Hindawi
Security and Communication Networks
Volume 2021, Article ID 6667922, 16 pages
https://doi.org/10.1155/2021/6667922

Research Article

A Table Overflow LDoS Attack Defending Mechanism in Software-Defined Networks

Shengxu Xie, Changyou Xing, Guomin Zhang, and Jinlong Zhao

Command & Control Engineering College, Army Engineering University of PLA, Nanjing, China

Correspondence should be addressed to Changyou Xing; changyouxing@126.com

Received 10 October 2020; Revised 3 January 2021; Accepted 12 January 2021; Published 29 January 2021

Academic Editor: Petros Nicopolitidis

In order to achieve requirements such as fast search of flow entries and mask matching, OpenFlow hardware switches usually use TCAM to store flow entries. Limited by the capacity of TCAM, the current commercial OpenFlow switches can only support hundreds of thousands of flow entries, which makes SDN network using OpenFlow hardware switches vulnerable to the threat of flow table overflow attack. Among them, low-rate DoS (LDoS) attack against table overflow poses a serious threat to SDN networks due to its high attack efficiency and concealed flow, and it is also difficult to detect. In this regard, this paper analyzed two types of LDoS attack flow against table overflow and proposed an attack detection and defense mechanism named SAIA (Small-flow Analysis and Inport-flow Analysis) through the design of table overflow prediction and flow entries deletion strategy. Experiments conducted through the SDN network environment showed that SAIA can effectively detect and suppress LDoS attack flows in the flow table in large-scale network conditions and verified that the deployment of SAIA is lightweight. At the same time, SAIA implemented the flow entry deletion strategy based on LRU when the flow table overflows in a nonattack situation, which further enhances the stability of the network.

1. Introduction

Software-Defined Networking (SDN) features such as centralized control, separation of forwarding and control, and network programmable make network management simple and flexible. Due to its good combination with various cloud services, SDN has been widely deployed in recent years, and its security has become the focus of attention in the industry. However, the introduction of the new architecture also brings a lot of new problems [1], among which security is the most noteworthy, including the security of the data plane [2, 3]. The data plane not only will be affected by existing attacks in traditional networks (such as DDoS attacks [4]) but will also bring new types of attacks due to its own architecture, the most typical of which are flow table overflow attacks [5, 6].

In order to meet the requirements of flow entry mask matching, SDN hardware switches that support the OpenFlow protocol all use Ternary Content Addressable Memory (TCAM) to store and search flow entries. However, the high-power consumption and high price of TCAM make the capacity of TCAM in SDN switches very limited. Currently, in the existing commercial SDN switches, the number of flow entries that can be stored is only hundreds of thousands [7]. However, in order to achieve fine-grained flow management in an SDN network, many flow entries need to be installed to handle the ever-increasing traffic, which is prone to the problem of insufficient flow table space. At the same time, when the flow table space is insufficient, SDN needs to carry out extra processing for the new flow arriving in the network, which makes the network performance decline sharply. Therefore, the flow table overflow problem in SDN has been extensively studied, including the optimization of the flow table overflow problem [8] and the mitigation of the flow table overflow attack [9, 10].

Among various attacks against SDN flow table overflow, LDoS attacks have not received extensive attention. LDoS attackers send attack flow to SDN network at a very low rate and density by means of OpenFlow protocol, which sets two timeouts for each flow entry. The SDN controller will

generate corresponding flow entries for the attack flows of these different headers and install them in the switch. These flow entries will occupy the flow table space until the timeout period expires. Since the attacker can easily detect the size of flow table space of the switch [12] and the timeout period of the flow entries [13], they easily launch LDoS attacks against the flow table, which makes the flow table space overflow all the time, thus causing a sharp decline in network performance.

To solve the impact of LDoS attacks against flow table overflow on SDN networks, this paper first analyzes the two typical types of low-rate attack flows and proposes the LDoS attack detection and defense mechanism SAIA based on small-flow and inport-flow statistical analysis, which includes algorithms such as flow table overflow prediction, attack flow identification, and flow entry deletion. The experimental results show that SAIA can be deployed lightly and has a better detection effect in larger-scale topologies. The main contributions of the paper are as follows:

(i) Two types of LDoS attack flow against table overflow are analyzed and their impact is evaluated on the performance of SDN network through experiment

(ii) Designed attack detection and defense system SAIA is based on small-flow and inport-flow analysis to mitigate the table overflow LDoS attacks

(iii) Through experiment, the effectiveness of the SAIA system against table overflow LDoS attacks is verified, and at the same time this verifies that SAIA can effectively mitigate the problem of table overflow under nonattack conditions

The rest of the paper is organized as follows. Section 2 introduces the background of SDN, OpenFlow and LDoS attacks, and the motivation of the research on table overflow LDoS attacks. Section 3 analyzes the mechanism, model, and defensive difficulties of table overflow LDoS attacks. Section 4 designs and implements a detection and defense system SAIA against table overflow LDoS attacks. Section 5 evaluates the SAIA through experiments. Finally, Section 6 concludes the paper.

2. Background and Motivation

2.1. SDN and OpenFlow. SDN is defined as a new network architecture with separation of forwarding and control and software programmable. It abstracts the traditional control plane function from each network node and reconstructs the original distributed control network architecture into a centralized control network architecture. Its basic architecture is shown in Figure 1, which mainly includes three planes and two interfaces: application plane, control plane, and data plane, and northbound interface and southbound interface.

Among the many southbound interface protocols, OpenFlow has become an established standard. As a communication protocol, OpenFlow connects peer-to-peer data plane forwarding devices and SDN controllers. Its

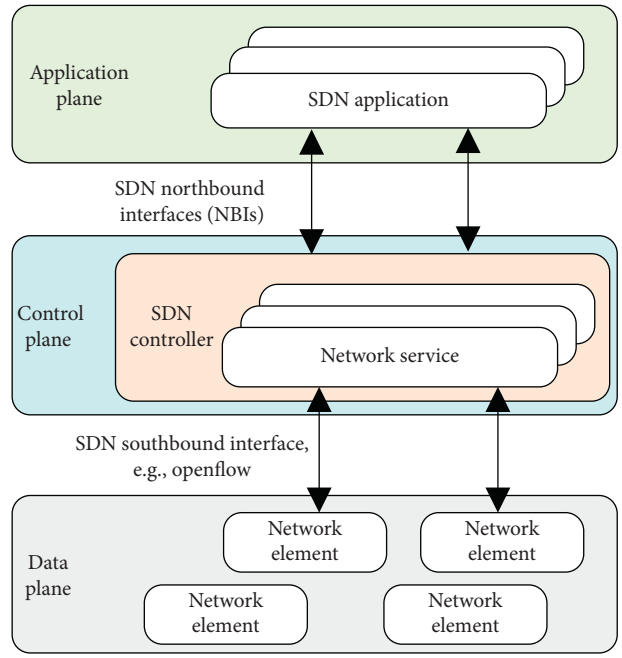

Figure 1: Basic architecture of SDN.

architecture is shown in Figure 2. The SDN switch, supporting OpenFlow v1.3, mainly consists of three parts: group table, pipeline, and secure channel. The flow table on the pipeline can be regarded as a collection of a set of policy entries (i.e., flow entries), and the data packet can jump between multiple flow tables on the pipeline for matching flow entries. Group table is used to aggregate different flow entries with the same instruction to improve the resource utilization of flow table. The secure channel is the interface between the OpenFlow switches and the controller to protect the communication between them from interference. So far, OpenFlow has developed from v1.0 to v1.5, and there are more and more field types in the match field to achieve more fine-grained network management and control [14].

Figure 3 describes the basic process of the network flow forwarding process in the OpenFlow-based SDN environment. Pkt_0 and Pkt_1 are the first two packets of the same flow (with the same header field). When the OpenFlow switch S receives the first data packet of the new flow sent by the Host1 to the Host2, the processing of the flow is shown by the red arrow in the figure. First, the switch S receives the new flow and performs matching in the flow table; after the corresponding flow entry is not matched, the *packet_in* message encapsulating the header information of the flow is sent to the controller; after receiving the *packet_in* message, the controller generates the corresponding flow rule according to the network management policy and sends the *flow_mod* message to switch S to install the corresponding flow entry of the flow rule; finally, switch S carries out forwarding or other operations according to the flow entry. When the switch receives a data packet of an old flow (the corresponding flow entry already installed in the switch) sent by the Host1 to the Host2, the forwarding process of the data packet is shown by the green arrow in the figure. That is,

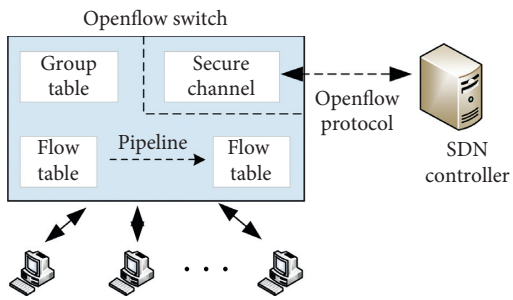

Figure 2: Architecture of OpenFlow v1.3.

