Computer Networks 193 (2021) 108075

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A dynamic and lightweight framework to secure source addresses in the SDN-based networks

Qizhao Zhou [a,*], Junqing Yu [a,b,*], Dong Li [b]

[a] School of Computer Science & Technology, Huazhong University of Science and Technology, People's Republic of China
[b] Center of Network and Computation, Huazhong University of Science and Technology, Wuhan, People's Republic of China

ARTICLE INFO

Keywords:
Software defined network
Source address validation
Network security

ABSTRACT

We consider the problem of source address validation implementation (SAVI) in a Software-Defined Network (SDN) environment. The integration of SAVI and SDN can further address the challenges in a typical architecture such as the complexity of SAVI's deployment and the acquisition of security data in the access layer. A key aspect of this campaign consists of filtering forged packets and verifying the authenticity of the source address. A common strategy is to create bindings between the IP address of a node and a property of the host's network attachment. However, problem still accompany with the deployment of SAVI, including the performance cost of the SDN controller caused by the redundant validation process, especially in the case of an overflow of the flow table and other anomalous conditions in the network. Our contribution in this paper is to design and implement a dynamic framework for lightweight SAVI based on SDN (D-SAVI), which is an enhancement of SDN setups to allow proper source address validation on downstream network ingress ports, without incurring a large performance overhead. Initially, we proposed a fine-grained dual-level structure to match flow entry flexibly to complete the dynamic deployment of SAVI. A priority-based validation mechanism was added for further efficiency optimization. We then designed a state partition and transition module to optimize network performance under anomalous conditions, especially the communication performance between the controllers and switches in the SDN-based networks with a global view. Consequently, under the premise of network security, D-SAVI could filter out, with better packet forwarding efficiency, packets that do not match existing binding relationships. The experimental results demonstrate that our implementation provides source address verification with less resource consumption than existing methods.

1. Introduction

The proliferation of large-scale network like cloud data centers and campus networks has driven the need to secure network data. Currently, attackers can use spoofed source addresses to prevent tracing, conceal their identities, and defeat source-based filtering when performing Distributed Denial of Service attacks (DDoS) or poisoning attacks [1,2] against a data center or campus network [3,4]. Conversely, these networks have enormous bandwidth demands due to the recent implementation of bandwidth-intensive applications including Hadoop, VMWare, and Xen [5]. The bandwidth-intensive applications usually transfer huge quantities of data among thousands of servers, which complicates the verification of a spoofed address. Therefore, it is critical to confirm the source address of the host that sent the packet and verify the authenticity of several packets in these large-scale networks. To solve these problems, the IETF SAVI group [6,7] made significant efforts to standardize the mechanism for validating source addresses on access devices, and they developed methods to prevent nodes attached to the same IP link from spoofing each other's IP addresses. SAVI devices validate the source address of a packet based on a binding-validation model. However, the binding relationships cannot be exchanged between devices dynamically, and SAVI devices function individually, which seriously discourages their deployment on large scale networks.

To meet the source address validation requirements in large-scale networks that maintain larger address spaces and dynamic traffic [8,9], SDNs offer an opportunity to overcome the defects existing in traditional architecture. The Open Networking Foundation (ONF) presents a high-level architecture for SDNs [10] that is vertically split into three main functional layers: an infrastructure layer, a control layer, and an application layer. As the SDN controller has a global view that listens

* Corresponding authors.
E-mail addresses: zhouqizhao@hust.edu.cn (Q. Zhou), yjqing@hust.edu.cn (J. Yu), lidong@hust.edu.cn (D. Li).

https://doi.org/10.1016/j.comnet.2021.108075
Received 27 July 2020; Received in revised form 24 March 2021; Accepted 27 March 2021
Available online 6 April 2021
1389-1286/© 2021 Elsevier B.V. All rights reserved.

for all the IP address accessing from hosts, it can centrally maintain a source address binding table and determine the validation rules efficiently. By focusing on the source address validation problem based on SDN, the preliminary design of the SDN-based SAVI [11–13] rapidly revolves SAVI and confirms it to be a reasonable mechanism for implementing IP spoofing management in the SDN-based networks. Unlike conventional methods [14–16], a central controller greatly facilitates the validation task because the SAVI based on SDN can check the validity of each passing packet and drop illegal ones following rules issued by the controller. However, the current method has numerous disadvantages and requires efficiency improvements. Our main contributions are summarized below.

- We designed a dynamic SAVI-based framework for SDN-based networks to facilitate the use of limited resources and provide centralized management.
- In the first level of the flow table, we propose a state partitioning and transitioning model for the dynamic source address validation binding relationships to detect anomaly activities and provide differentiated security management.
- With full autonomy, our model dynamically realizes a balanced flow entry lifecycle according to real-time environmental parameters in an SDN-based network.
- We evaluate the proposed framework in a simulated SDN-based environment based on different performance parameters for more efficient and secure data transmission in a data center network.

The rest of the paper is organized as follows. Section 2 describes the related works and provides the current position of the paper. Section 3 illustrates the background and motivation of our work based on securing source addresses in SDN networks. Section 4 describes the D-SAVI prototype system and presents our method for dynamic and lightweight source address validation, Section 5 evaluates the proposed approach by conducting comprehensive experiments, and finally, Section 6 concludes the paper.

2. Related work

The analysis and detection of spoofed addresses is a vital research area both in academia and industry. We have derived our work from many related studies that focus on the subject of source address validation especially the mechanism based on SDN-based approaches.

Wu et al. proposed a prototype implementation of the IP Source Address Validation Architecture (SAVA) [8], aimed at the numerous attacks based on the forged source address problem. SAVA provides transparent network services and binding relationships to ensure that every packet received and forwarded holds an authenticated source IP address. The IETF working group is the first to standardize the source address validation mechanisms as a prototype system for SAVI [6,7]. By establishing a binding relationship between the IP source address, MAC address, and port of each access host, SAVI deploys the implementation to the switch in the access layer and can filter spoofed packets. Conversely, binding relationships could also alleviate the flood of spoofed source addresses in a distributed environment. Moreover, Virtual Anti-Spoofing Edge (VASE) [14], Virtual Source Address Validation Edge (VAVE) [15,16] and Source Address Validity Enforcement (SAVE) [17] are both implemented under SAVA. VASE configures the export router to check the source address when forwarding packets, and filters out the packets with forged source addresses. However, it fails to block the forged source address within the local area network, especially in the large address space environment. VAVE gave their by establishing a source address protection zone or address bindings. This design is meant to protect users behind SAVI switch from being spoofed by other users in the same domain. However, these schemes must upgrade the underlying network forwarding equipment, and both incur large deployment cost overheads in dynamic topological situations. Few papers are devoted to building a source address validation based on dynamic analysis of the network environment.

