototype security. Because they are subject to numerous threat and attack types, SDNs have certain challenges and problems, including flow table management, real-time traffic updates, SDN controller scalability, switch performance, and integrations with conventional networks. In SDN, security needs a major attention due to vulnerable threats involvement [24]. Due to vulnerabilities, state changes in OpenFlow switches were studied. Stateful rules are fed into SDN data plane switches and according to the local state information the state decisions are made. Most of the attacks in SDN are detected based on network traffic. To manage secure traffic flow, the Traditional Network Architecture (TNA) of SDN has followed firewall rules [25]. Firewall policies are also one of the efficient methods that are used to ignore malicious traffic into the network.

3. Related Work

This section presents all the previous research works concerning SDN security. We investigate three attack types: spoofing, overloading, and DDoS attacks. An IP spoofing attack impersonates a device such as switch, hub, or another network device [26], which subsequently leads to the next level—man-in-the-middle attacks. This type of attack can be detected by designing an application that ignores a packet whose verification result is deemed to be the result of IP spoofing. IP spoofing attacks were detected in SDN architectures by the CDNi network [27], in which Application Layer Traffic Optimization (ALTO) servers were deployed to map the network and transmission costs. Packets arriving at the switches are encrypted but must match specified partition ID and IP addresses. When spoofed packets are detected they are first blocked and then traced. However, encrypted packets require time to decrypt, which tends to increase the network delay when large numbers of packets arrive at a switch. Spoofing attacks not only involve IP addresses but also can occur during link discovery [28, 29].

Links between switches can be forged by intruders, causing link failure and inhibiting fast communication. Switches are unable to bear the cost of flow table management under a flow table overloading attack [5]. This type of attack develops over time and eventually leads to a DDoS attack. In [30] DoS attacks were identified using a proposed method that builds a hopping multicast tree. Attackers were analyzed using game theory, but this theory was not described in detail. An attacker targets a node to be attacked in the network during the hopping period. In this way, legitimate nodes are compromised by attackers. This work was designed to resist DoS attacks present in multicast routing.

The authors of [31] employed an approach that used user-switch remapping by SDN-based proxy switches, which were responsible for relaying data flows between users and servers. A greedy algorithm was also used to maximize the rate of attack segregation. A traffic detector was deployed on each proxy switch along with threshold values to determine whether a packet was participating in a DDoS attack. Initially, users were randomly mapped to proxy switches; then, after detecting an attacker; legitimate users are remapped to a different proxy. However, this remapping process makes the network complex. The authors of [32] reported that controllers are vulnerable to flooding attacks (i.e., DDoS attacks). Therefore, they proposed a feasible method to guard against this attack. In this method, a temporary table "T" is created in the controller that maintains the IP addresses of forwarded packets. This table also maintains a count of the packets received from each unique IP address. The controller uses hard timeouts and idle timeouts to segregate normal entries from attackers. DDoS attacks are identified by a corresponding increase in the packet counts. However, this method was unable to tolerate bulk attacker traffic.

Data Path Identification (DPID) in SDNs was intended to detect attacks on forwarding devices [33]. In this work, the proposed dynamic defense method was based on the Client-Puzzle model for managing devices. The DPID model is composed of a 5-tuple, and the deployed controller works under the Client-Puzzle model. DoS attackers are usually detected based on their identities, but attackers can forge their identities and create unique identities using advanced tools. New flow validation plays a significant role in SDNs because of the different threats involved. In [7], all incoming packets and messages were verified and unmatched packets were stored; if the received message was an asynchronous message then that particular flow was removed. Spoofing attacks in SDNs are related to DDoS attacks and have been analyzed by several authors [10, 34, 35]. CAuth mechanism, TopoGaurd, and POX controllers were used to mitigate attacks by monitoring incoming packets. All the preceding methods required a long time for identification, which allows malicious packets to participate in the network for longer periods. Numerous researchers in the SDN field have designed security mechanisms. Security is one of the major requirements for participating in networks; the goal is to create a riskless environment for storage and data transmissions.

4. Problem Formulation

SDNs are characterized mainly by their use of the OpenFlow protocol, which manages flow traffic intelligently. Lara and Ramamurthy [36] designed a policy-based security scheme for SDNs called OpenSec. OpenSec was implemented to convert security policies into a humanly readable set of OpenFlow messages. All the derived policies are input into OpenSec; these policies are modifiable by end-users. However, malicious end-users can also modify policies to meet their needs, and these human-readable policies can be used by many intruders.

Intruders interfere at every SDN layer; they have even caused link spoofing in links that connect switches [28, 29]. SDNs include link discovery between switches because of the possibility of link failures and spoofing attacks. The Local Fast Reroute (LFR) algorithm was proposed to solve link failures but failed to identify the reason why a link failed. The purpose of link discovery is to minimize link spoofing attacks. To identify link spoofing attacks, the switches were verified using a keyed-hashing for message authentication method [29] that detected attacker forgeries based on generated fake port IDs. Switches are vulnerable not only to link spoofing but also to flow table overloading attacks [5]. Every switch was initially open to this type of attack, which was mitigated using a peer support strategy. Based on this strategy, SDNs were constructed using peer-to-peer topology. However, these topology-based SDNs were subject to flow table attackers who launched their attacks on the switches present at a location. Consequently, peer-to-peer topology failed because it resulted in more switch damage, leading to high switch replacements. Flow table overloading attacks tend to lead to next-level DDoS attacks.

A DDoS attack is a common attack type that can be involved in the failure of any network type. In [6], DDoS attacks were resolved by using a multiqueue SDN controller scheduling algorithm based on a time-slice allocation strategy, in which user traffic is maintained in logical queues based on the number of switches linked to a controller. However, the queue maintenance overhead for each switch makes the system complex; hence, this approach was not applicable for large scale networks. SDNs have some challenges and limitations that previous research work has addressed by focusing on attack identification schemes. However, this paper addresses the problems in SDN network based on security issues, because the design of peer-to-peer topology affects multiple switches, and humanly readable OpenSec policies are most likely to be modified by malicious users during complex queue maintenance operations in multiqueue SDN controllers.

5. Proposed Work

In this section, we first provide an overview of our proposed work and then demonstrate that SDNs are vulnerable to attacks by presenting our classifier along with other methods that can mitigate harmful attacks such as flow table overloading, DDoS, and link spoofing attacks. Flow table overloading attacks and link spoofing target switches, whereas DDoS attacks target controllers.

5.1. System Overview. As discussed in the previous section, SDNs are exposed to numerous threats and vulnerabilities. Our work specifically focuses on inhibiting three attack types: flow table overloading, link spoofing, and DDoS attacks. A multiple-controller SDN design is constructed based on a star topology to mitigate the effects of flow table overloading attacks. The star topology is designed to resolve the limitations inherent to previous peer-to-peer topologies. Due to the topology of peer-to-peer connections, flow table overloading attacks tend to damage all the switches at a particular location because they are connected to each other. We resolve this issue by constructing a star topology from the controller. Then, by applying the Discrete-Time Finite-State Markov Chain (DTMC) model, switch status is conveyed to the controller. Link spoofing attacks between switches are prevented by verifying switches using an unsupervised hashing method. In the controller, anomaly and normal data packets are classified using a hybrid fuzzy and L1-ELM classifier, which works in conjunction with a neural network. Identified anomalies are blocked, and an alert message is sent to the switches to notify them of the presence of an anomaly. In this work, not all the malicious flows enter the controller: attackers detected by switches are blocked by the switches themselves, and attackers who escape detection by hiding their origins from the switches are identified by the controllers.