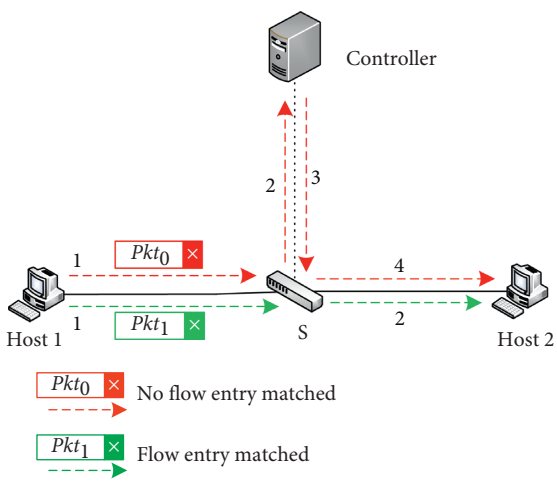

Figure 3: Forwarding process of flow in SDN environment.

the switch matches the flow entry corresponding to the packet header and carries out forwarding and other operations on the packet according to the action of the flow entry.

2.2. LDoS Attack. The LDoS attack is an attacker's use of security vulnerabilities in the adaptive mechanism of the network protocol. By sending periodic short pulse data streams, the network has always been in an unstable state, which has an important impact on the normal operation of the network.

At present, LDoS has been extensively studied but most of the research on LDoS attacks is still in the traditional network [15]. For example, LDoS attacks against TCP timeout retransmissions, which send pulsed traffic to cause TCP packets to be discarded due to buffer overflow during network transmission, and the TCP connection will be dropped by periodic attack traffic [16, 17]. Low-rate HTTP server DoS attacks send a large data slowly as request or receiving the response from the server slowly using very small TCP window size in order to keep the connection, so that legitimate user link requests are discarded [18]. Low-rate DoS attacks against application servers send enough requests to occupy the service queue and the legitimate service request was discarded [19, 20].

In the research of LDoS attacks in SDN networks, most of them are based on the advantages of SDN network centralized control and flexible data collection, which defends against LDoS attacks against terminal servers [21, 22]. Although there have been some researches on LDoS attacks against SDN architecture [23], the overall situation is relatively small.

2.3. Table Overflow and Its Impact. Here, let us first introduce the TCAM used by SDN hardware switches. During TCAM lookup, all data in the entire table entry space are queried at the same time. The lookup speed is not affected by the amount of data in table space, and a lookup can be completed in one clock cycle. Compared with typical memory search algorithms (such as linear search, binary search, and hash table), the TCAM circuit compares all stored data in parallel, which effectively shortens the search time. Meanwhile, the state of TCAM at each bit contains not only "0" and "1," but also another "Don't care" state implemented by the mask. Thanks to the third state of TCAM, it can conduct both accurate match search and fuzzy match search, so it is widely used in SDN hardware switches to meet the requirements of flow entries in OpenFlow.

Unfortunately, TCAM has high-power consumption and high price, which greatly limits the flow table capacity in SDN switches. At present, only hundreds of thousands can be stored at most. As an important resource of SDN network, SDN switch flow table space is easy to cause network performance degradation due to overflow either under normal condition or under an attack [24]. When a new flow reaches a switch that has already saturated the flow table space, the controller has three processing methods: (1) drop the new flow directly; (2) forward the new flow through *packet_out* message; and (3) select a flow entry from the switch to delete and then install the corresponding flow entry for the new flow. The first method will cause many packets losses in the network, resulting in a sharp decline in network performance. The second method will cause a large number of flows to be forwarded through the controller, causing a serious burden on the controller, and significantly increases the network delay. The third method will cause the original flow entry to be deleted, and the average latency of the flow increases due to frequent updates of the flow entries. For example, the experiments in Section 5.1 of this paper show that the flow table overflow caused by the attack causes the delay of packets in the network to increase from almost 0 ms to about 50 ms, while the change of throughput drops from 1 Gb/s to 10 Mb/s. Such a network situation is unacceptable.

At present, most researches on the flow table overflow of OpenFlow switch focus on how to solve the flow table overflow problem caused by large traffic flows in a normal network environment. For example, set an appropriate timeout value of flow entries to reduce the total number of flow entries in the flow table space [25–27]; balance the utilization of flow table space by redirecting flows from switches with high flow table space utilization to switches with sufficient free flow table space [28–30]; aggregate flow entries [31–33]; find out the most suitable flow entry to delete when the flow table space is in its saturation status [34–36].

The above researches are all mitigation mechanisms for the table overflow under the normal network environment. In addition, under the attack of OpenFlow switch, the table overflow will be more serious, and how to deal with it is still a problem to be solved. Zhou et al. proposed a strategy to build a new flow aggregation algorithm and a multilevel flow table architecture to defend against the overflow attack launched by the attacker [5]. Cao studied the effects of the LOFT (Low-Rate Flow Table) overflow attacks on the SDN network, proposed a method for attackers to detect the SDN network configuration and build low-rate attack traffic, and gave two simple methods to prevent the network configuration detection [13]. However, there is no effective scheme to detect and mitigate the attack flow. Xu et al. studied the potential targets of table overflow attack and used 3 metrics to detect the attack flows: GFFC (Growth of Foreign Flow Consumption), DFA (Deviation of Amount), and CFE (Commonness of Flow Entry) [6]. This work is mainly aimed at the defense of attacks that send a large number of flows in a short time. The corresponding mitigation mechanism is to limit the speed of installing flow entries from the controller by token bucket, so as to avoid exhausting the flow table space.

To sum up, there is no systematic research on the detection and defense of table overflow LDoS attack. The main reason is that LDoS attack has strong concealment, not easy to be discovered and defended. Nevertheless, the flow table overflow LDoS attack has an important effect on the performance of SDN. Therefore, this paper mainly studies the detection and defense mechanism of flow table overflow LDoS attack. Based on the previous conference manuscripts [37], this paper has carried out many extensions, including detailed introduction of background, further analysis of attack flow, optimization of flow table overflow mitigation module design, and large-scale experiments.

3. Analysis of Flow Table Overflow LDoS Attack

3.1. Controller Strategy Model. According to the OpenFlow protocol specification, each flow entry contains a timeouts component, which is used to specify the maximum amount of survival time and idle time of the flow entry in the switch before expiration. The controller needs to set the timeout values when installing the flow entry to the switch, which contains two values: *hard_timeout* and *idle_timeout*. *Idle_timeout* means that the flow entry will be deleted by the switch if it does not match the subsequent data packet within the period after the data packet is matched. *Hard_timeout* means that the switch will actively delete the flow entry from the time of installation to the end of the time, regardless of the match situation of the flow entry. When *hard_timeout* and *idle_timeout* are both set to 0, the flow entry will not be deleted; and when one of *hard_timeout* and *idle_timeout* is set to 0 and the other is set to a positive number, the lifetime of the flow entry depends on nonzero timeout item; if both timeout values are set to positive numbers, the survival time of the flow entry depends on whichever expires first.

In order to avoid increasing network delay due to the deletion of flow entries and ensure that switches can actively delete outdated flow entries, the controller generally adopts the flow entry timeout strategy, as shown in the following equation, where t is a nonzero positive number. Table 1 lists the configuration of default timeout values for flow entries in typical controllers:

$$\begin{cases} hard\_timeout = 0, \\ idle\_timeout = t. \end{cases} \tag{1}$$

According to this policy, even if no packet is matched within time t, the flow entries on the switch will be retained in the flow table space after installation.

3.2. LDoS Attack Model against Table Overflow. According to the flow table space LDoS attack model proposed by [13], after getting the switches' flow table capacity by [12], an attacker can send a large number of flows to the network with packet intervals less than the timeout of flow entries. These attack flows are clustered on the target switch in the network, which makes the flow table space of the target switch always be in saturated state, thus causing the influence of denial of service to the new normal flow. In addition, according to different attack traffic distribution patterns, an LDoS attacker can use the following two types of attack flows to occupy the flow table space.

Type-I attack flow uses a long flow to occupy the switch's one flow entry space. In this case, the attacker sends a small packet to the network and then resends the flow to the network before the corresponding flow entry timed out, so that the flow entry on the switch would remain active due to the continuous matching. As shown in Figure 4(a), the vertical lines of the same color represent each packet of an attack flow and the time interval between two packets is slightly less than the timeout.

Type-II attack flow uses multiple small flows to occupy the switch's one flow entry space. In this case, the attacker uses a small flow similar to the Type-I attack flow to occupy the flow table space of the switch. However, the difference is that the duration of each flow is very short, and the newly generated flow is constantly replaced by the new flow. In other words, when the new attack flow arrives, the space left by the deleted flow entries due to the mismatch of flows is occupied by attack flow again. Compared with Type-I attack flow, Type-II attack flow is more difficult to detect due to its changing flow characteristics. As shown in Figure 4(b), the vertical lines of the different color represent each packet of

Table 1: Timeout configuration in the typical controller.

Controller	Default timeout		Sample projects timeout		
	Hard (s)	Idle (s)	Project name	Hard (s)	Idle (s)
POX	0	0	l3_learning	0	10
RYU	0	0	rest_router	0	1800
ONOS	0	10	Fwd	0	10
OpenDaylight	600	300	L2switch	3600	1800
Floodlight	0	5	learning_switch	0	5

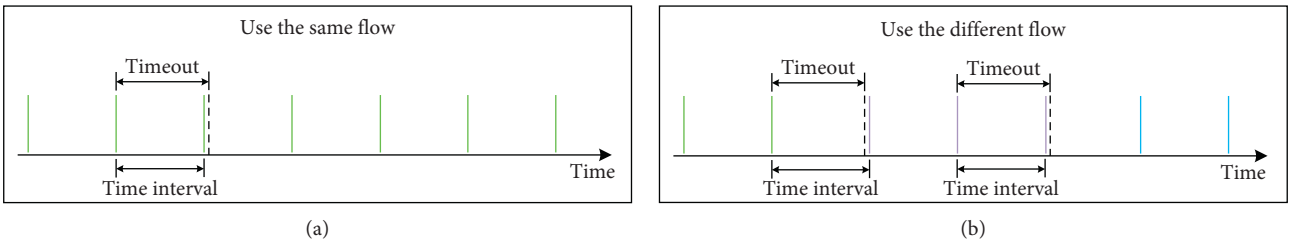

Figure 4: Different attack flow patterns. (a) Type-I attack flow and (b) Type-II attack flow.

different attack flows and the time interval between the two attack flows is slightly larger than the timeout.