In the industry, Google has deployed an SDN to interconnect its data centers across the globe [18], and many big IT companies have joined SDN consortia such as the ONF and the OpenDaylight initiative [19], which confirms the importance of SDN from an industrial perspective. Because SDN is capable of a global topological view and central control patterns, it is one of the most promising solutions and has gained much attention in the source address validation field in recent years. With SDN, traditional source address validation can be interpreted by flow rules in SDN-supported switches, that can be issued by logically centralized controllers based on network real-time situations. Content Delivery Networks Interconnection (CDNi) [20] was a framework used to detect fraudulent IP addresses and defend against attacks generated by them. BGP-based Anti-Spoofing Extension (BASE) [1] is an anti-spoofing protocol designed to fulfill incremental deployment properties. It encrypts IP source addresses with asymmetric key cryptography and verifies the keys to ensure authenticity. However, such designs require additional secure key agreement protocols because key generation and public-key distribution are accomplished by individual hosts, and are not suitable for large-scale networks. Owing to the increasing use of SDN in the management of large networks like cloud data center, a significant body of research has particularly studied the deficiencies and efficiencies of these source address validation systems, and several controller-based approaches have focused on this problem. SDN-based Integrated IP Source Address Validation Architecture (ISAVA) [4] leverages the SDN controller to compute the forwarding path for each prefix pair and tries to upgrade domain routers to accommodate OpenFlow specification. Source Address Validation in Software Defined Networks [11] was the preliminary design and implementation to enable SAVI functionalities in SDN networks. With its deployment, network administrators can now enforce SAVI in their networks by merely integrating a module on the controller, rather than purchasing SAVI-capable switches and replacing legacy ones. SDN-Ti [12] is a general solution to trace back and identify attackers in IPv6 networks based on SDN. SDN-Ti avoids the heavy storage overhead and synchronization problems and supports multiple IPv6 address assignment scenarios. Source Address Validation for SDN Hybrid networks (SAVSH) [13] locates nodes for the SDN switch replacement and deploys filtering rules on them with desirable IP prefix level filtering accuracy. It requires only a few SDN devices but produces a desirable IP prefix level validation effect. More importantly, it can self-adapt to network dynamics. These SDN-based SAVI implementations can provide maximal source address validation effect by deploying minimal SDN switches into traditional networks, which maintains existing network assets maximally to promote incremental system deployment. However, these validation frameworks have difficulty coping with situations in which both the victim and the attacker are in live migration. Under diverse network environment circumstances, current SDN-based SAVI mechanisms could incur large flow table overhead and communication latency in SDN controllers. Thus, SAVI protection is not adequate and attacks can occur by spoofing traffic with source addresses bound on another SAVI device. This drawback discourages the deployment of SAVI in current large-scale SDN-based networks.

3. Background and motivation

On examining RFC7513 [21], it can be observed that, in contrast to simple Access Control Lists (ACL)-based ingress filtering, SAVI takes a binding approach for spoofed address detection and takes (DHCPv4/v6) leases into account. However, the main issue faced in traditional SDN setups with SAVI implementation is that source address validation is performed on the controller, placing the system under high strain during larger spoofing events, and leading to delayed forwarding and additional latency at ingress ports. Fig. 1 illustrates the initial design of the current SDN-based SAVI [11–13]. First, the controller installs several

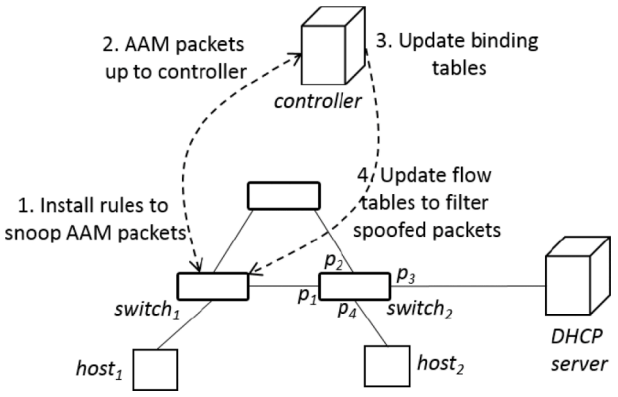


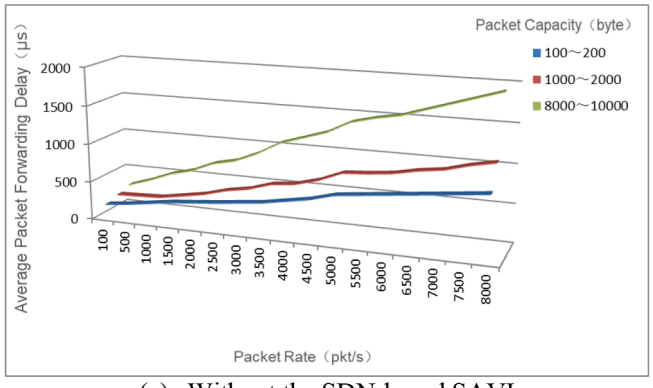
Fig. 1. Initial design of SDN-based SAVI.

flow rules in an OpenFlow switch to snoop Address Assignment Mechanism (AAM) packets. Then the switch will send each AAM packet to the controller as an OpenFlow packet-in. Next, the controller updates the binding tables according to the address assignment state indicated by the snooped AAM packets. Finally, if the binding table is updated, the controller modifies the corresponding flow table of the switch to install or remove a filteried entry. However, the current mechanism still has several limitations.

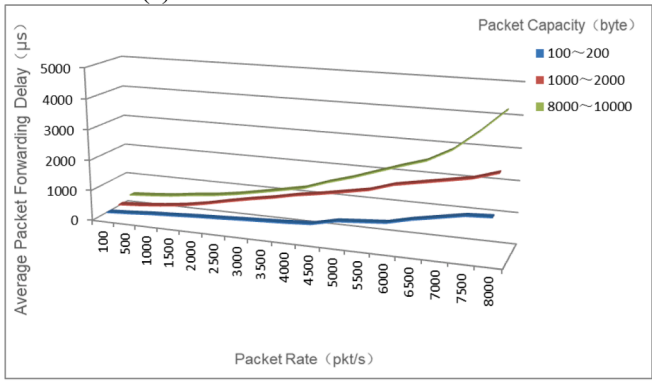
A network performance test was conducted for the initial design of the current SDN-based SAVI, using the experimental network in Section 5. The average packet forwarding delay was tested by obtaining the timestamp data in the Link Layer Discovery Protocol (LLDP), which represents the performance of the communication between controllers and switches in diverse network environment circumstances including different traffic volumes and the anomaly status of the controller.