Header	Action	Statistics	Trust value

Figure 2: Enhanced flow table.

Figure 3 depicts the overall architecture of the SDN in which two controllers are connected to a centralized controller. Switches are connected to the controllers, and hosts are connected to the switches. The switches are deployed in a star topology to limit the effect of a flow table overloading attack to a specific area. Link spoofing attacks are overcome by performing verification using the unsupervised hashing method. Finally, node packets are classified as either suspected or normal. To perform this confirmation, the controller is equipped with a hybrid classifier that can differentiate between normal and anomalous packets.

5.2. Packet Evaluation. Each packet from a user is initially evaluated by the switches based on the trust value of the user. The flow table consists of three fields as shown in Figure 1. For trust value evaluation, we added a "trust value" field to the flow table. This trust value is an estimation based on the user's previous successful transmission and packet drop history. Initially, the trust value field is zero, but it is subsequently incremented or decremented gradually based on the user's performance. This enhanced flow table is depicted in Figure 3.

Figure 2 shows the additional trust value field. Here, one trust value increment is equal to 10 successful packet transmissions, and the trust value is decremented by 1 for every 10 dropped packets. When the trust value exceeds a threshold, the packets are allowed; otherwise, they are rejected. Trust values are automatically updated in the users' storage based on their packet transmissions. An updated trust value cannot be replaced with the same value. It is unnecessary to store packet history here because doing so would increase the space complexity. The trust values are updated with respect to the success or failure of transmissions. This initial packet evaluation supports a higher level of security by allowing only evaluated packets into the network. Furthermore, not only the trust value but also the other flow table fields are verified; when they match, the packets are allowed.

Packets are also validated using the flow table entries present in their headers by considering the packet's features and then reacting based on the packet flow received. The considered features are the source and destination IP addresses, the incoming port number, the source and destination MAC addresses, and the VLAN ID. The reactions to flows are alert, block, and forward. This initial packet evaluation minimizes smaller threats that attempt to slow down the network.

5.3. Mitigating Flow Table Overloading Attacks. A flow table overloading attack targets the switches. Usually, switches have limited space for storing flow table entries. This limitation was noted by attackers, who then sent fake packets to fill the switch flow tables. Consequently, the controllers were forced to install new rules to manage the space problems that can occur in switches. To solve this problem, we introduce the DTMC model to update the switch status information. The status of a switch can be idle, busy, or transmitting; the finite states are determined by the DTMC. The DTMC model assumes that (i) requests arriving from users are independent; (ii) switches have limited flow table capacity and may drop some legitimate packets; and (iii) each switch has a different probability of migrating a flow packet from one switch to another.

Figure 4 illustrates the transition model from one state to another. In this DTMC model, we propose three finite switch states. The Markov parameters involve (i) the model's possible states, (ii) predicting the possible transitions between states, and (iii) determining the probabilities of all state transitions. Using this model, the future states of switches can be predicted based on their current state.

Discrete time is denoted as "n" (i.e., "$n = 0, 1, 2, \ldots, \infty$"); here, arriving requests are modeled using Bernoulli p distributions for random variables. The geometric distributed service time is represented as "q" and "X_n" denotes the state of system after the completion of "n" transitions. To define the probability "P" of switches transitioning from one state to another, we use

$$p_{ij} = P_r\left(X_{(n+1)} = j \mid X_n = i\right), \quad \forall n = 0, 1, 2, \ldots \tag{1}$$

$$\begin{aligned} p_{ij} &= P_r\left(X_{(n+1)} = j \mid X_n = i\right), \ X_{(n-1)} = i-1, \\ X_{(n-2)} &= i-2 \cdots X_0 \quad \forall n \geq 0, \ i, j, i-1, \ldots \in S. \end{aligned} \tag{2}$$

In (2), "S" represents the set of sample spaces. This yields the probability of state "j" occurring after "$n+1$" transitions from the previously obtained transitions. The term "p" represents the user request, and "q" in (4) denotes the service time, which is geometrically distributed. Then, the possible transitions are estimated as follows:

$$P_{r,00} = P\left[p\left(1-p\right)\right] \tag{3}$$

$$P_{r,21} = P\left[q\left(1-p\right)\right] \tag{4}$$

$$P_{r,12} = P\left[p\left(1-q\right)\right] \tag{5}$$

$$P_{r,22} = P\left[pq + \left(1-p\right)\left(1-q\right)\right]. \tag{6}$$

Equation (4) represents a null request with a single completed service, and (6) provides the probability of a switch remaining in the same state. As per this DTMC model, we assume that the system is initially in state "i" at time $n = 0$ and the transition probability is

$$r_{ij,n} = P_r\left(X_n = j \mid X_0 = i\right). \tag{7}$$

In (7), "X_0" denotes the initial state and "X_n" denotes the final state. To determine the destination state, "j," we first determine the state before the destination state. Therefore, "X_0" defines the initial state in which the switch started but not the "$(n-1)$th" state. Therefore, "$r_{ij,0}$" is given as follows:

$$r_{ij,0} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Hence, the one-step probability is "$r_{ij,1} = P_{ij}$." Further, with respect to (5), the transition probability extends back to