3.3. Difficulty Analysis of the Detection and Defense Mechanism Design. The table overflow LDoS attacks, compared with traditional attack methods, have the following three detection and defense difficulties.

3.3.1. The Low-Rate Characteristics of the Attack Traffic Makes It Difficult to Detect. Different from traditional flooding DDoS attacks, LDoS attack flows have the characteristics of less traffic and low rate, which can be well hidden in the normal network traffic, so the attack flow is difficult to detect. As shown in Figure 5, we analyzed the network traffic in the univ1 dataset with 17 million packets [38] and the results show that more than 90% of the flows lasted less than 1 second. These flows accounted for 30% of the data packets, while the average data packet size was 124 bytes (including the packet header). In addition, as shown in Figure 6, the number of flows with only one packet accounts for 60% of the total number of flows. Therefore, an attacker can take advantage of this characteristic of network traffic and through careful design and send a low-rate attack flow to the network to simulate the small traffic that exists in the real network, thereby causing the flow table overflow. At the same time, since the packet size of each flow is small, these attack flows are difficult to detect by network defenders.

3.3.2. The Information Available for Analysis Is Limited. Without changing the architecture of the SDN switch and adding additional functions, the controller can only collect very limited information. The network traffic is forwarded through the matching of the flow table, and the statistical fields in the flow entries only include the number of packets, the number of bytes, and the duration of the flow entry. It is difficult to extract useful information from this data to detect the attack flow.

3.3.3. In a Normal Network State, Mitigation of Table Overflow Needs to Be Considered Comprehensively. Even in a normal network state, there is still a problem of table overflow caused by burst traffic. Therefore, the flow table overflow defense mechanism needs to be able to alleviate the flow table overflow under normal network conditions.

For problems (1) and (2), an attacker needs to aggregate the low-rate attack flow on the target switch when conducting LDoS attack, so it can consider analyzing the flow information on the target switch to find the attack flow. It can be seen from Figure 5 that, in the real network, normal flows have an obvious heavy-tailed distribution. As discussed in Section 3.2, if an attacker uses a Type-I attack flow, the switch will keep the flow entry for a long time. But the number of packets and bytes matching the flow entries will be very small, which will be very different from the normal flow of the network. If an attacker uses Type-II attack flows, its characteristics are similar to the large number of small flows that exist in the network. To detect such attack flows, we propose a method based on inport-flow statistics. This method uses the entire network information maintained in the controller to find out the source of the low-rate flow to the switch suspected of being attacked and then analyzes the low-rate flow by establishing a <switch, inport> pair. Based on this, the change trend of the number of flows over time can be constructed to distinguish between attack flows and normal flows. The information required for the entire detection process that can be seen from the above analysis can be obtained from the existing SDN architecture without any modification to the architecture or OpenFlow protocol.

For problem (3), we can effectively combine the flow table overflow detection and defense scheme under LDoS attack with the flow table overflow mitigation scheme under

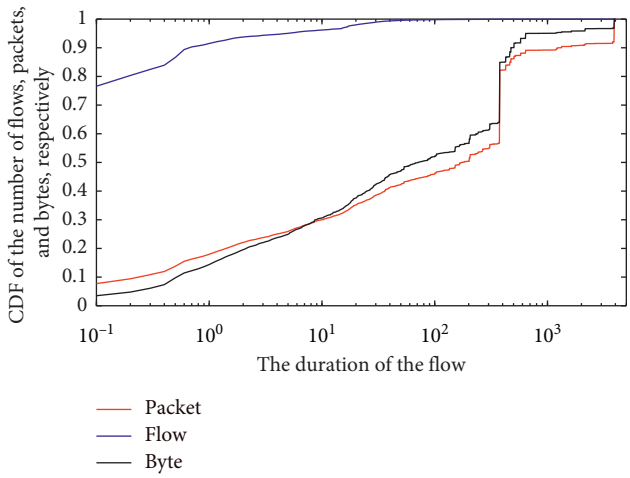

Figure 5: The CDF of the number of flows, packets, and bytes correspond to the duration of flow.

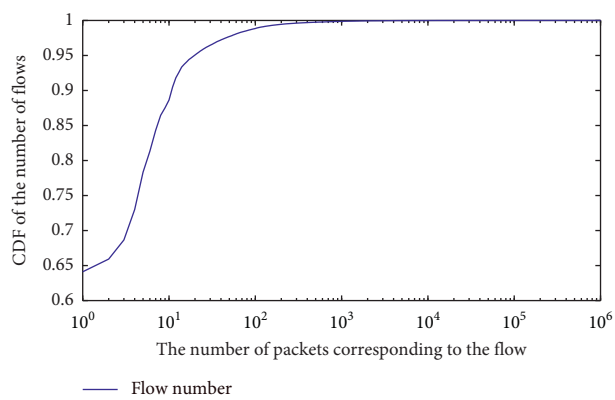

Figure 6: The CDF of the number of flows correspond to the packets number of flows.

normal network to better solve the flow table overflow problem under normal network conditions.

4. Detection and Defense Mechanism Design

The corresponding detection and defense mechanism named SAIA were proposed in this paper to effectively detect and defend the table overflow LDoS attacks. As shown in Figure 7, SAIA is composed of four main modules: data collection module (DCM), overflow prediction module (OPM), attack detection module (ADM), and overflow mitigation module (OMM). In the implementation, these modules are all running as applications of the controller, so they can be dynamically deployed.

In order to accurately describe the working mechanism of SAIA later, the symbol definitions involved are given in Table 2.

4.1. Data Collection Module. The DCM is mainly responsible for obtaining the data required for LDoS detection. There are mainly two ways to collect data: (1) active acquisition, that is, to acquire related data by actively sending a request to the SDN switch, such as sending *flow-stats-request* to request the counterfield of flow entries, and (2) passive recording, that is, recording the switch relevant messages that the switch reports to the controller, such as the number of new arrival flows corresponding to the *packet_in* events. Table 3 lists all the messages between the controller and the switches used by the DCM and the corresponding data that can be analyzed.

In order to determine whether the flow in the network is a large flow or a small flow, we need to know the packet rate (packets/s) and byte rate (bytes/s) of the flow arriving at the switch. In addition, in order to distinguish between attack flows and normal flows, we need to further construct the trend of the number of flows over time. Therefore, we design a data collection algorithm based on periodic sampling; see Algorithm 1 for details. DCM will record the data of each statistical metric in each sampling period in a period of time (20 sampling periods were used in our experiment). After each sampling, the collected data are sent to the OPM for further analysis in the form of events, while $S_i\_inport_k$, $A_{i,jk}(t)$, $P_{i,jk}(t)$, and other data are sent to the ADM to detect whether there is an inport sending Type-II attack flow.

4.2. Overflow Prediction Module. The main function of OPM is to predict the time when the table overflow occurs to help OMM to selectively delete a certain number of flow entries before the overflow event occurs, thereby effectively avoiding the occurrence of flow table overflow.

In order to accurately predict the overflow event of the flow table in the network, based on the analysis of the information of switch flow table, a prediction algorithm for the flow table overflow is proposed. This algorithm combines the current usage of the switch's flow table space and predicts the number of new flows arriving at the switch in the next sampling period to determine whether the flow table will overflow. See Algorithm 2 for specific algorithm description.