First, the current design of SDN-based SAVI is not sensitive to real-time traffic in the SDN-based network. As Fig. 2 depicts, the growth trend in the average packet forwarding delay under the deployment of SDN-based SAVI is significantly higher than in the original OpenFlow environment. Therefore, as expected, the network transfer performance will be inferior in a larger-scale network with a huge volume of traffic. The reason can be divided into two aspects. First, the current SDN-based SAVI was designed at a "switch-level". Once a candidate address is found, it will execute source address validation and block all related packets from the switch without considering the differentiated network environment circumstances. A large number of normal packets are still forwarded through the switch, and when faced with drastic changes in network traffic, the continuous burden of packet routing on the controller will reduce packet forwarding efficiency. Conversely, the elephant flows easily to trigger the overflow of the flow table. Due to the large number of flows and the capacity of the packet, such a "switch-level" mechanism will exhaust the Ternary Content Addressable Memory (TCAM) in the flow table [22]. Moreover, the current SDN-based switch was designed to implement priority ranking of the rule block, and the priority of those binding relationships in SDN-based SAVI is static, which is insufficient to provide a quality of service (QoS) guarantee for a network in which the traffic changes dynamically. For example, the priority of the validation module is higher than the AAM snooping module and sequential searching of the binding relationships in the flow rules will indirectly increase the cost of packet forwarding. Knowing these vulnerabilities, some attackers mount an attack on the SDN controller with a large volume of traffic to incapacitate the entire environment and make it harder to validate the authenticity of the source address and select the anomaly packets from normal packets. As a result, it will further delay the forwarding of normal packet and hamper the performance of the SDN controller.

Second, the current SDN-based SAVI was designed without considering the intensity of an attack on the network. The binding relationship is static, which would validate the source address of all incoming packets indiscriminately. According to diverse research (e.g., [23]) a concern about the IP spoofing threat to large-scale networks, e.g. a campus network or distributed data center is that the limited percentage of spoofing packets has a negligible influence on the performance of OpenFlow switches. However, the additional validation process will burden the forwarding of normal packets, provided that the percentage of spoofing packets is low. As shown in Fig. 3, since the binding relationships are installed statically in the current SDN-based SAVI, when the percentage of spoofing packets is lower than 30%−40%, as shown in the line chart, the average packet forwarding delay increases because of the repeated validation. Although the controller can still build and maintain the global binding table for ensuring the IP security of received packets, this flaw reduces the efficiency of normal packet forwarding in the SDN-based network. Furthermore, when a larger percentage of suspected spoofing packets in the network must be validated, the average packet forwarding delay increases again because of the



(a) Without the SDN-based SAVI



(b) With the SDN-based SAVI

Fig. 2. Performances of the SDN-based SAVI under different network conditions.

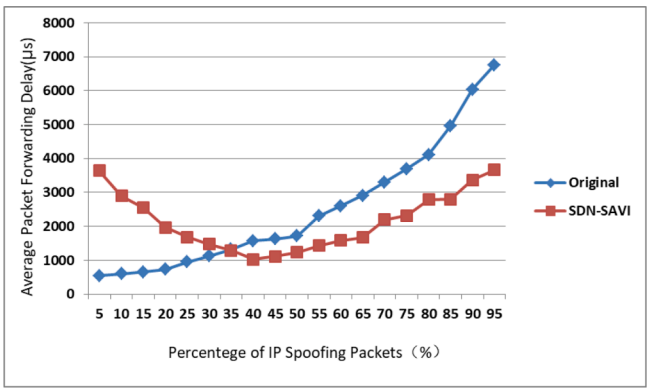


Fig. 3. Average packet forwarding delay under different IP spoofing conditions.

considerable load increase on the SDN controller. Consequently, a flooded controller will exhibit poor responsiveness to the flow requests from other unaffected flow switches and the balance between the security and efficiency of packet forwarding becomes a source address validation bottleneck in the SDN-based environment.

Primarily, although the current SDN-based SAVI mechanism has enhanced security of the source addresses, both the aforementioned limitations may degrade communication performance in the network, which is a significant drawback under diverse network environments. In some cases, when faced with a large number of packets flooded to trigger massive table-misses and packet-in messages in the data plane, the resources of different components of the SDN infrastructure would be exhausted, including TCAM in the data plane, the bandwidth of the control channel, and the controller's CPU cycles. In this study, we design a dynamic and lightweight framework, to optimize the network performance of SDN-based SAVI while maintaining the security level of the source address.

4. Dynamic SAVI solution

To overcome the above-mentioned difficulty, a framework called D-SAVI is presented to realize dynamic and lightweight source address validation in the SDN-based network. D-SAVI takes advantage of the SDN architecture that has a global topological view and central control pattern for lightweight monitoring of global changes and network parameter variations. Fig. 4 depicts the basic architectural components of D-SAVI to secure source addresses in SDN networks with full autonomy. It consists of three main modules: *Flow Table Management, Flow Collector/Extractor, and Dynamic Validation*.

In D-SAVI, only the traffic flowing from/to the monitored asset is mirrored to limit the impact of mirroring on network performance. The *Flow Table Management Module* in D-SAVI optimizes the control of the flow table during the flow entry matching procedure. Instead of deploying source validation without considering the actual traffic and attack condition in the SDN-based networks, D-SAVI optimizes the control of the flow table during the flow entry matching procedure.

For the fine-grained management of the flow table, we divide the flow table into two levels: validation rules and forwarding rules. The flow table converts the source address validation process into "address-level", and the previous burden of source address validation on the controller is partially distributed to the switches. Then, the *Flow Collector/Extractor Module* collects various statistics that the controller obtains from the switch without additional query operations and burdens accompanied by sending the *OFPT_STATS_REQUEST && REPLY* message (Fig. 5) is replied. The collected traffic is periodically extracted and exported to retrieve the flow features for detecting suspicious behavior relevant to the spoofing source address more economically. Finally, inspired by the limitation that the current SDN-based SAVI frequently executes redundant validation of binding rules, we designed a *Dynamic Validation Module* based on the partition and transition state to reduce unnecessary operations. By extensively analyzing the possibility of the candidates being attacked or requiring urgent validation, the packet forwarding delay between the controller and switches decreases. Additionally, a random selection and polling module has been added to real-time monitoring and increases the security level. The module enables the priorities of binding rules to be rearranged and writes them into the flow table by referring to the dynamic QoS of an SDN-based network.

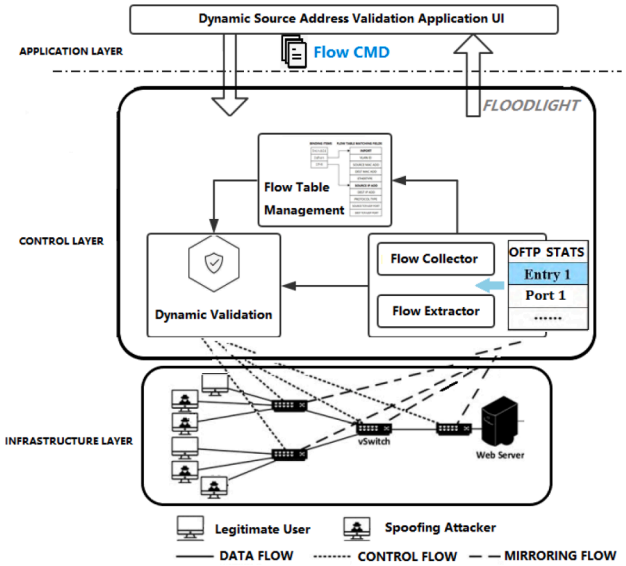


Fig. 4. High-level architecture of proposed D-SAVI.