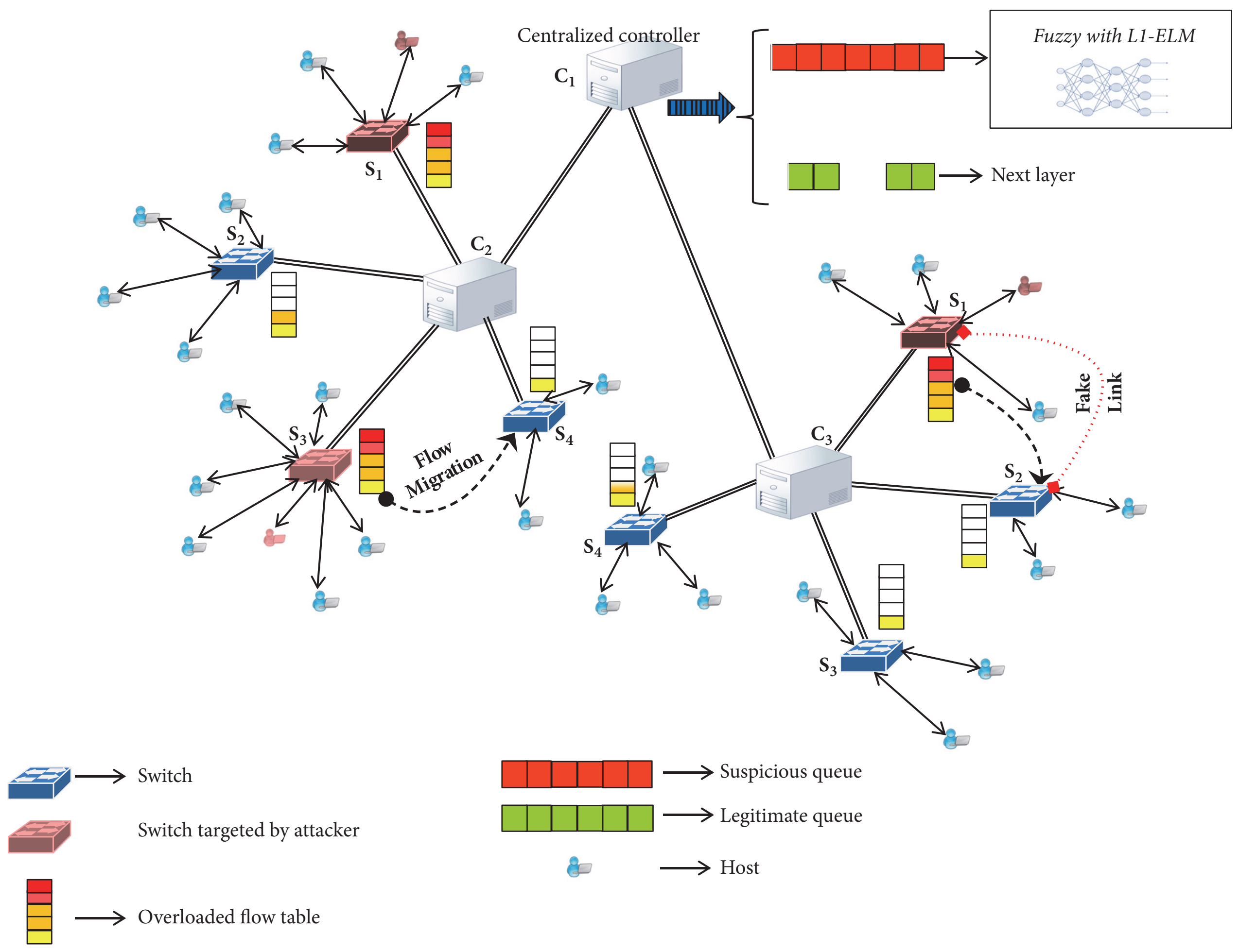


FIGURE 3: Overall SDN architecture.

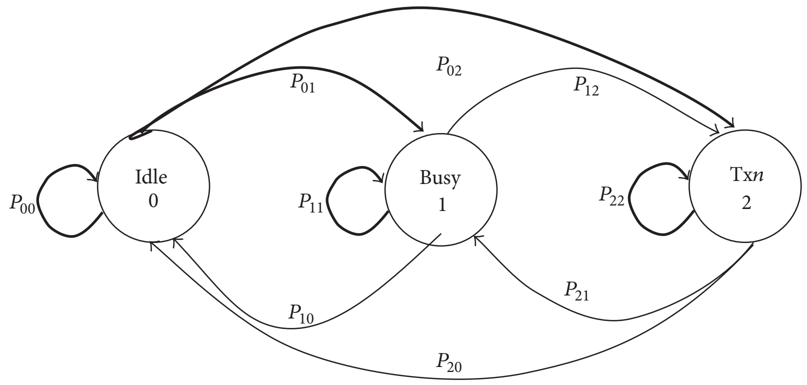


FIGURE 4: DTMC state transition model.

its original state, that is, "$r_{ii=1}$." Next, the total probability for the *"n-step transition"* is estimated as shown in the following:

$$r_{ij,n} = \sum_{a=1}^{m} r_{ia}(n-1) P_{aj}, \quad \forall i, j \in S. \tag{9}$$

Equation (9) is a recursive equation representing that when state "$n-1$" is reached, the major requirement is to determine the final state, which is denoted by "$X_n = j$" and formulated in the following:

$$P(X_n = j) = \sum_{i=1}^{m} P(X_0 = i) r_{ij}(n). \tag{10}$$

Overall, the total numbers of transition states determined after completion of the initial state are as follows:

$$r_{ij,n} = \sum_{i=1}^{m} P_{i1} r_{ij} (n-1), \quad \forall i, j \in S. \tag{11}$$

Using DTMC, we can identify the final states of switches, and these states are periodically updated to the controllers. Using this information, the flows from overloaded switches (i.e., busy switches) are migrated to idle switches. This novel DTMC model minimizes flow table overloading attacks, which are common attack types in SDNs. By reducing the effects of this type of attack in an SDN, the throughput of the entire SDN can be increased.

5.4. Spoofing Attack Prevention. This section discusses the prevention of link spoofing attacks, which occur during flow table migration from one switch to another. To prevent link spoofing attacks, the controllers are responsible for validating the switches and then executing the migration process. Switches are validated using an unsupervised hashing method performed by the controller. Hash functions can be broadly classified into two categories: unsupervised hashing and supervised hashing. An unsupervised hashing method constructs hash functions as "$[h_1, h_2, \ldots, h_b]$" for "*b-bit*" binary hash codes. Hash functions are generated as binary code using unlabeled data. Here, the hash for a switch is constructed using its corresponding identity "S_i." Then, the hash function is formulated as follows:

$$h_r(S_i) = \begin{cases} 1, & \text{if } v_r^T S_i \\ 0, & \text{otherwise,} \end{cases} \quad r = 1, 2, \ldots, b. \tag{12}$$

Before migrating a flow, the controller requests hash values from the switches. Using (12), the switches generate the hash values and send them to the controller. The controller validates the hashes received from the switches and creates a secure link for migrating the flow. To determine the "rth" hash bit of "S_i", the term "v_r" denotes a random vector value for that particular switch, and that random vector value is generated based on utilization of the kernel function [37]. The controller is connected to the switches during deployment; therefore, the generated vector values are shared via wired communication.

The hash is generated using (12); generated hash functions cannot be used by the switch throughout its entire lifetime because always using the same hash function can aid attackers. The generated kernel function is altered when the generated hash function expires. Supervised hashing methods are different than unsupervised hashing methods and require prior knowledge about the hashes to be generated. The steps describe the message exchanges between the switches and the controller to prevent link spoofing attacks (see Algorithm 1).

After validating the hash values, the controllers can prevent the SDN from being subject to link spoofing attacks. Spoofing attacks interrupt the network either to aggregate data or to negatively affect the network by destroying its ability to maintain security. Because of the severity of these attacks in SDNs, many organizations refuse to use SDNs altogether.

```
Step 1. Start
Step 2. Controller updates the states of switches
Step 3. Controller requests hash functions
Step 4. Switches response with hash functions
Step 5. Hash validation
        If (Original hash)
          {
            Creates secure link (Goto Step 6)
        Else
          Invalid switch
          }
Step 6. Begin migrating the flow between switches
Step 7. End
```

Algorithm 1: Steps involved for prevention of spoofing attack.

5.5. Hybrid Classifier. Fuzzy logic is used to consider the significance of three packet features, and because of the huge number of arriving packets, a network neural network is used to provide faster computation. A conventional classification algorithm such as an SVM has high computational overhead and critical kernel selection is required. In contrast, the fuzzy approach minimizes the computational complexity by using simple rules, and, in this paper, the NN is integrated to provide faster classification results. The switches in an SDN are matched with the flow traffic sent by users; a matched flow will react based on the packet header values, and unmatched (a suspected attack flow or a new flow) packets are sent to the controller. The controller maintains two queues: a suspicious queue and a legitimate queue. When suspect flows are sent to the controller, they are placed into the suspicious queue, while normal flows are placed into the legitimate queue and forwarded for further processing in the application layer. All the controllers are equipped with data repositories for storing information such as device IDs, input ports, output ports, and switch IDs.