As shown in lines 3–7 of Algorithm 2, the prediction of the number of new flow arrivals is mainly based on the changing trend (slope) of new flow arrivals. When the slope k_1 between two points of $pi(t_n)$ and $pi(t_{n-1})$ is greater than the slope k_2 between two points of $pi(t_{n-1})$ and $pi(t_{n-2})$, the predicted slope k_0 between two points of $pi(t_{n+1})$ and $pi(t_n)$ is equal to k_1 plus the logarithm of the increase of the slope; otherwise k_0 is equal to k_1. This algorithm not only effectively predicts the number of new flows arriving at the switch but also enlarges the predicted value according to the change of the number of new flows arriving. Figure 8 is a schematic diagram of new flow arrival quantity prediction. The black point in the figure is the sampling value of the sampling point, and the red point is the predicted value of the sampling point. It can be seen from the area 1 in the figure that when the number of new flows increases suddenly, the algorithm also has good prediction results. At the same time, combined with the flow table overflow mitigation mechanism, when the usage rate of the flow table space is higher, it is necessary to appropriately enlarge the predicted value of the number of new flows, as shown in the areas 2 and 3 in the

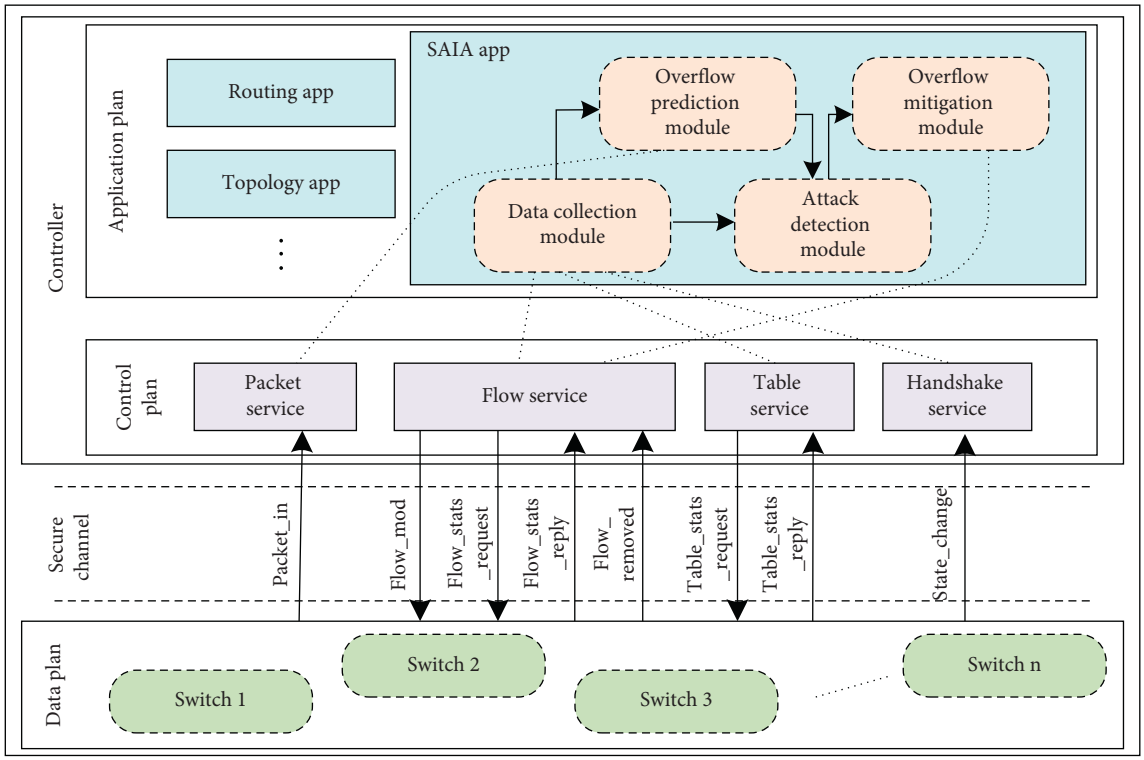

Figure 7: Table overflow attack detection and defense mechanism logic architecture.

Table 2: Symbol definitions.

Symbol	Definition
S_i	SDN switch i
C_i	The flow table capacity of the switch i
T	Sampling period
$A_i(t)$	The number of active flow entries in the switch i at t th sampling
$p_i(t)$	The number of new flows arriving at the switch i during t th sampling time
$F_{ij}(t)$	Flow entry j in the switch i at t th sampling
$P_{ij}(t)$	The number of matched packets for the entry j in the switch i at t th sampling
$B_{ij}(t)$	The number of matched bytes of entry j in the switch i at t th sampling
$D_{ij}(t)$	Duration time of entry j of in the switch i at t th sampling
$S_i\_inport_k$	Network access port k of the switch i (switch is edge switch)
$A_{i,jk}(t)$	The number of flow entries installed on switch i corresponding to the flow from the inport k of switch j
$P_{i,jk}(t)$	The number of new flows from the inport k of switch j to the switch i during t th sampling time

Table 3: Message type description.

Message type	Message	Function	Analyzed data
Controller-to-Switch	*flow-mod*	Install the flow entries to the switch	—
	flow-stats-request	Request the switch for flow entry statistics field information	—
	table-stats-request	Request the switch for flow table space usage	—
Asynchronous	*state-change*	Send switch connection and disconnect events to the controller	Update the number of SDN switches
	packet-in	Forward packets to the controller	The arrival rate of new flow
	flow-stats-reply	Reply to flow entry statistics field information to the controller	Flow entries matched packet, bytes, and duration
	table-stats-reply	Reply to flow table usage to the controller	Number of active flow entries
	flow-removed	Reports to the controller that the switch deletes the flow entry	Delete the corresponding flow entry data in the record

Note: OpenFlow specification v.1.3 was referenced for SAIA design.

```
Topo = Controller.topo
C_i = Topo(i).capacity
while (time elapse T)
    for S_i in Topo
        send table_stats_request message to S_i
        A_i = length(table_stats_reply.active_count)
        send flow_stats_request message to S_i
        for F_ij(t) in flow_stats_reply
            D_ij(t) = F_ij(t).duration
            P_ij(t) = F_ij(t).packet
            B_ij(t) = F_ij(t).byte
            if timeSpan = = ilde_timeout/2 and F_ij(t) is A_i,jk(t)
                A_i,ik(t) = A_i,jk(t) + 1
    p_i(t), P_i,jk(t)//count the number of packet_in in this period
```

Algorithm 1: Data collection.

```
input: pi, Ai, T
output: pre(t_{n+1})//predicted value of Ai at the next sampling time t_{n+1}
k_1 = (pi(t_n)−pi(t_{n−1}))/T
k_2 = (pi(t_{n−1})−pi(t_{n−2}))/T
if k_1 > k_2
  k_0 = k_1 + log_10(k_1−k_2)
else
  k_0 = k_1
end
pi(t_{n+1}) = k_0 * T + pi(t_n)
pre(t_{n+1}) = Ai(t_n) + pi(t_{n+1})
```

Algorithm 2: Table overflow prediction.

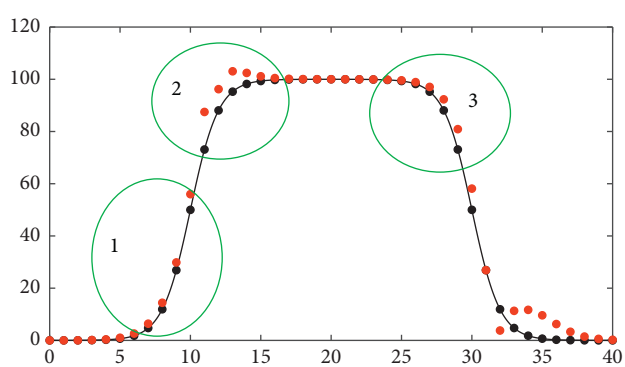

Figure 8: New flow arrival quantity prediction algorithm.

figure. This can effectively avoid the flow table overflow caused by the inability of accurately predicting a large quantity of new flows.

When the predicted number $pre(t_{n+1})$ of flow entries in switch i is greater than the flow table space size C_i of the switch, some flow entries need to be deleted in advance to make room for the new arrival flows. Therefore, it is necessary to send the collected data of switch i to the ADM for detection and find out the suspected attack flow for deletion.

4.3. Attack Detection Module. The main work of ADM is divided into two parts: (1) when the OPM predicts that there will be a table overflow in switch *Si*, ADM will analyze the flows through switch *Si* based on the collected data information to detect the Type-I attack flow; (2) it periodically analyzes the data such as $S_i\_inport_k$, $A_{i,jk}(t)$, and $P_{i,jk}(t)$ collected by data collection module as every half *idle_timeout*, so as to detect the network inport of Type-II attack flow.

For Type-I attack flow, in order to effectively distinguish it from the normal flow in the network, the ADM uses factors such as average packet interval, average packet size, and flow duration to evaluate the probability that the flow is an attack flow. The specific evaluation strategy is shown in the following equation, where the purpose of introducing parameter α is to optimize the impact of average packet interval on the evaluation results, whose value can be appropriately adjusted for different network scenarios. The constant *avgb* is the average packet size (number of bytes) in the network:

$$f^1_{\text{attack}}(t) = \alpha \cdot \frac{D_{ij}(t) - idleTime}{P_{ij}(t)} + (1-\alpha) \cdot \frac{P_{ij}(t)}{B_{ij}(t)} \cdot avgb. \tag{2}$$

Since all packets for each flow are counted, for a normal flow, even with a heavy-tailed distribution, the average time interval of all packets is significantly smaller than the attack flow, and the average size of the packets is significantly larger than the attack flow. Therefore, the attack flow can be discriminated by defining a threshold $sh1$ according to different network scenarios. When $f^1_{\text{attack}} < sh1$, the flow is considered as an attack flow; when $f^1_{\text{attack}} >= sh1$, the flow is considered as a normal flow. When the module detects the Type-I attack flow, the attack flow information is sent to the OMM for precisely deleting the corresponding flow entries of the attack flow.