4.1. Fine-grained control of flow table

The current binding rules in SDN-based SAVI [11] are enforced by flow rules in switches, which bind source addresses to switch ports in the <Address, Switch, Port> format. For these rules to coexist with the flow rules generated by other applications such as forwarding, the SDN-based SAVI prototype uses OpenFlow's "multiple tables" feature (Fig. 6). With this as a reference, we designed a priority-based dual flow table to mitigate the insensitivity of the current SDN-based SAVI to the diverse network environment circumstances.

Note that a packet is not necessarily required to match all the binding relationships for the source address validation before it is forwarded. Here we divide the current flow table into two levels for source address validation and routing decision. Specifically, as Table 1 depicts, in the

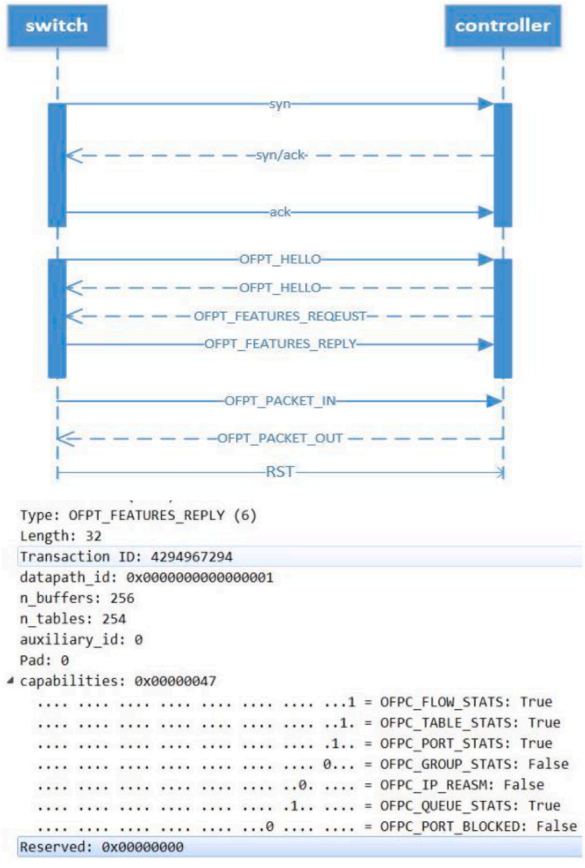


Fig. 5. Typical example of OFPT_STATS_REQUEST && REPLY message in SDN networks.

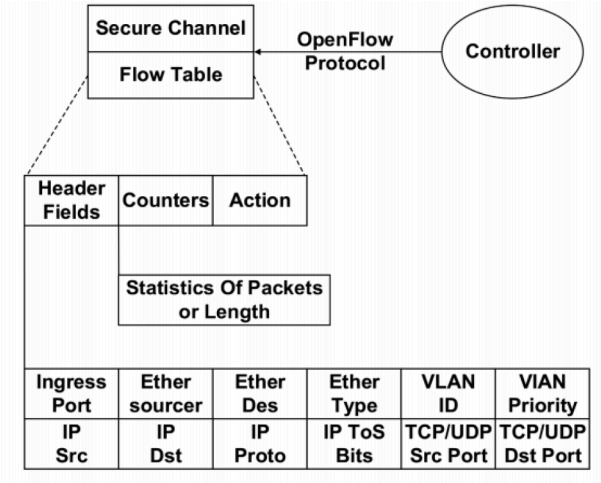


Fig. 6. Content of original SDN flow table.

Table 1
Example of dual flow table and actions.

No	First level			Second level	
	Switch port	Source IP	Source MAC	Action	Matched packet number
1	P1	H1_IP	H1_MAC	Goto	N1
2	P1	*	*	Drop	N2
.....					
m	*	*	*	Goto	Nm

Flow Table Management module, we divide the flow rule table into two levels: validation and forwarding rules. The binding relationship in the first level of the flow table is used to store the host network address binding condition. To enhance security and optimize the efficiency of the source address validation mechanism, the corresponding match fields in the second level of the D-SAVI flow table are only responsible for forwarding packets. Further, since the binding format <Address, Switch, Port> overlaps with the content of the current matching fields of OpenFlow v1.5, we integrated the program of source address validation with the existing flow table matching fields (Fig. 7). In this design, the validation of the source address is more fine-grained. By establishing such "address-level" relationships, the validation process in the SDN-based SAVI could make reasonable use of the processing capabilities of OpenFlow's switches, and the previous burden of source address validation on the controller could be partly distributed to the switches in the access layer, which is more appropriate for high-traffic environments like cloud data centers.

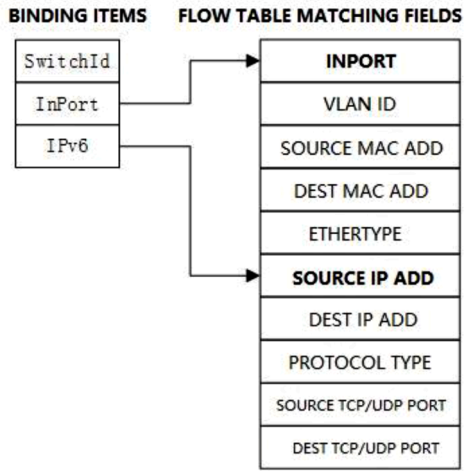


Fig. 7. Binding relationships in the proposed D-SAVI.

The process of controlling the fine-grained flow table that consists of the validation and forwarding processes is illustrated in Fig. 8. It is composed of. When the newly arriving packet enters the access layer switch, it will be validated by a two-level flow table, and only the packet that is correctly matched and validated by the binding relationships will enter the next step for further packet forwarding. In summary, the benefits gained by adopting a fine-grained dual flow table structure include the following.

- The previous burden on the SDN controller of source address validation could be partly alleviated because of the redundant binding relationship validation being reduced.
- D-SAVI ensures differentiated executions in the forwarding of normal packets and the discarding of spoofing packets under "address-level" distinctions.
- The SDN controller may rearrange the priority of the binding table and write the binding relationship into the table according to the statistical results obtained by the dual flow table structure.