The hybrid classifier improves the performance over a neural network alone by combining a fuzzy logic system with the L1-Extreme Learning Machine. To do this, the classifier first extracts several features from the flow packets of hosts. The fuzzy logic system considers three features as significant: the source IP address, the source port number, and the destination port number. The port numbers stored in the controller repository provide the initial support for detecting anomalies. The fuzzy logic system is applied as an initial verification using these three packet features. Figure 5 depicts the workings of the hybrid classifier with the combined fuzzy logic system and L1-ELM. Packet features are broadly categorized as basic, content-based, time-based, and connection-based features. For the fuzzy system, the three basic features are considered as a "continuous" type. In our work, flow packet features play a significant role in detecting and mitigating the three attack types. The classification results distinguish normal packets from anomalous packets.

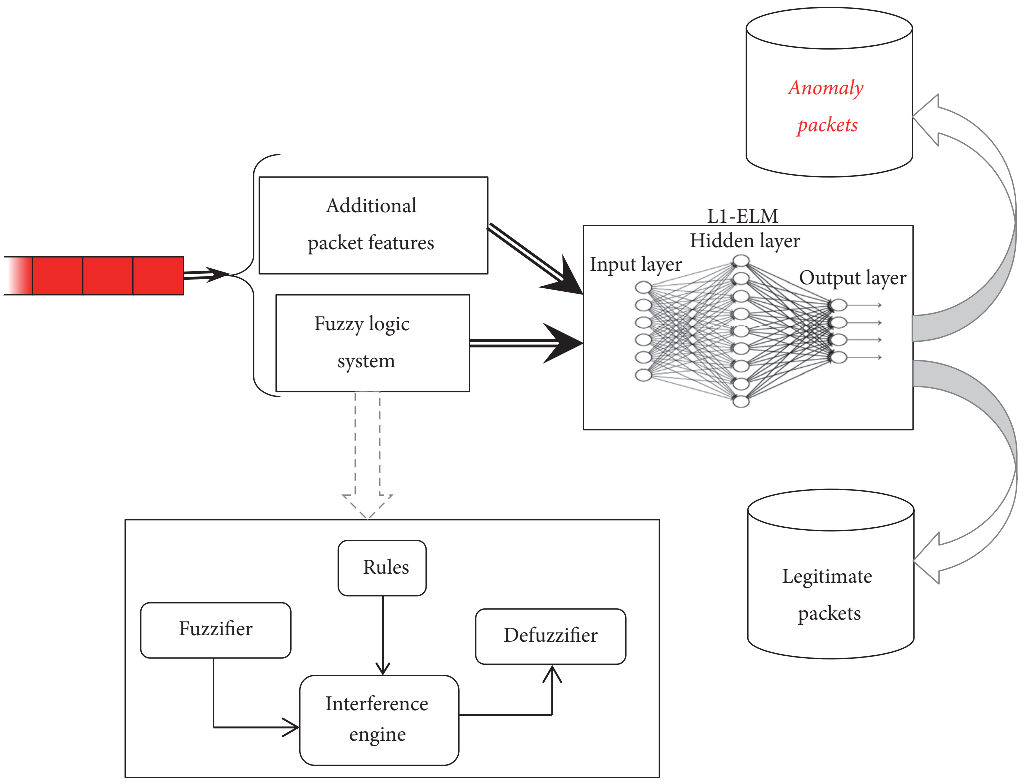


Figure 5: Hybrid classifier.

(i) Fuzzy Logic System. Fuzzy logic is used to perform an initial verification of flows using the three significant packet features described above. When all three features are legitimate, the final output is "HIGH"; otherwise, the output is "LOW" for all cases.

(1) Source IP Address. An IP address is simply an identifier allocated to each device linked to the Internet via switches and routers. The source IP address is significant; hence, it should be verified to avoid vulnerabilities to IP spoofing attacks. Such spoofs interfere with links and can participate as source devices with fake source IP addresses.

(2) Source Port. A port is defined as a communication endpoint. The port number is related to the IP address of a particular host. Port numbers are 16-bit unsigned integers that range from 0 to 65,535. The source port field defines whether to send the packet with a UDP or TCP header.

(3) Destination Port. This field of the packet header defines the destination application.

These three basic unique features can identify the data streams with which the hosts are communicating. The results obtained from the fuzzy logic system are provided to the subsequent process along with additional packet features. The crisp output obtained from the fuzzy logic system is input to the L1-ELM for classification, as shown in Table 1.

The crisp output obtained fuzzy is denoted as Z which is either HIGH or LOW; this result is given as $z \in Z$. z is the current output for a given set of inputs that belong to the fuzzy output Z. This output is input to the L1-ELM and used to formulate its output. Normal packet features will result in a HIGH crisp value, while anomalous packet features produce LOW crisp values that tend to cause the packet to be dropped.

Table 1: Crisp output obtained from fuzzy is input into the L1-ELM for classification.

Specification	Feature	Type	Description
	Src_IP addr	Continuous	IP address of source
Fuzzy logic	Src_port	Continuous	Port number of source
	Des_port	Continuous	Port number of destination

(ii) L1-ELM. The L1-ELM is a neural network designed with "m" hidden layer nodes that produce an output based on weighted values. The output is formulated as follows [38]:

$$f(x) = h(x)\beta \quad (13)$$

$$h(x)\beta = \sum_{i=1}^{m} \beta_i h_i(x), z. \quad (14)$$

By applying (14) to (13), we obtain the output function. The terms used in the above equations are as follows: z denotes the obtained results from the fuzzy logic system based on the three significant features, $\beta = [\beta_1, \ldots, \beta_m]^T$ denotes the vector of the output weights of the hidden layer, "$x \in \mathfrak{R}^n$" represents the "n" input variables, and "$h_i(x)$" defines the "ith" activation function corresponding to the weight vector of the input layer, "w_i."

Table 2: Limitations and problems of the state-of-the-art approaches.

State-of-the-art	Problems	Limitations
Peer support strategy	(i) Peer topology (ii) Attacks target specific locations	(i) Peer support strategy is less effective (ii) Single controller
Multiqueue SDN controller scheduling algorithm	(i) Separate queues for each switch (ii) Requires effective maintenance system	(i) Single controller (ii) Complex to manage requests from switches

The L1-ELM requires a training phase based on prior packets results. Here we have used only significant features that do not require repeated training; therefore, prior labeled packets are used as training data. In contrast, the testing data is the data obtained from the user's current flow. The input variables, "n," include packet features such as the destination IP address, protocol type, Ethernet source, Ethernet destination, and service. Then, the hidden layer output matrix is

$$H = \begin{pmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{pmatrix} = \begin{pmatrix} h_1(x_1) & \cdots & h_m(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_1) & \cdots & h_m(x_1) \end{pmatrix}. \quad (15)$$

By solving a least-squares problem, we can obtain the optimal solution as follows.

$$\beta^* = \frac{1}{H}Y. \quad (16)$$

In (16), the term "$1/H$" denotes the inverse matrix of "H." Then, "$Y = [y_1, \ldots, y_N]^T$" is the target vector (i.e., the output values). Finally, this process classifies packets as either normal or anomalous. In accordance with the considered packet features, abnormal features are predicted and identified as belonging to anomalous packets. An abnormality in a feature such as a change in type is detected as an anomaly.