For Type-II attack flow, since each attack flow has a short time, it is difficult to evaluate them effectively through the above equation. Thus, we designed the identification pair < switch S, inport P>, which depicts the case that the traffic of a switch S comes from network access port P. As shown in Figure 9, it is necessary to establish the identical pairs between 8 network access ports and 5 network switches, such as $< S_1$ and S_2_inport$_1$>. The miscalculation caused by actions such as network source address spoofing can be effectively overcame in this method. According to the analysis in 3.3.1, a large number of small flows with only a single packet exist in the network. In order to effectively distinguish them from Type-II attack flow, we use the following equation to evaluate the flows with switch i and network access port as the port k of switch j. Because Type-II attack flow sends new flows continuously, the switch generates new flow entries to replace the old ones that are about to expire, which makes the difference value of $A_{i,jk}$ between the two consecutive counts small while $P_{i,jk}$ is larger. Because the statistical period is half of *idle_timeout*, for normal network access ports, even if there are a great deal of small flows of single data group, the difference value of $A_{i,jk}$ between two consecutive statistics is very large:

$$f^2_{\text{attack}}(t) = \frac{\sum_{n=0}^{3} P_{i,jk}(t_n)}{1 + \sum_{n=0}^{4}\left(\left|A_{i,jk}(t_{n+1}) - A_{i,jk}(t_n)\right|\right)}. \quad (3)$$

Similarly, $sh2$ is defined according to different network scenarios. When $f^2_{\text{attack}} > sh2$, $S_j\_inport_k$ is considered to be the attack node of Type-II attack flow; when $f^2_{\text{attack}} <= sh2$, the inport is considered to be a normal network entry port. When the network inport of Type-II attack flow is detected, the <switch, inport> identification pair is sent to the OMM immediately, so that the OMM can delete the flow entries of flows that come from the specific network inport when the switch flow table overflows.

4.4. Overflow Mitigation Module. The OMM has two main functions: (1) maintaining the network access port of Type-II attack flows detected by the ADM and (2) selecting the appropriate flow entries to delete.

For the attack network access port maintenance function, when detected for the first time, the network access port is recorded in the form of <switch, inport> pair and deleted after a certain time (the experimental setting is 5 times *idle_timeout*). The purpose of the deletion here is to remove the attack port mark timely and effectively after the attacked port becomes normal.

For the function of selecting the appropriate flow entries to delete, in order to avoid network performance degradation caused by flow table overflow, SAIA actively deletes a certain number of n ($n = pre(t_{n+1}) - 0.9 * C_i$) flow entries by selecting the most suitable flow entries when the flow table is about to overflow. The logic for deleting the flow entries is shown in Figure 10. Here, combining the ADM to detect the results of the two types of attack flows, it is divided into three steps to select the flow entries to be deleted.

Firstly, when the ADM detects a Type-I attack flow, it directly deletes n flow entries according to the descending order of the f^1_{attack} values (see Figure 10(b)).

Secondly, if no Type-I attack flow is detected, or if the number of detected attack flows is less than n, check the <switch, inport> table maintained by the mitigation module. If there is an attack port, select the remaining number of flow entries that need to be deleted from the flow entries with the inport constraint (see Figure 10(c)).

Otherwise, select the remaining flow entries to delete from all flow entries in the switch that will overflow the flow table (see Figure 10(d)).

In the last two steps, the Least Recently Used (LRU) algorithm is used to select the most suitable flow entry to be deleted. This is because the flow entry with the largest LRU value is generally the flow entry that is about to expire, so deleting it has little impact on the network, and LRU is easy to implement without changing the switch [39]. Therefore, we designed the following equation to calculate the LRU value of the flow:

$$f_{LRU} = \frac{\beta \cdot idleTime}{(1-\beta)\cdot\left(P_{ij}(t_n) - P_{ij}(t_0)\right)}. \quad (4)$$

IdleTime is the time interval between the last match of the flow entry and the current moment, and $P_{ij}(t_n) - P_{ij}(t_0)$ is the number of packets that the flow entry matched in the last n samples. From the equation, it can be seen that the higher the value of f_{LRU}, the smaller the number of packets that the flow entry matched, and the flow entry is not matched for a long time. To a certain extent, the corresponding entries of the flow with certain attack flow characteristics can be selected with a high probability.

5. Implementation and Evaluation

By building a simulation environment of RYU + Mininet, we verified the proposed attack detection and defense system through experiments. The simulation environment is deployed on two computers, one of which runs RYU controller (CPU i5-7300HQ 2.5 GHz and RAM 16G), and the other runs Mininet (CPU i5-3470 3.2 GHz and RAM 8G). The experimental network topology is shown in Figure 9. The controller processes *table_miss* by installing flow entries of four different matching fields of ARP, ICMP, TCP, and UDP, where the matching field of ARP is {eth_dst, eth_src, eth_type}, the matching field of ICMP is {eth_dst, eth_src,

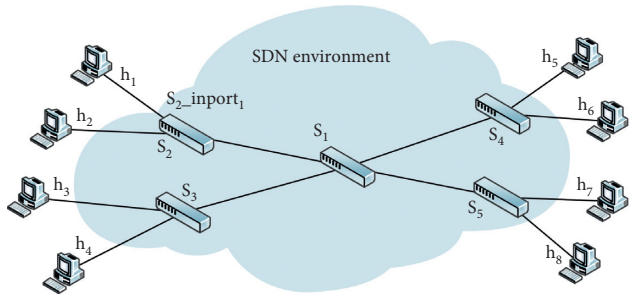

Figure 9: Network topology.

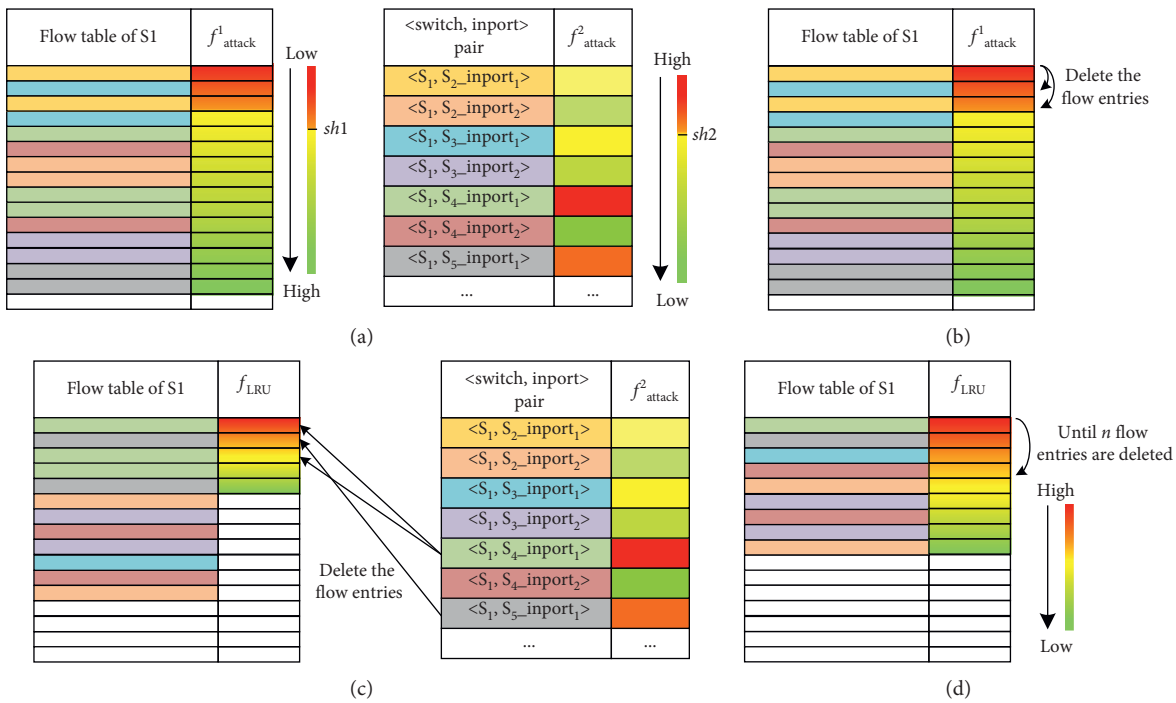

Figure 10: OMM steps to delete flow entries. (a) Data maintained by OMM when the flow table is about to overflow. (b) Delete the flow entries with lower f^1 attack value. (c) Delete the flow entries with higher f_{LRU} value within higher f^2_{attack} value pair. (d) Delete the flow entries with higher f_{LRU} value.

eth_type, ip_proto}, and the matching field of TCP (UDP) is {eth_dst, eth_src, eth_type, ip_src, ip_dst, ip_proto, TCP(UDP)_src, TCP(UDP)_dst}. Table 4 shows the basic parameters of the experiment in this section. Some different parameters will be further explained in the specific experiment.

5.1. Evaluation of the LDoS Attack Impact. We tested the impact of the flow table overflow attack on the network performance without SAIA protection in the controller to verify the harmfulness of the flow table overflow. That is, when the switch flow table space is full, the new flow arriving at the switch is forwarded to the corresponding port of the switch through the controller without installing the corresponding flow entry. Figure 11 shows the comparison between the attack flow and the network transmission delay and throughput under normal conditions, respectively.

It can be seen from Figure 11(a) that when no flow table overflow occurs (black line), for a normal flow, except for the first packet, the latency of subsequent packets is almost 0. This is because the first packet of the new flow arriving at the switch does not have a corresponding flow entry for matching, and the controller needs to process the flow and then install the corresponding flow entry to the switch. When Type-I attack flow causes the table overflow (red line), the delay of all data packets of the new flow is about 50 ms. This is because the controller cannot install the corresponding flow entry of the new flow for the switch, resulting in each packet of the flow being regarded as a new flow. However, when Type-II attack flow causes the table overflow (blue line), the first few packets of the new flow have a large delay. This is because the Type-II attack flow may have a few empty flow table spaces for a short time during the conversion process, which results in the installation of flow entries of normal packets. As the density of the Type-II attack flow increases, the number of initial packets with high

Table 4: Simulation parameters.