4.2. Lightweight feature collection

As repeated validation limits the efficiency of packet forwarding, one reasonable method to reduce the redundant validation process is to classify the candidate addresses in binding relationships. To obtain real-time features in the SDN-based network, the SDN controller encapsulates the *Read-State* information into the *Packet-in* packet, which is periodically delivered between the SDN controller and virtual flow switches to provide statistical information. With the *OFPT_MULTIPART_REPLY* message, this periodic operation in OpenFlow v1.5 allows us to collect various statistics that the controller obtains from the switch without additional query operations and burdens. The periodic operation is lightweight, is easily integrated with existing SAVI modules, and

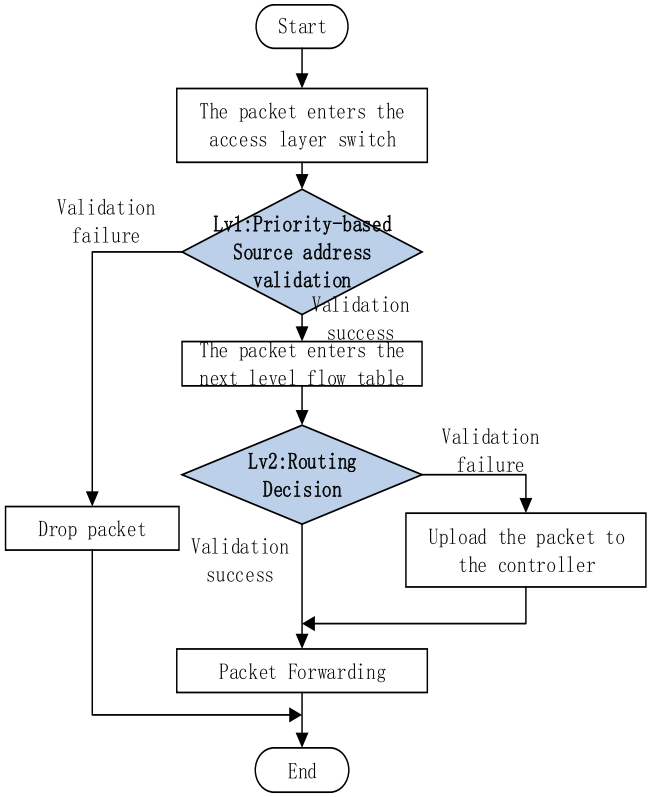


Fig. 8. Packet processing architecture in the dual flow table.

naturally dovetails with the goals of the classification model.

We implemented a *Flow Collector/Extractor* module (Fig. 9) in the SDN controller by utilizing the multi-threaded approach in JAVA. Typical statistics (Table 2) collected from the OpenFlow vSwitch (OVS) in the access layer could be used to recognize different source addresses with very an overall network load that was extremely low. To make better use of CPU time, we created threads for all the OVSs in the access layer to process the data. By maintaining a pool of running threads, there was no thread startup and shutdown overhead when they were run. Subsequently, to reduce the expensive access to archive data, the characteristics of the aggregate data query were analyzed, and the data collection period was limited. Overall, by utilizing such a multi-threaded data collection module, the SDN-based network feature can be gathered and analyzed through the switch in full detail, as there is no sampling involved in the collection process. The lightweight feature collection module can be employed as a basic program block for the analysis and classification of the spoofing address in the following parts.

4.3. Dynamic validation model

To counter the problem that the current SDN-based SAVI was designed without considering the intensity of a spoofing attack, we propose a state partition and transition mechanism (Fig. 10) in the binding relationships for source address validation in the first level of the fine-grained flow table. Network performance will degrade because

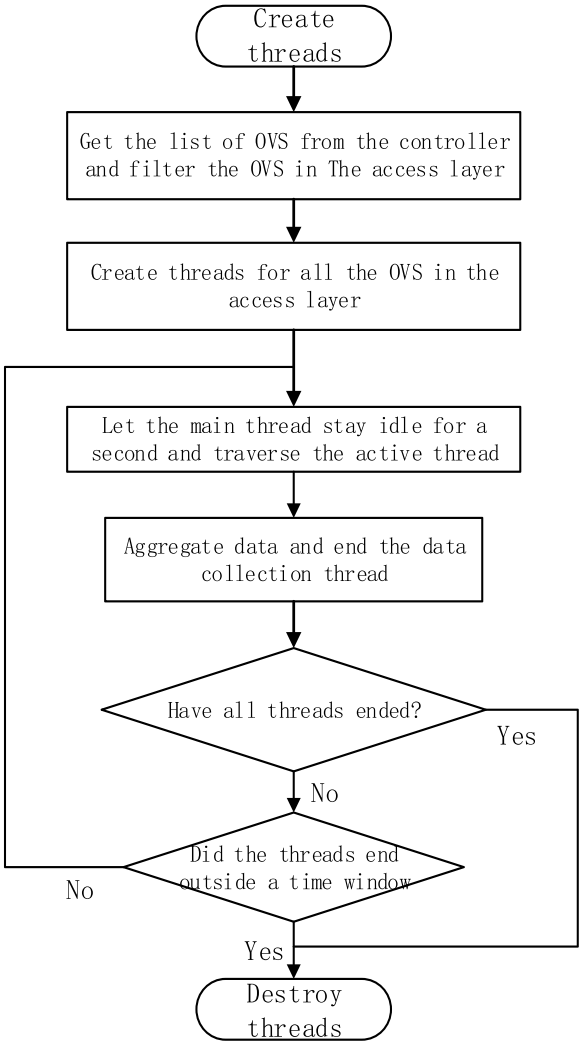


Fig. 9. Multi-threaded flow collector/extractor module.

Table 2
Typical example of features by OFPT_STATS_REQUEST querying.

Category	Feature
Flow	Number of successfully matched packets
	Flow duration
Flow table	Active number of flow entries
	Number of historical queries in each flow entries
	Active match fields in each flow entry
Port	Port throughput
	Number of port fault occurring
	Number of packets in each port
Queue	QoSs statistics

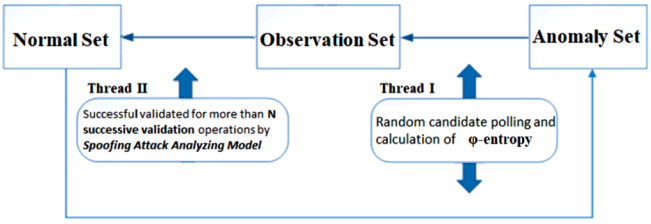


Fig. 10. Partitions and the state transfer conditions of the candidates' states.

of the repeated pooling operations done by the SDN controller; therefore, the core idea of this module is to optimize network performances while ensuring security. As we mentioned above, the anomaly flows and spoofing attacks are both key factors in establishing whether a candidate requires validation. Host have more legitimate services that access the network, and we intended to classify various candidate addresses to reduce the polling number when the source address validation module is activated. First, in this study, we made the following assumptions about the standard of anomaly candidates.

- Among those characteristics of IP spoofing attacks, the SYN flooding and SMURF style attacks [24] are the most frequent. In such cases, it is unlikely that the packets arriving at switches during an IP spoofing attack match an entry in the switch's flow table. Thus, counters in the flow table have a high probability of being unpaired addresses.
- The launch of a large-scale IP spoofing attack is usually accompanied by drastic changes in network traffic, especially the incoming traffic of the switch port in the access layer [25].

In terms of implementation, we applied the state partition to the candidate addresses and classified the states of candidates into three types. Fig. 10 depicts the partition of the candidates' states and the conditions when executing a transition from one state to another. An important reason for this issue is that the current SDN-based SAVI requires repeated source address validation on all candidate ports constantly. All the existing binding relationships are regarded as a candidate in an *Anomaly Set*, which means that these binding rules are activated continuously to ensure the security of the source address in the SDN-based environment.