6. Performance Evaluation

In this section, we discuss the performance of our proposed work. This section consists of subsections that describe the simulation setup, flow analyses, and a comparative analysis. We demonstrate that our proposed work effectively validates flows in an SDN to maintain a secure SDN environment. We perform a comparative analysis with two peer strategies for SDNs: the strategy from [4] and the multiqueue SDN controller scheduling algorithm [5].

Table 2 lists the limitations and problems that exist in the state-of-the-art works that are overcome in our work through SDN design and construction. We provide comparative results and a discussion.

6.1. Simulation Setup. For this work, we built simulations in the OMNeT++ environment. The SDN was built using OpenFlow controllers, switches, and hosts. This simulation tool was programmed using the C++ language and installed on a computer running a Windows 7 operating system with a CPU running at 3 GHz. The OMNeT++ simulator has good performance and supports a graphical user interface. Our implementation for an SDN environment was designed using the "OpenFlow INET Framework" which includes an OpenFlow module specifically designed for constructing SDN environments in OMNeT++.

Table 3: Key parameters for our simulation.

Parameters	Value
Number of controllers (OpenFlow)	3
Number of switches	8
Number of hosts (per switch)	4
(Hosts) topology	Star topology
Flow table size	8,000 entries
Buffer capacity	10
Trust value	100 (maximum)
Number of flows	116
Average attack rate	25 requests per seconds
Simulation time	300 seconds
Flow timeout	2 seconds

Figure 6 shows the simulation setup in OMNeT++. The network is designed as a multicontroller SDN. It consists of OpenFlow switches connected to controllers in the SDN. We assign a centralized controller to manage the two other controllers linked to it. Hosts are connected to controllers via switches. We apply our proposed novel methodologies to this setup using the simulation parameters shown in Table 3. Malicious traffic packets in this proposed SDN environment are generated according to the simulation time; both legitimate packets and malicious packets are allowed into the switches to measure the performance of proposed algorithms. Based on the temporal values, malicious traffic with fake packet features is generated. This simulation uses the hybrid classifier that integrates the *NN*; the number of hidden layers used here range from 5 to 10. The number of hidden layers can be altered according to the process.

Table 3 lists the significant parameters required to perform our simulation. Using these specifications, our overall network design is set up as shown in Figure 6. The flow packets from the hosts can contain attacks, and these attacks are identified by the controllers.

6.2. User Flow Validation. Our work concentrates on three main attack types: flow table overloading, spoofing, and DDoS attacks. We mitigate and detect these attacks based on packet features. In this section, we analyze the packet flows involved in SDN. The packet flow is analyzed by the switches based on the entries saved to the flow table as shown in Table 4.

As shown in Table 5, host packet flows are evaluated by the switches and controllers to safeguard the SDN and make it less vulnerable to attacks. Each user flow is validated by the switches using the packet features. The results of our work are tabulated in Table 5.

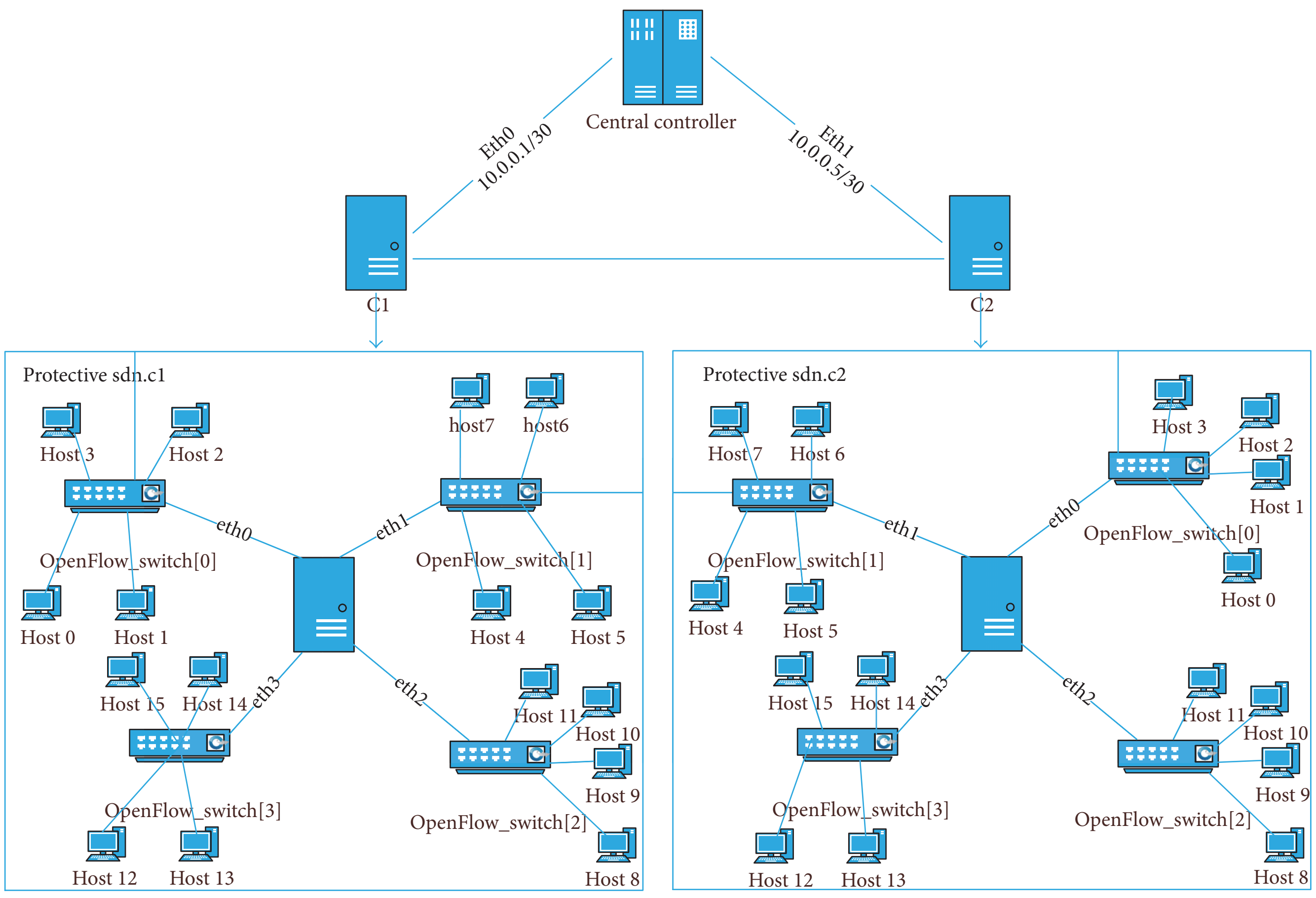


Figure 6: Simulation setup.

Table 4: Flow table rules.

Flow	Attacks	Action
(i) Ethernet source & destination (ii) Source & destination IP (iii) Source & destination port (iv) Protocol type	(i) Flow table overloading attack (ii) DDoS attack (iii) Spoofing attack	(i) Alert (ii) Block (iii) Forward

Table 5: Obtained flow analyses result.

Flow metrics	Values
Total number of flows	116
Flows reached destination	34
Sources blocked	75
Total anomalies present	82
Detection and mitigation time	2-3 seconds

6.3. *Comparative Analysis.*

In this section, the effectiveness of our proposed work is studied using graphical plots that show comparisons. We analyze five factors; three are based on holding time and the remaining two factors are delay and failure ratio.