	Controller routing strategy				SAIA system related parameters				
Parameter	Routing algorithm	Match fields	*hard-timeout*	*idle-timeout*	Type-I attack *sh1*	Type-II attack *sh2*	α	*Avgb*	β
Value	Dijkstra	8 tuples	0 s	20 s	10	2	0.8	128	0.6

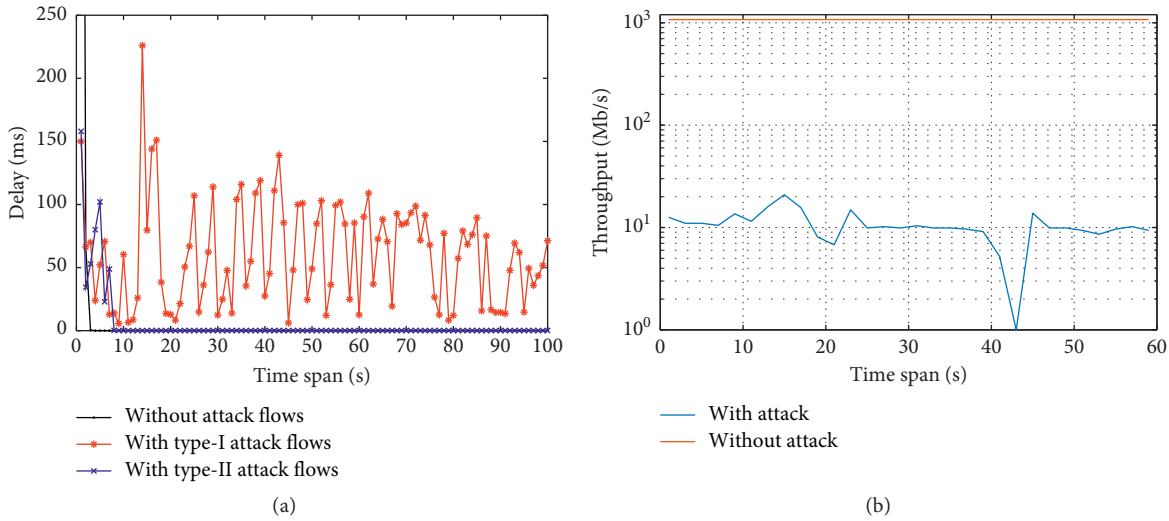

Figure 11: The impact of attacks on network. (a) The impact of attack on network delay. (b) The impact of attack on network throughput.

latency of a new flow will also increase. Accordingly, when the network is under attack, the network throughput also drops significantly. It can be seen from Figure 11(b) that the network throughput decreases from 1 Gb/s under normal conditions to 10 Mb/s after being attacked.

5.2. Evaluation of the Flow Table Usage Prediction. The effectiveness of the prediction algorithm is evaluated through experiments which use the topology environment shown in Figure 9 and generate traffic by 8 hosts. The controller only uses SAIA's DCM and OPM to collect relevant data and predict the changes of the flow table space of switch S1. The experimental results are shown in Figure 12.

It can be seen from Figure 12 that the prediction algorithm can make relatively accurate predictions on the usage of the flow table space, and, in most cases, the predicted value is slightly higher than the actual value, which is helpful to avoid table overflow caused by burst flood flows. Further, when the number of flow entries is at a high level, the relative error between the predicted value and the actual value is less than 10%, and almost all of them are slightly higher than the actual value, indicating that the flow table prediction algorithm can be well applied to the detection and defense system and can effectively deal with the burst flood flows.

5.3. Evaluation of the Detection and Defense System. To further verify the effectiveness of the detection and defense system, we first performed an experimental evaluation on the active deletion function of the flow entries according to Section 4.2. In the experiment, the controller sets the size of the flow table space of switch S_1 to 500 flow entries, while the switch does not limit the size. The result is shown in Figure 13, in which the red line is the change curve of the number of flow entries in S_1 over time. As can be seen from the figure, the number of real-time flow entries in the switch has always been below 500. Even when the network load is large, the number of flow entries is only close to 500 but never exceeds. It can be concluded that SAIA's active flow table deletion function can effectively delete a certain number of flow entries when the flow table is about to overflow, thereby avoiding the occurrence of table overflow.

In addition, we analyzed the detection and defense capabilities of the system against flow table overflow LDoS attacks through another four sets of experiments. In the topology environment of Figure 9, the hosts h2, h4, h6, and h8 are defined as normal users, and they generate network background flows with different distributions to each other through the D-ITG tool. Hosts h1, h3, h5, and h7 are controlled by the attacker and use hping3 tool to generate attack flows, among which h1 and h3 generate Type-I attack flows and h5 and h7 generate Type-II attack flows. Four groups of experiments were conducted for the ratios of different flows. The ratios of benign user's flows, Type-I attack flows, and Type-II attack flows were [1, 0, 0], [0.5, 0.5, 0], [0.5, 0, 0.5], and [0.4, 0.3, 0.3], respectively. When the flow table overflow is predicted, the deleted flow entries in each group are shown in Figure 14, where benign, attack1, and attack2, respectively, represent the flow entries corresponding to normal user's flow, Type-I attack flows, and Type-II attack flows. x-axis represents the order in which flow table deletion events occur, and y-axis represents the number of deleted flow entries.

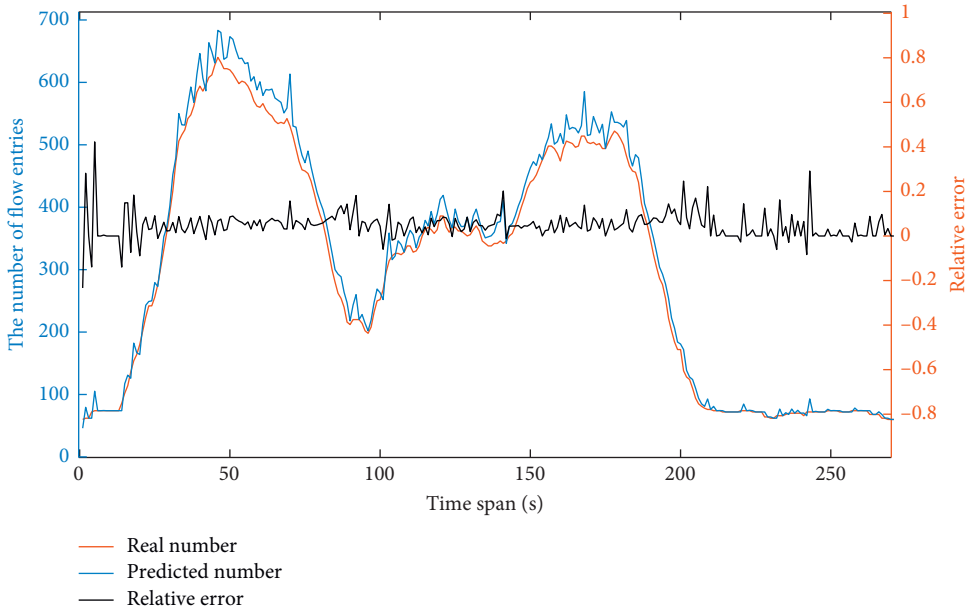

Figure 12: Prediction effect.

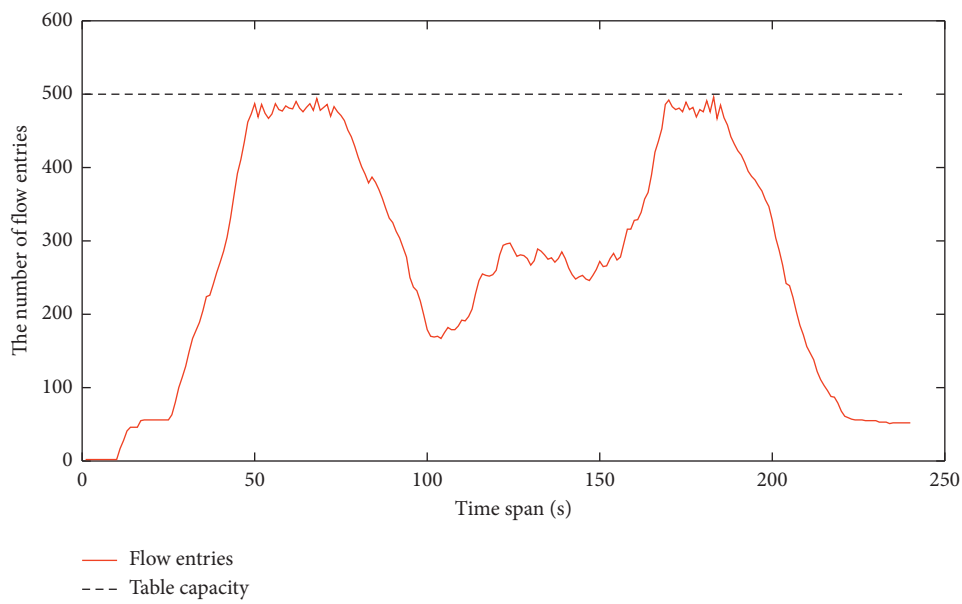

Figure 13: Active deletion of flow entries.