4.3.1. Entropy-based anomaly alert

We propose an entropy-based anomaly alert model to monitor the real-time state of network flows. Shannon's entropy is a concept in information theory that is used to measure randomness or uncertainty. Alternatively, it could also be described as a measure of the association between information content and a random variable. Commonly, higher entropy means a system with higher randomness. When a spoofing attack or elephant traffic occurs in SDN networks, the distribution of flow features will significantly change. By analyzing the entropy upheaval, it is theoretically possible to classify preliminarily the candidate with a suspected forged source address in the network. As the source address spoofing attack is usually accompanied by drastic changes in

network traffic, when monitoring changes in entropy, it would be a prudent to suspect that a candidate host is an anomaly.

In [26], the authors proposed a new information theory measure φ-entropy based on Shannon's entropy. As Eq. (1) depicts, p_i denotes the probability of occurrence of event X and the parameter φ is used to adjust the sensitivity of measuring the frequency of events. In such cases, the argument about the monotonicity of φ-entropy holds that when φ>0, $H_\varphi(x)$ increases monotonically with respect to the parameter φ but to be accurate, the increase and decrease of $H_\varphi(x)$is related top_i. According to the theory of information entropy, when a spoofing attack occurs, the degree of randomness of the host address in the flows will decrease, and the overall entropy value will decrease. The follow-up works [27,28] proved that when the parameter $\phi > \ln 2$, the decision function of $H_\varphi(x)$ is first a concave and then a convex at (0,1).

$$H_\varphi(x) = -\frac{1}{\sinh(\varphi)}\left(\sum_{i=1}^{n} p_i \sinh(\varphi \log_2 p_i)\right) \tag{1}$$

Our entropy-based solution is based on the φ-entropy (Eq. (1)) of matching fields and events namely the source IP address (sIP), source port (sPort), destination IP (dIP), and destination port (dPort) during a time-window. Features obtained by the *Flow Collector/Extractor* module are subsequently fed to the module periodically. In our case, a time window of 30 s was chosen to accomplish near real-time detection consistently with related literature [29]. In our simulation experiment on SDN networks, we set φ as 0.3. Commonly, in a real network environment, φ can be adjusted according to the actual network conditions, and then a suitable threshold can be selected. In the initial stage of the spoofing attack, or a relatively slow spoofing attack, the characteristics can be appropriately enlarged for issuing early and more accurate warnings.

In the actual process, the matched fields of each candidate binding relationship are parsed. Then, $H_\varphi(x)$can be obtained by substituting it in Eq. (1). Naturally, when all destination IP addresses in the network managed by the controller have the same probability of occurrence, $H_\varphi(x)$achieves its maximum value. As Fig. 11 depicts, the φ-entropy of dIP, dPort, sIP and sPort will change during the predominance of attack flows and elephant flows on the SDN networks. Long-lived elephant flows normally transport large volumes of data in enterprise networks and tend to consume substantial bandwidth and fill network buffers from end to end. As a result, a continuous increase in the φ-entropy significance of dIP and dPort could be tested because the concentration of packets for a particular destination IP and port will be very much greater than another destination IP. Subsequently, as the packets arriving at the switches are unlikely to match an entry in the switch's flow table during the spoofing attack, no malicious packet-in messages will be generated after the delivery of a corresponding flow, leading to a sudden decrease in the φ-entropy of sIP and sPort. Consequently, the suspected abnormal state would be detected when the standard deviation calculated by the entropy-based anomaly alert model exceeds the threshold. Since the value of the φ-entropy threshold is important, it is evaluated in Section 5.2.

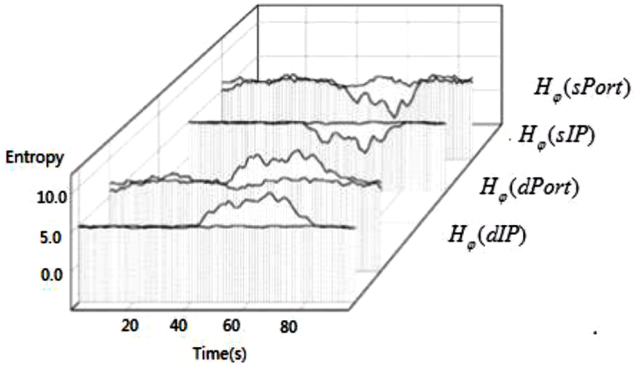


Fig. 11. φ-Entropy of dIP, dPort, sIP, and sPort.

Because the limited percentage of spoofing packets has a negligible influence on the performance of OpenFlow switches in a large-scale network, the normal candidates will be transitioned to *Observation Set* for further detailed detection. Conversely, those candidates which trigger drastic changes in traffic will be transitioned back to *Anomaly Set* periodically to minimize the potential risk in the SDN-based environment.

4.3.2. *Xgboost-based spoofing attack detection*

To filter redundant candidates in *Anomaly Set*, we designed a model to classify and locate the candidate in *Observation Set* that has a real impact on the network and urgently requires source address validation. Commonly, the number of failed matches recorded by the controller could be seen as significant proof of a host or port requiring repeated validation. We designed a spoofing attack analysis model on the XGBoost framework [31,32] with reference to the work in [30], to analyze the real-time condition of whether a candidate is benign to the SDN-based networks. Combined with the global perspective and centralized control of SDN, we analyzed the observed statistics in the entire SDN network to decide whether a candidate requires urgent validation. The classification model under XGBoost automatically utilizes the multi-core of a single machine's CPU for parallel computation, and on a scale beyond billions of examples calculated out-of-core while using far fewer resources than present systems. These features enable our classification model to detect potential spoofing attacks effectively and save the controller's memory resources.

In our classification model, we used the latest comprehensive InSDN [33] dataset, which is used as a training set to verify the performance of intrusion detection systems. The new dataset includes benign and various attack categories that can occur in the different components of the SDN platform. Table 3 represents the all the selected features for the SDN context from the InSDN data. They are composed of eight types of characteristics, namely Network-identifiers attributes (Nos.1–7), Byte-based attributes (Nos.8–9), Packet-based attributes (Nos.10–17), Inter-arrival Times attributes (Nos.18–21), Flow timers attributes

Table 3
Typical features in the InSDN dataset.