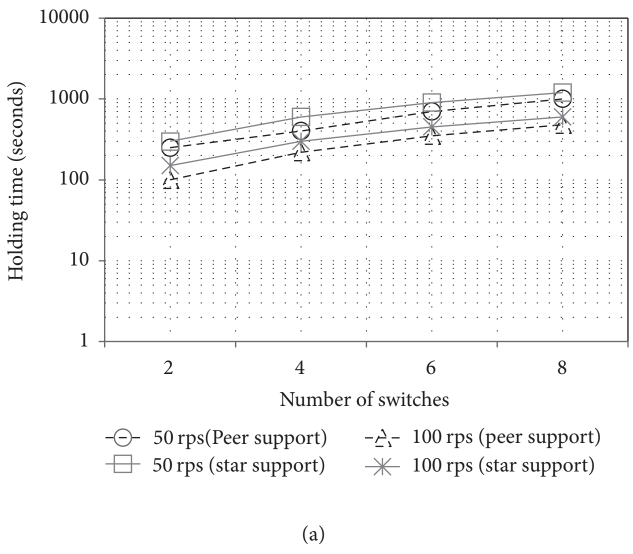
These metrics are plotted against the number of switches involved in our SDN.

6.3.1. *Effectiveness of the Star Topology in Mitigating Flow Table Overloading Attacks.*

In previous works, flow table overloading attacks were simulated based on a peer support strategy. Here, we mitigate this attack using star topology support and a migration procedure. To determine the efficiency of this attack in our SDN, we simulated the attack process at an average attack rate. Holding time is used to analyze the capacity of our SDN under a flow table overloading attack. The graphical charts plot the number of switches against holding time. Here, the holding time is significant, because it varies with respect to changes in workload.

Figure 7 shows the comparative results for holding time with respect to requests per second and peer strategy. From this result, we can see that the holding time increases linearly and is higher compared with the previous peer strategy for SDN. Here, the workload and average flows have smaller impacts on holding time. However, even after applying the peer strategy in the SDN, the holding time shows a much smaller increase, which means that the flow table overloading attacks affected the switches under the peer strategy.

In both Figures 7(a) and 7(b), the proposed star topology-based SDN obtains better results than does the peer strategy, indicating that it minimizes the effects of flow table overloading attacks. This result occurs because our approach uses the idle flow space available in the switches, countering the attack by migrating flows from busy to idle switches. Thus, this



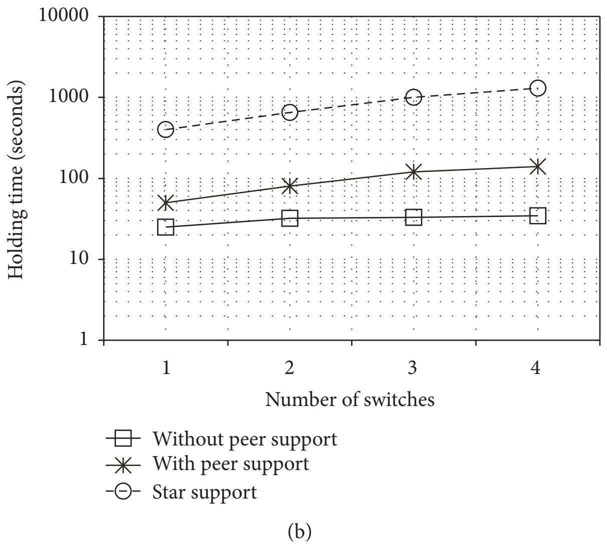


Figure 7: Holding time based on (a) requests per second (RPS) and (b) peer strategy.

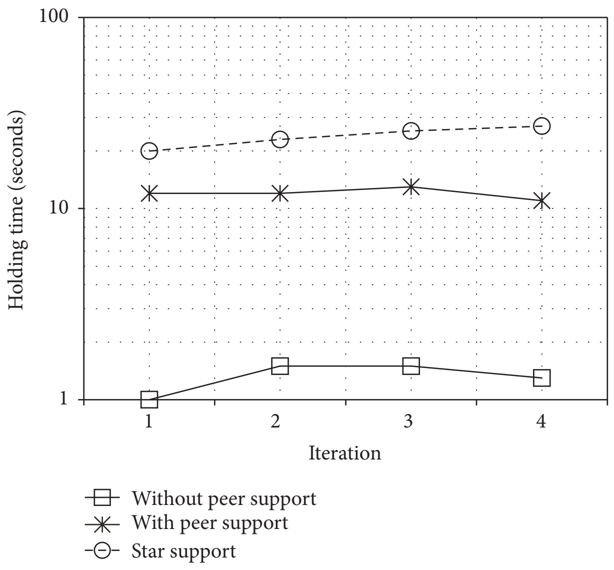


Figure 8: Holding time effectiveness under flow table overloading attack.

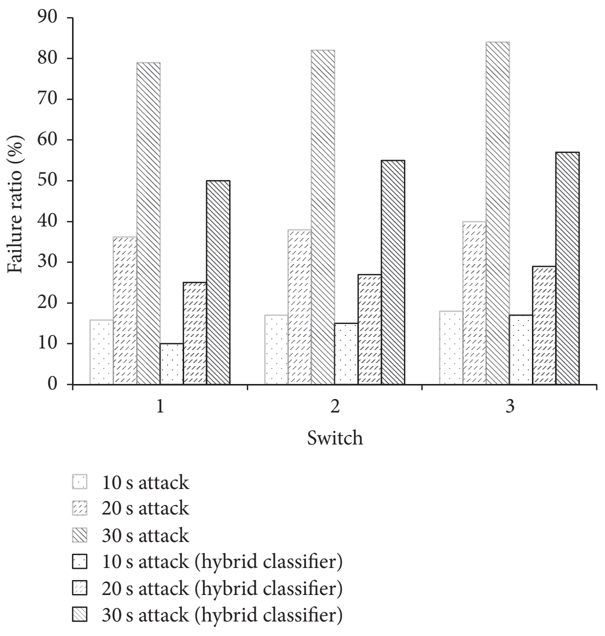


Figure 9: Comparison of switch failure ratios.

design gives an SDN a remarkable level of protection from flow table overloading attacks. In both Figures 7 and 8, the holding times are measured in seconds and include the delays between switches and controllers. However, here, we can ignore those delays because they are very small (i.e., measured in milliseconds). These millisecond delays have little impact on the holding time and thus can be ignored.

We evaluated the performance of our proposed SDN compared to previous works that use the peer support strategy. Usually, this attack first defeats a specific targeted switch and then moves on to the next switch. To evaluate this attack in our simulation, we designed attackers that continuously send flow traffic to switches; then, we diagnose the switch performances.

Figure 8 depicts the holding time comparison between the peer support approach and our star support approach. The star support approach allows a higher holding time for switches. Our SDN design with star topology support is approximately 5 times higher than that of the SDN using the peer support strategy when the same number of switches are used in the SDN. Overall, the star topology is effective against flow table overloading attacks.

6.3.2. Effectiveness of the Proposed SDN Design against Other Attacks. To evaluate our proposed SDN design against other attacks, we must analyze the performance of attacks on the network over a certain time period. Attackers lead to higher switch failure ratios (based on the exposure of each switch to the attack).