It can be seen from Figures 14(a) and 14(b) that when there are only normal flows in the network, the flow entries that SAIA should delete in response to the flow table overflow events are all flow entries corresponding to the normal flow; and when the normal flows and the Type-I attack flows account for 50%, respectively, in the network, the flow entries deleted by SAIA only contain the flow entries corresponding to the attack flows. This shows that the impact of Type-I attack flow on the network can be identified and suppressed effectively by the SAIA system. In Figure 14(c), when the normal flows and Type-II attack flows account for 50%, respectively, in network, the deleted entries of real attack flows account for 70%. The reason for certain misjudgment is that SAIA determines the Type-II attack flow based on the traffic information of the network port; that is, it identifies the network port, not the specific attack flow. The 70% recognition rate also means that most attack flows can be detected and suppressed. For the Type-I attack flow during the 1st and 6th deletion process, through further analysis of the feedback information of the switch, it is found that the one-time ARP data flow was mistakenly classified as attack flows. Since ARP only needs to interact once in the end-to-end communication of the actual network, deleting the corresponding flow entry will not only not affect the

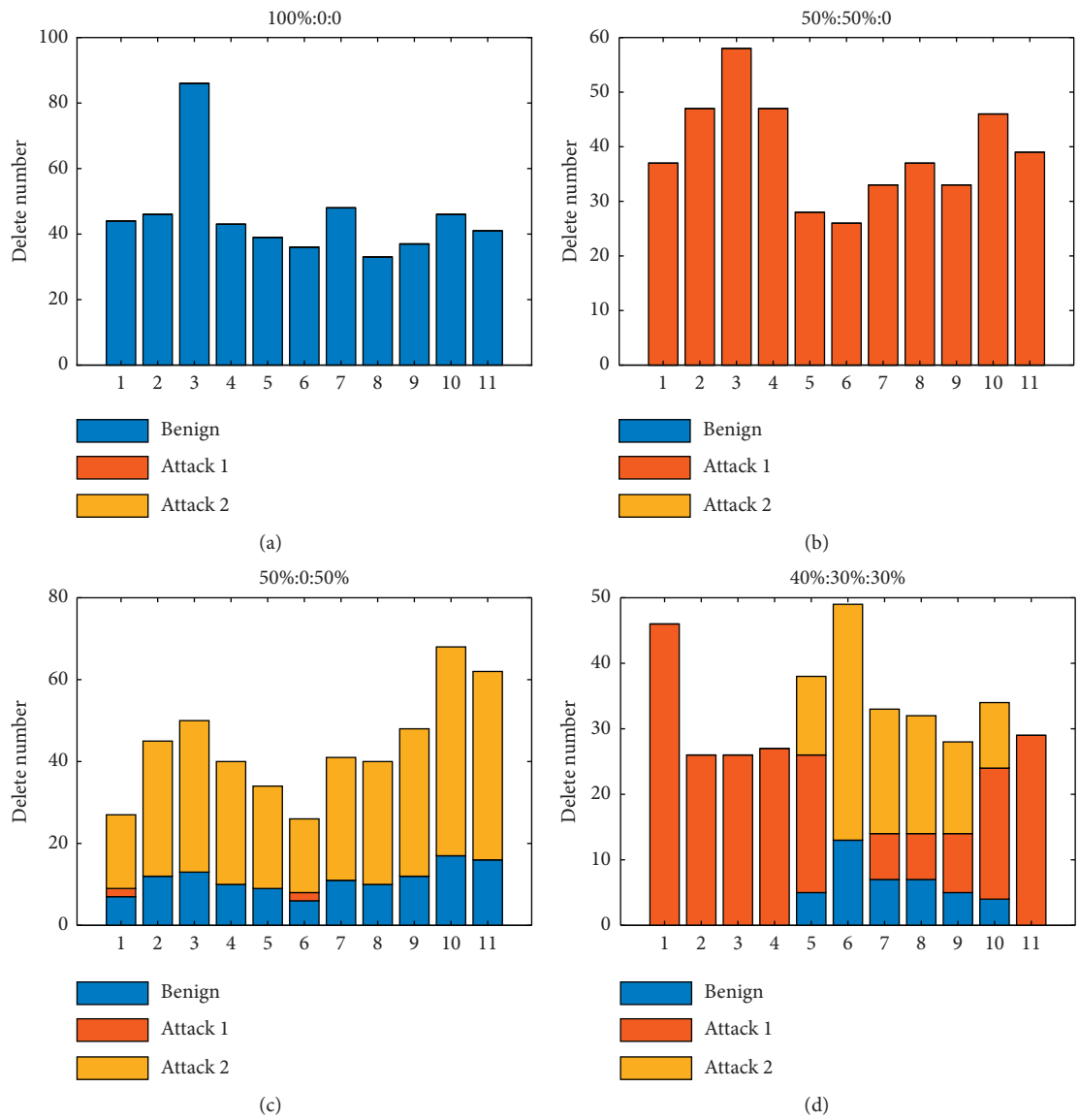

Figure 14: Flow entries deletion.

normal network communication, but also reduce the utilization rate of the flow table space. In Figure 14(d), when the three types of flows account for 40%, 30%, and 30%, respectively, many entries of Type-I attack flows with most obvious attack characteristics are deleted first, and then entries of Type-II attack flows are deleted. In summary, it can be found that the table overflow LDoS attacks can be detected and defended effectively by SAIA proposed in this paper.

In order to evaluate the resource utilization of the SAIA detection and defense system at different network scales, we launched two sets of experiments: the CPU resources occupied by the SAIA detection and defense system at different network scales and the extra total bandwidth of the secure channels occupied by the systems at different network scales. As can be seen from Figure 15, the CPU resources occupied by SAIA attack detection and defense system showed a linear increase (about 8.5%/100 switches), and the additional bandwidth of total security channel occupied by SAIA also increases linearly (about 0.7 Mbps/100 switches). This shows that SAIA is a lightweight detection and defense system.

Finally, the detection performance of the SAIA detection and defense system under different network scales was experimentally verified. At each network size, 50 experiments were done. As shown in Figure 16, the SAIA detection and defense system can have excellent detection effects under different network scales. Even when the network scale is 300, the detection success rate of two attack types can still reach 90%, and this part of the attack flow is preferentially deleted when the flow table is about to overflow.

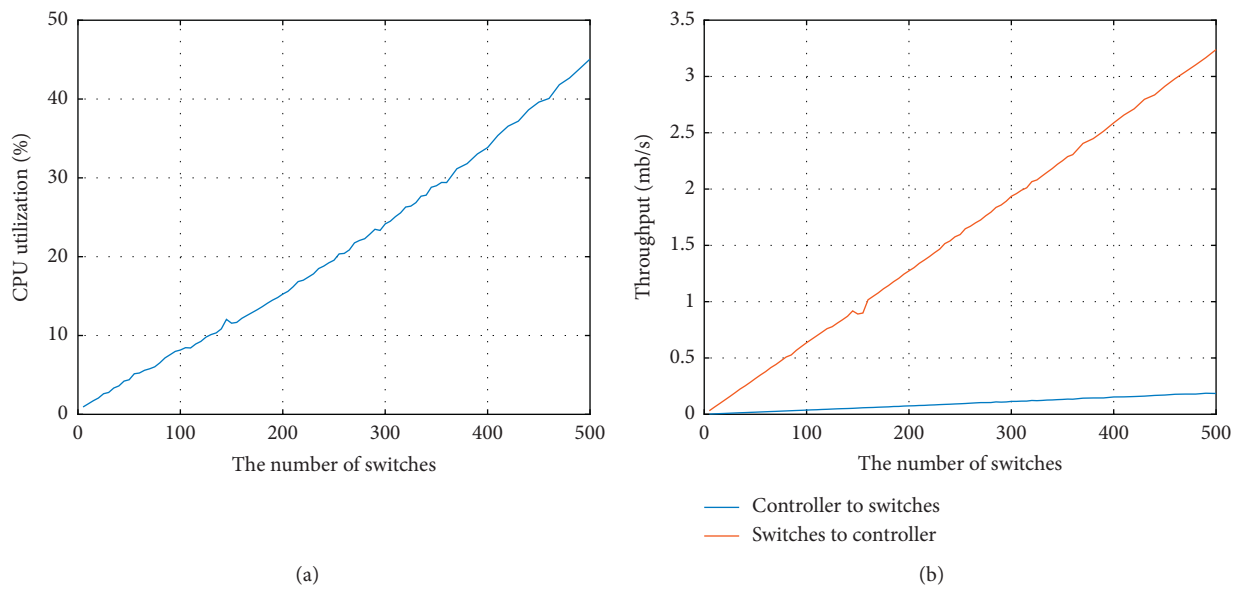

Figure 15: The resource consumption of SAIA system under different network sizes. (a) CPU utilization. (b) Throughput.

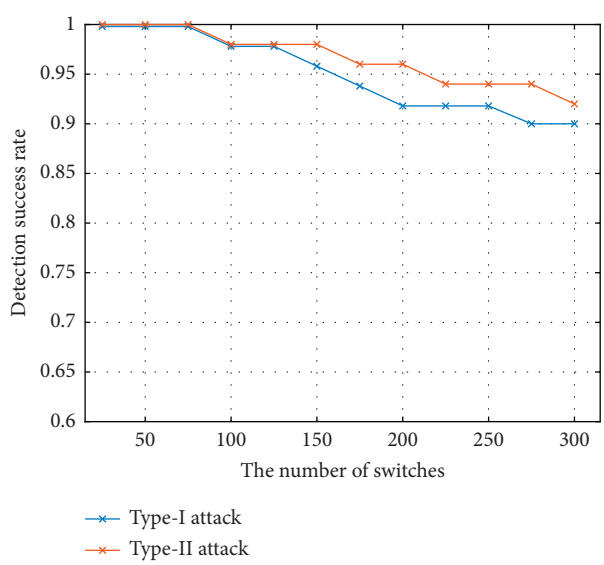

Figure 16: The detection effect of the system under different network scales.