No.	Attribute name	No.	Attribute name
1	Flow-id	29	SYN-Flag-Cnt
2	Src-IP	30	RST-Flag-Cnt
3	Src-Port	31	PSH-Flag-Cnt
4	Dst-IP	32	ACK-Flag-Cnt
5	Dst-Port	33	URG-Flag-Cnt
6	Protocol-Type	34	CWE-Flag-Cnt
7	Timestamp	35	ECE-Flag-Cnt
8	Fwd-Header-Len	36	Down/Up-Ratio
9	Bwd-Header-Len	37	Fwd-Seg-Size-Avg
10	Tot-Fwd-Pkts	38	Bwd-Seg-Size-Avg
11	Tot-Bwd-Pkts	39	Fwd-Byts/b-Avg
12	TotLen-Fwd-Pkts	40	Fwd-Pkts/b-Avg
13	TotLen-Bwd-Pkts	41	Fwd-Blk-Rate-Avg
14	Fwd-Pkt-Len	42	Bwd-Byts/b-Avg
15	Bwd-Pkt-Len	43	Bwd-Pkts/b-Avg
16	Pkt-Len	44	Bwd-Blk-Rate-Avg
17	Pkt-Size-Avg	45	Init-Fwd-Win-Byts
18	Duration	46	Init-Bwd-Win-Byts
19	Flow-IAT	47	Fwd-Act-Data-Pkts
20	Fwd-IAT	48	Fwd-Seg-Size-Min
21	Bwd-IAT	49	Flow-Byts/s
22	Active-Time	50	Flow-Pkts/s
23	Idle	51	Fwd-Pkts/s
24	Fwd-PSH-Flags	52	Bwd-Pkts/s
25	Bwd-PSH-Flags	53	Subflow-Fwd-Byts
26	Fwd-URG-Flags	54	Subflow-Fwd-Pkts
27	Bwd-URG-Flags	55	Subflow-Bwd-Byts
28	FIN-Flag-Cnt	56	Subflow-Bwd-Pkts

(Nos.22–23), Flag-based attributes (Nos.24–35), Flow-based attributes (Nos.36–52), and Subflow-based attributes (Nos.52–56). For the training set, we use a training subset that is 10% of the record.

However, it would be wasteful of system resources to process these multidimensional features when the increase in classification precision is very limited. Reducing the number of features is a significant step for building a lightweight D-SAVI framework; therefore, we traversed each feature, selected each value of the feature as its splitting point, and calculated the gain-loss using XGBoost. After all the features were traversed, the most characteristic value of the gain-loss was the split point. The value of importance could be convincing evidence that a feature has the maximum information gain and chi-square statistic for spoofing attack detection. As a result, we ranked these features and selected the 16 most important noes (he features highlighted in blue in Table 3) for the spoofing detection module. The evaluation of the selected features is discussed in Section 5.2.

Then, we differentiated the candidates based on certain significant features related to the spoofing source address, and categorized them as malicious or benign to the network environment. As for the condition of a state transition, if there is no potential flow that exceeds the frequency or capacity threshold in the SDN-based environment, then, the candidates are successfully validated after more than *N* successive validation operations are transferred to the *Normal Set*. In terms of lightweight designing, the selection of *N* should balance the performance of the controller and switches (which will be verified in Section 5.4).

4.3.3. Priority-based flow entry

To a modern SDN switch with space for at most thousands of flow entries, an explosion of flow tables could pose a severe problem to forwarding in the SDN-based networks. The validation rules of D-SAVI might be either rarely matched or frequently used within a given time. Therefore, they may have different importance or priority values to maintain the different capacities of TCAM. By focusing on the static priority of flow entry in current SDN-based SAVI, which is not sufficient to provide lightweight QoS guarantees, we further designed a dynamic binding mechanism in D-SAVI with priority-based validation rules as shown in Fig. 12. In our optimized validation rules, each OVS port in the access layer has a validation rule and a corresponding wildcard rule, which lets the switch establish an appropriate trust binding relationship between the network address and the switch port by analyzing the related address and allocation packets. The priority of the wildcard rule was set lower than the original to ensure that the spoofing binding relationship could be preferentially matched. Therefore, if the packet successfully matches the second rule, it must fail to match preceding rule. In such cases, the forwarding of packets from other unaffected source addresses can be smoothed. After a few priority-based rearrangements, the space occupation of TCAM is lower than the original flow table because the low-frequency binding relationships would be deleted automatically at the end of the timeout. With the savings of TCAM space compared to a conventional SDN-based SAVI, this module is expected to benefit including minimizing the packet forwarding delay and fast configuration of the flow table.

```
1. <InPort=p4, SrcIP=*, Action=goto table2>
2. <InPort=p1, SrcIP=h1, Action=goto table2>
3. <InPort=p2, SrcIP=h2, Action=goto table2>
4. <InPort=p3, SrcIP=h3, Action=goto table2>
…
m. <InPort=*, SrcIP=*, Action=Drop>
```

(a) SDN-SAVI's validation rules

```
1. <Priority=5, InPort=p1, SrcIP=h1, Action=goto table2>
2. <Priority=4, InPort=p1, SrcIP=*, Action=Drop>
…
n. <InPort=*, SrcIP=*, Action=goto table2>
```

(b) Dynamic SDN-SAVI's validation rules

Fig. 12. Typical examples of D-SAVI's validation rules.

For implementation, as depicted in Fig. 13, the forwarding module in our fine-grained flow table will check the interaction between the address allocation packets between the hosts and address allocating servers such as DHCPv6 packets. Then, these copies of the control packets are sent to the controller for further analysis. Among the real-time statistical data, we calculate the ratio of dropped packets in the candidate relationships using Eq. (2). The ratio of dropped packets could be convincing evidence of whether a candidate binding relationship is matched at high frequency. The related network device is safer when the ratio of dropped packets is lower. The controller could rearrange the priority of the binding table and write the binding relationship into the table according to the various *DropPacktesRatio*. As a result, the packet will match the highest priority binding rules and the most reasonable rules. Details of the efficient optimization due to the rearrangement of *DropPacktesRatio* in a priority-based flow entry can be found in Section 5.3.

$$DropPacketsRatio = \frac{DropPackets}{DropPackets + NormalPackets} \tag{2}$$

4.3.4. Differentiated security management

Generally speaking, the false positives of the attack analyzing model would possibly occur in a non-attack state [25]. In the case where the state partition and transition model could not override every kind of spoofing attack on the SDN-based environment, we added a random selection and polling module to realize real-time monitoring of the suspected anomaly candidates in the last time window. A certain percentage (which is tested in Section 5.4) of all the candidates, according to their transitional order, were chosen as suspected spoofing candidates for random validation. We retested it with the entropy-based anomaly alert module, and those candidates that trigger the anomaly alert would be transitioned back to the *Anomaly Set* for comprehensive validation. As a result, the random selection and polling module could keep track of the source addresses in the binding relationships against the potential risk of a spoofing attack.

Then, to provide differentiated security management in D-SAVI, which is elaborated in Section 5.4 after candidate binding relationships are classified, differentiated actions (Fig. 14) must be taken to reduce the redundant validation process. The differentiated security guarantee works with high efficiency if the source address validation task is huge, Also, in consideration of the centralized control feature of the SDN network, we can easily trace back to the candidates that contain the attackers by marking every packet with <Address, Switch, Port>, and then perform the source filtering at the source address.

5. Result and analysis

Considering that the limitations in current SDN-based SAVI might degrade communication performance in large-scale SDN-based networks like a cloud data center, we test the proposed scheme in a simulated SDN-based data center network to validate the correctness of the

Rule ID	Priority	InPort	Source IP	Other	Successful Matches	Counters
1	a	p1	h1	……	→	N1
2	a	p2	h2	……	→	N2
3	a	p3	h3	……	→	N3
……						……
m	0	*	*	……	→	Nm

Fig. 13. Priority-based flow table in proposed D-SAVI.