Figure 9 shows a comparison plot of the failure ratio with respect to switches. We analyzed the attacks that focus on a

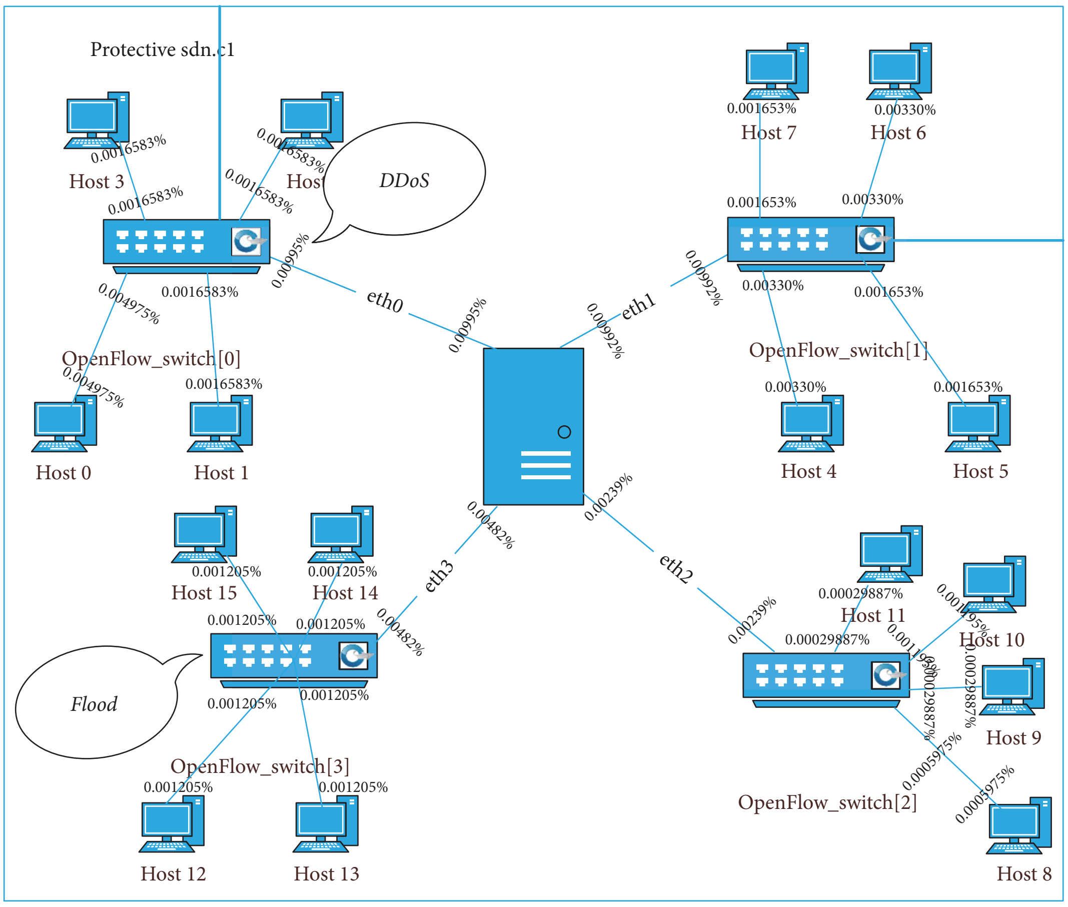


Figure 10: Attacks over switches in our SDN design.

single switch in an SDN. Our work results in a reduced failure ratio compared to previous works on SDNs.

The effects of flooding and DDoS attacks on our SDN design are shown in Figure 10. Further action is taken by switches: when a new flow is suspect, it is forwarded to the controller, where the packets are classified as anomalous or normal. When anomalous packets are found, all the switches are alerted with information about the attacker.

In Figure 11, Switch 1 and Switch 2 have large differences in delay. Switch 1 is subjected to a larger number of packets from the attacker, whereas Switch 2 has not received much traffic from the attacker. Because the traffic occurring at each switch differs according to the number of attackers involved, delay is a significant metric: the delay increases as the number of attackers increases or as their activity increases. Attacks targeting the switches will lead to increased delay in processing legitimate flow traffic. As discussed, our approach migrates flows to balance the flow table entries even when an attack is not occurring. Figure 11 illustrates the minimized delay of our approach compared with that of previous approaches. The delay minimization occurs for two reasons: flow migration and the two-queue maintenance scheme used by the controller for all the switches.

Finally, to our knowledge, this paper is the first work that concentrates on mitigating and preventing three different

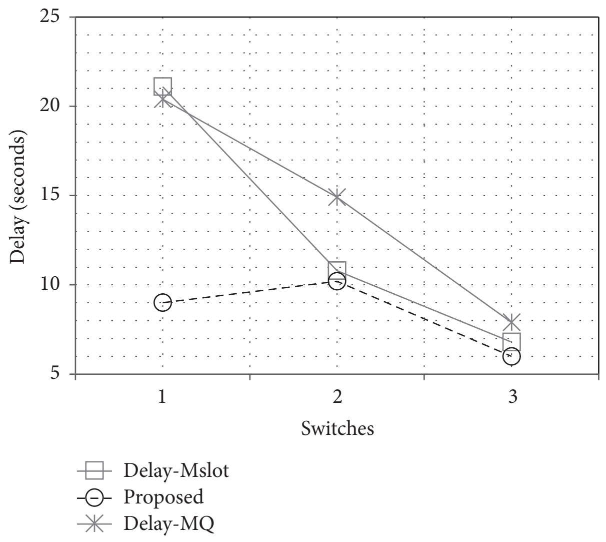


Figure 11: Comparison of delay.

attack types by validating user traffic flows. Our research work addressed the challenges to and the issues and limitations of SDNs that existed in previous research works. We have moved SDNs a step forward with this paper, which solves the performance issues resulting from common attacks.

7. Conclusion

In this paper, we have addressed the challenging task of improving security for SDNs. This work addresses three common attack types: flow table overloading attacks, DDoS attacks, and spoofing attacks. We proposed a multicontroller SDN design with star topology support to secure the network environment by validating user flows. We used the DTMC model in the controller to update switch states and an unsupervised hashing method to avoid spoofing attacks. Finally, we use a hybrid classifier in the controllers established by combining a fuzzy logic controller with the L1-ELM classifier, which executes over a neural network and can differentiate normal flows from anomalous flows received from users. This proposed SDN design is thoroughly analyzed using a simulated network environment. The extensive results, which analyze network performance improvements, indicate that our SDN solves the security issues addressed in this study. We plan to extend this work in the future using cryptography to further enhance user data security. And also in future we have planned to validate the network flows using this work in a large environment.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper was supported by the National Key Research & Development (R&D) Plan of China under Grant no. 2017YFB0802205, the National Science Foundation of China under Grant no. 61672249, and the Shenzhen Fundamental Research Program under Grant no. JCYJ20170413114215614.

References

[1] M. C. Dacier, H. König, R. Cwalinski, F. Kargl, and S. Dietrich, “Security challenges and opportunities of software-defined networking,” *IEEE Security & Privacy*, vol. 15, pp. 96–100, 2017.

[2] D. B. Rawat and S. R. Reddy, “Software defined networking architecture, security and energy efficiency: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 325–346, 2017.