6. Conclusion

The new architecture of SDN not only enhances the network control capability but also brings some new security threats. Table overflow LDoS attack is a destructive attack. In this paper, two typical flow models of table overflow LDoS attack were studied, and corresponding SAIA detection and defense mechanism were proposed. In SAIA, the data collection module can effectively collect relevant data in real time through active and passive methods, such as flow table, flow entries, and the number of matched bytes of a flow entry. The table overflow prediction module can predict table overflow with 10% relative error and can adapt to sudden traffic in network. The attack detection module and the attack mitigation module can detect the attack flow and effectively delete the corresponding flow entries to provide the flow table space for the normal new flow. Experiments show that SAIA can be deployed in a light-weight manner on a large scale and can have better attack detection and defense effects. At the same time, in the absence of attack flow, when the flow table overflows, the corresponding flow entries can be deleted according to the LRU algorithm to alleviate the network performance degradation caused by table overflow.

In the future work, improvements can be made from the following two points: first, combine the controller's whole network view with the number of new flows of each edge switch and use intelligent algorithms such as machine learning to predict the flow table overflow of the core switch more accurately; second, machine learning method can be considered to improve the detection accuracy of attack flow.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

An earlier version of this paper was presented at the International Conference on Big Data and Security.

Conflicts of Interest

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

Acknowledgments

This research was supported in part by the Natural Science Foundation of China (nos. 61379149 and 61772271) and the

China Postdoctoral Science Foundation under Grant no. 2017M610286.

References

[1] C. Yoon, S. Lee, H. Kang et al., "Flow wars: systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514–3530, 2017.

[2] 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, pp. 1701–1725, 2017.

[3] Y. J. Tzang, H. Y. Chang, and C. H. Tzang, "Enhancing the performance and security against media-access-control table overflow vulnerability attacks," *Security and Communication Networks*, vol. 8, no. 9, pp. 1780–1793, 2015.

[4] K. Kalkan, G. Gur, and F. Alagoz, "Defense mechanisms against DDoS attacks in SDN environment," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 175–179, 2017.

[5] Y. Zhou, K. Chen, J. Zhang, J. Leng, and Y. Tang, "Exploiting the vulnerability of flow table overflow in software-defined network: attack model, evaluation, and defense," *Security and Communication Networks*, vol. 2018, Article ID 9804061, 15 pages, 2018.

[6] T. Xu, D. Gao, P. Dong, C. H. Foh, and H. Zhang, "Mitigating the table-overflow attack in software-defined networking," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1086–1097, 2017.

[7] D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[8] B. Isyaku, M. B. Kamat, K. bin Abu Bakar, M. S. M. Zahid, and F. A. Ghaleb, "IHTA: dynamic idle-hard timeout allocation algorithm based openflow switch," in *Proceedings of the 2020 IEEE 10th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pp. 170–175, Malaysia, April 2020.

[9] 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*, vol. 12, no. 2, pp. 231–246, 2019.

[10] M. Zhang, J. Bi, J. Bai et al., "A priority-aware strategy against the flow table overflow attack in SDN," in *Proceedings of the SIGCOMM Posters and Demos*, pp. 141–143, New York, NY, USA, 2017.

[11] R. Rabie and M. Drissi, "Applying sigmoid filter for detecting the low-rate denial of service attacks," in *Proceedings of the 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 450–456, Las Vegas, NV, USA, January 2018.

[12] J. Leng, Y. Zhou, J. Zhang, and C. Hu, "An inference attack model for flow table capacity and usage: exploiting the vulnerability of flow table overflow in software-defined network," 2015, https://arxiv.org/abs/1504.03095.

[13] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, and J. Zheng, "Disrupting SDN via the data plane: a low-rate flow table overflow attack," *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 238, pp. 356–376, 2018.

[14] NOF, Openflow specificaitons, 2020, https://www.opennetworking.org/software-defined-standards/specifications/.

[15] L. Liu, H. Wang, Z. Wu, and M. Yue, "The detection method of low-rate DoS attack based on multi-feature fusion," *Digital Communications and Networks*, vol. 6, no. 4, pp. 504–513, 2020.

[16] Z. Wu, Q. Pan, M. Yue, and L. Liu, "Sequence alignment detection of TCP-targeted synchronous low-rate DoS attacks," *Computer Networks*, vol. 152, no. 7, pp. 64–77, 2019.

[17] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants," in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols For Computer Communications—SIGCOMM '03*, pp. 75–86, Karlsruhe Germany, August 2003.

[18] J. B. R. Sharma, "Detectability of low-rate HTTP server DoS attacks using spectral analysis," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 954–961, Paris, France, August 2015.

[19] G. Maciá-Fernández, J. E. Díaz-Verdejo, and P. García-Teodoro, "Evaluation of a low-rate DoS attack against application servers," *Computers & Security*, vol. 27, no. 7-8, pp. 335–354, 2008.

[20] G. Maciá-Fernández, R. A. Rodríguez-Gómez, and J. E. Díaz-Verdejo, "Defense techniques for low-rate DoS attacks against application servers," *Computer Networks*, vol. 54, no. 15, pp. 2711–2727, 2010.

[21] K. Hong, Y. Kim, H. Choi, and J. Park, "SDN-assisted slow HTTP DDoS attack defense method," *IEEE Communications Letters*, vol. 22, no. 4, pp. 688–691, 2018.

[22] T. Lukaseder, L. Maile, B. Erb, and F. Kargl, "SDN-assisted network-based mitigation of slow DDoS attacks," in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, pp. 102–121, Orlando, FL, USA, October 2018.

[23] K. S. Sahoo, D. Puthal, M. Tiwary, J. J. P. C. Rodrigues, B. Sahoo, and R. Dash, "An early detection of low rate DDoS attack to SDN based data center networks using information distance metrics," *Future Generation Computer Systems*, vol. 89, pp. 685–697, 2018.

[24] R. Mohammadi, M. Conti, C. Lal, and S. C. Kulhari, "SYN-Guard: An effective counter for SYN flooding attack in software-defined networking," *International Journal of Communication Systems*, vol. 32, no. 17, 2019.

[25] Q. Li, N. Huang, D. Wang et al., "HQTimer: a hybrid ${Q}$ -Learning-Based timeout mechanism in software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 153–166, 2019.

[26] K. Kannan and S. Banerjee, "Flowmaster: early eviction of dead flow on sdn switches," in *Proceedings of the International Conference on Distributed Computing and Networking*, pp. 484–498, Coimbatore, India, January 2014.

[27] B. Sooden and M. R. Abbasi, "A dynamic hybrid timeout method to secure flow tables against DDoS attacks in SDN," in *Proceedings of the 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pp. 29–34, IEEE, Jalandhar, India, December 2018.

[28] S. Qiao, C. Hu, X. Guan, and J. Zhou, "Taming the flow table overflow in openflow switch," in *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 591-592, ACM, Florianópolis, Brazil, August 2016.

[29] X. N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Optimizing rules placement in OpenFlow networks: trading routing for better efficiency," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, pp. 127–132, ACM, Chicago, IL, USA, August 2014.

[30] Z. Guo, R. Liu, Y. Xu, A. Gushchin, A. Walid, and H. J. Chao, "STAR: preventing flow-table overflow in software-defined networks," *Computer Networks*, vol. 125, no. 9, pp. 15–25, 2017.

[31] C. Wang and H. Y. Youn, "Entry aggregation and early match using hidden markov model of flow table in SDN," *Sensors*, vol. 19, no. 10, Article ID 2341, 2019.

[32] B. Leng, L. Huang, X. Wang, H. Xu, and Y. Zhang, "A mechanism for reducing flow tables in software defined network," in *Proceedings of the 2015 IEEE International Conference on Communications (ICC)*, pp. 5302–5307, London, UK, June 2015.

[33] S. Luo and H. Yu, "Fast incremental flow table aggregation in SDN," in *Proceedings of the 2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, IEEE, Shanghai, China, August 2014.

[34] H. Yang and G. F. Riley, "Machine learning based flow entry eviction for OpenFlow switches," in *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–8, IEEE, Hangzhou, China, August 2018.

[35] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, "Rules placement problem in OpenFlow networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1273–1286, 2016.

[36] M. K. A. Khan, V. K. Sah, P. Mudgal, and S. Hegde, "Minimizing latency due to flow table overflow by early eviction of flow entries in SDN," in *Proceedings of the 2018 9th international conference on computing, communication and networking technologies (ICCCNT)*, pp. 1–4, IEEE, Bangalore, India, July 2018.

[37] S. Xie, C. Xing, G. Zhang, and J. Zhao, "Research on table overflow ldos attack detection and defense method in software defined networks," in *Proceedings of the International Conference on Big Data and Security*, pp. 80–97, Springer, Melbourn, Australia, August 2019.

[38] Data Set for IMC 2010 Data Center Measurement [EB/OL], 2019, http://pages.cs.wisc.edu/~tbenson/IMC10_Data.html.

[39] A. Zarek, Y. Ganjali, and D. Lie, *Openflow Timeouts Demystified*, University of Toronto, Toronto, Canada, 2012.