[3] K. Benzekki, A. El Fergougui, and A. E. Elalaoui, “Software-defined networking (SDN): a survey,” *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[4] F. X. A. Wibowo, M. A. Gregory, K. Ahmed, and K. M. Gomez, “Multi-domain software defined networking: research status and challenges,” *Journal of Network and Computer Applications*, vol. 87, pp. 32–45, 2017.

[5] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, and J. Shen, “Defending against flow table overloading attack in software-defined networks,” *IEEE Transactions on Services Computing*, no. 99, 2016.

[6] Q. Yan, Q. Gong, and F. R. Yu, “Effective software-defined networking controller scheduling method to mitigate DDoS attacks,” *IEEE Electronics Letters*, vol. 53, no. 7, pp. 469–471, 2017.

[7] T. Xu, D. Gao, P. Dong, H. Zhang, C. H. Foh, and H. Chao, “Defending against new-flow attack in SDN-based internet of things,” *IEEE Access*, vol. 5, pp. 3431–3443, 2017.

[8] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, “Lineswitch: tackling control plane saturation attacks in software-defined networking,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1206–1219, 2017.

[9] Q. Yan and W. Huang, “A DDoS detection and mitigation system framework based on spark and SDN,” in *Proceedings of the International Conference on Smart Computing and Communication*, vol. 10135, pp. 350–358, 2016.

[10] N. M. Sahri and K. Okamura, “Collaborative spoofing detection and mitigation—SDN based looping authentication for DNS services,” in *Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference, COMPSAC*, pp. 565–570, Atlanta, GA, USA, June 2016.

[11] F. Shahzad, M. A. Khan, S. A. Khan, S. Rehman, and M. Akhlaq, “AutoDrop: automatic DDoS detection and its mitigation with combination of Openflow and Sflow,” in *Proceedings of the International Conference on Future Intelligent Vehicular Technologies*, vol. 185, pp. 112–122, 2016.

[12] S. Zerkane, D. Espes, P. Le Parc, and F. Cuppens, “Vulnerability analysis of software defined networking,” in *Proceedings of the International Symposium on Foundations and Practice of Security*, vol. 10128, pp. 97–116, 2016.

[13] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, “DDoS attack protection in the era of cloud computing and Software-Defined Networking,” *Computer Networks*, vol. 81, pp. 308–319, 2015.

[14] H.-z. Wang, P. Zhang, L. Xiong, X. Liu, and C.-c. Hu, “A secure and high-performance multi-controller architecture for software-defined networking,” *Frontiers of Information Technology & Electronic Engineering*, vol. 17, pp. 634–646, 2016.

[15] H. Li, P. Li, S. Guo, and A. Nayak, “Byzantine-resilient secure software-defined networks with multiple controllers in cloud,” *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 436–447, 2014.

[16] M. Antikainen, T. Aura, and M. Särelä, “Spook in your network: attacking an SDN with a compromised openflow switch,” in *Proceedings of the Nordic Conference on Secure IT Systems*, vol. 8788, pp. 229–244, 2014.

[17] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, “Advanced study of SDN/OpenFlow controllers,” in *Proceedings of the 9th central & eastern european software engineering conference in*, Russia, 2013.

[18] B. Xiong, K. Yang, J. Zhao, W. Li, and K. Li, “Performance evaluation of OpenFlow-based software-defined networks based on queueing model,” *Computer Networks*, vol. 102, pp. 172–185, 2016.

[19] R. Zhou, Y. Lai, Z. Liu, and J. Liu, “Study on authentication protocol of SDN trusted domain,” in *Proceedings of the 2015 12th IEEE International Symposium on Autonomous Decentralized Systems, ISADS 2015*, pp. 281–284, Taiwan, March 2015.

[20] A. AlEroud and I. Alsmadi, “Identifying cyber-attacks on software defined networks: an inference-based intrusion detection approach,” *Journal of Network and Computer Applications*, vol. 80, pp. 152–164, 2017.

[21] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, “Security in software defined networks: a survey,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2317–2346, 2015.

[22] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, “Research trends in security and DDoS in SDN,” *Security and Communication Networks*, vol. 9, no. 18, pp. 6386–6411, 2016.

[23] S. Shin, L. Xu, S. Hong, and G. Gu, "Enhancing network security through software defined networking (SDN)," in *Proceedings of the 25th International Conference on Computer Communications and Networks, ICCCN 2016*, USA, August 2016.

[24] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A Survey on the security of stateful SDN data planes," *IEEE Communications Surveys &Tutorials*, vol. 19, no. 3, 2017.

[25] D. Satasiya, R. Raviya, and H. Kumar, "Enhanced SDN security using firewall in a distributed scenario," in *Proceedings of the 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016*, pp. 588–592, India, May 2016.

[26] A. Kaur and A. Bhandari, "Detection and mitigation of spoofing attacks by using SDN in LAN," in *Proceedings of the Sixth International Conference on Soft Computing for Problem Solving*, vol. 547 of *Advances in Intelligent Systems and Computing*, pp. 240–247, Springer, Singapore.

[27] N. I. Mowla, I. Doh, and K. Chae, "An efficient defense mechanism for spoofed IP attack in SDN based CDNi," in *Proceedings of the 2015 International Conference on Information Networking, ICOIN 2015*, pp. 92–97, Cambodia, January 2015.

[28] X. Zhang, Z. Cheng, R. Lin, L. He, S. Yu, and H. Luo, "Local fast reroute with flow aggregation in software defined networks," *IEEE Communications Letters*, vol. 21, no. 4, pp. 785–788, 2017.

[29] T.-H. Nguyen and M. Yoo, "Analysis of link discovery service attacks in SDN controller," in *Proceedings of the 2017 International Conference on Information Networking (ICOIN)*, pp. 259–261, Da Nang, Vietnam, January 2017.

[30] Z. Zhao, F. Liu, and D. Gong, "An SDN based hopping multicast communication against DoS attack," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 4, 2017.

[31] Q. Wei, Z. Wu, K. Ren, and Q. Wang, "An Openflow user-switch remapping approach for DDoS defense," *KSII Transactions on Internet and Information Systems*, vol. 10, no. 9, pp. 4529–4548, 2016.

[32] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," in *Proceedings of the 2015 International Conference on Information Networking, ICOIN 2015*, pp. 309–311, Cambodia, January 2015.

[33] Z. Wu, Q. Wei, K. Ren, and Q. Wang, "A dynamic defense using client puzzle for identity-forgery attack on the south-bound of software defined networks," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 2, pp. 846–864, 2017.

[34] A. M. AbdelSalam, A. B. El-Sisi, and V. Reddy, "Mitigating ARP Spoofing Attacks in Software-Defined Networks," in *Proceedings of the International conference on Computer Theory and Applications (ICCTA)*, Egypt, 2016.

[35] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: new attacks and countermeasures," in *Proceedings of the Network and Distributed System Security Symposium*, Internet Society, San Diego, Calif, USA, February 2015.

[36] A. Lara and B. Ramamurthy, "OpenSec: policy-based security using software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 13, no. 1, pp. 30–42, 2016.

[37] B. Demir and L. Bruzzone, "Hashing-based scalable remote sensing image search and retrieval in large archives," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 2, pp. 892–904, 2016.

[38] Y. Wang, D. Li, Y. Du, and Z. Pan, "Anomaly detection in traffic using L1-norm minimization extreme learning machine," *Neurocomputing*, vol. 149, pp. 415–425, 2015.