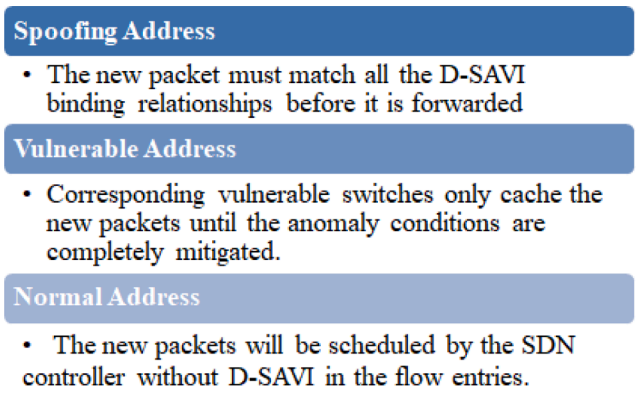


Fig. 14. Differentiated security management provided by D-SAVI.

logic and evaluate the performance of D-SAVI. Furthermore, we compare our implementations with the original OpenFlow environment and SDN-based SAVI scheme (SDN-SAVI) [11–13] in terms of efficiency and security.

5.1. Experimental infrastructure and methodology

Our experimental platform consisted of four DELL servers, two as the host node and the others as simulated edge switches. Each server was equipped with two Intel(R) Xeon(R) APU E5-2630 2.4 GHz 16-core processors and 64GB RDIMM memory. The OS used in each node is Ubuntu Server 16.04.2.

We simulated the multi-layer host topology based on Fat-Tree [34] which is a popular architecture in a distributed cloud data center. We set the bandwidth capacity for the core, aggregation, and edge layers' links to 1 Gbps, 4.8 Gbps, and 8 Gbps respectively. The basic information of the simulated topology in our experiments is shown in Fig. 15. We used two OVS switches [35] and Floodlight[1] as the SDN core switches and controller respectively. To simulate the topology and links between the host nodes for SDN, we use Mininet 2.3.0,[2] which supports OpenFlow v1.5 standard, to run a simulated and easily connected SDN-based network.

In general, two OVS switches were used as the core switch (S1-S2), and then the core switches (S3-S6) and edge switches (S7-S14) were simulated in the other servers. There were 40 nodes and 64 links in this testbed. The Distributed Internet Traffic Generator (DITG)[3] and workloads generated by SWIM[4] [36] were used to provide legitimate elephant traffic (Table 4).

We synthetically generated datasets for spoofing attacks based on spoofed addresses. As most spoofing attacks are launched using botnets, we employed BoNeSi [37] to simulate a UDP flood. BoNeSi could generate TCP, ICMP, UDP, and SYN flooding attacks based on a defined botnet size with different IP addresses. It is highly configurable because rates, data volume, source IP addresses, URLs, and other parameters can be adjusted. According to Akamai's Prolexic Q2 2014 Global DDoS Attack Report [38], the simulation of the spoofing attack is carried out in a certain percentage of TCP, ICMP, UDP, and SYN flooding, and is generated with 30k bots from the nodes of each attacker, which include H1, H6, H11, H16, H21, H26, H31, and H36, with different packet speeds (Table 5) and a constant packet size of 500 bytes for 60 s. To compare the different rates of incoming packets, we controlled the rate of normal traffic and attack traffic to increase and decrease the intensity of the attack on the SDN controller.

[1] https://github.com/floodlight/floodlight
[2] http://mininet.org/
[3] http://traffic.comics.unina.it/software/ITG/
[4] https://github.com/SWIMProjectUCB/SWIM

We divided the intensities of spoofing attacks into five different levels to quantitatively analyze the impact that source address validation has brought to the SDN networks. The different types of spoofing attack scenarios are shown in Table 6. To evaluate the network performance with source address validation schemes, we tested the benchmarks by obtaining the timestamp data in LLDP. Theoretically, this value depends on the distance from the vicious candidate to its nearest SDN checkpoint, link bandwidth, and traffic situation. Thus, we unified the properties of the transmitted packets and topology during the experiments.

5.2. Algorithm analysis

In this section, we provide the results and analysis of the algorithms applied in D-SAVI. Initially, the benchmark of XGBoost-based spoofing attack classifiers was calculated and we compared it with five other machine learning algorithms, including Decision Tree (DT) [39], Random Forest (RF) [40], the K-nearest Neighbor classifier (KNN) [41], Naive Bayes (NB) [42], and Support Vector Machines (SVM) [43]. As shown in Table 7, the choice of XGBoost could be justified by its excellent comprehensive performance on precision, recall and F1-Score under low training-time conditions.

As mentioned in Section 4.3.2, we selected the 16 most important features (The characteristics of the blue background) that have the maximum information gain and chi-square statistic for spoofing attack detection. To evaluate the selection of the number of features using Xgboost, we first listed 56 features, and then ranked them from most to least important. The experimental results in Fig. 16 show the performances of these features on the InSDN dataset. Along with the increase in the number of features, the selected features effectively improve the accuracy and the precision of the spoofing attack detection model. When the number of features reaches 15–20, there will be no significant increase in accuracy and precision. The empirical results show that the detection accuracy remains the same or improves marginally when using only the 16 most important attributes compared to all the 56 attributes based on the detection method.

For evaluating the performance of the entropy-based anomaly alert model, the value of the threshold of φ-entropy plays an important role. In a real network environment, parameter φ can be adjusted according to the actual network conditions and a suitable threshold can be selected. Thus, the estimation of the threshold may affect the overall efficiency of a detection system. In the simulation experiment, we set φ as 0.3. We ran an attack based on Type A (15% Spoofing Packets) on a single host 50 times to find a suitable threshold. This threshold is the highest entropy of all cases, which enables the controller to detect any attack with packets contributing to 15% or more of the incoming traffic. Table I shows the threshold compared to normal traffic values.

As shown in Table 8, the entropy of normal and attack traffic was measured. To determine the threshold for this experiment, packets were sent at a high-intensity during Type A attacks and were compared to normal traffic to find the attack thresholds. In the case of $\varphi = 0.3$, $MaxA > MinN$ and $MaxN > MinA$, which confirm a good distinction between attack traffic and normal traffic. Therefore, considering the false alarm rate and detection rate comprehensively, in the following experiments, the average value of both (*MaxA, MinN*) and (*MaxN, MinA*) can be used as the detection threshold of our entropy-based anomaly alert module.

5.3. Efficiency optimazation

Our D-SAVI extended the source address validation method and founded a remedy for defects by dealing with the source address validation during anomalous conditions in the network. Also, the packet will match the highest priority binding rules and the most reasonable rules because of the priority-based flow entry.

Initially, we measured the influence on network performance among different network types and topologies and repeated the experiment 30 times. Here we demonstrate the evaluation results in Table 9 and we

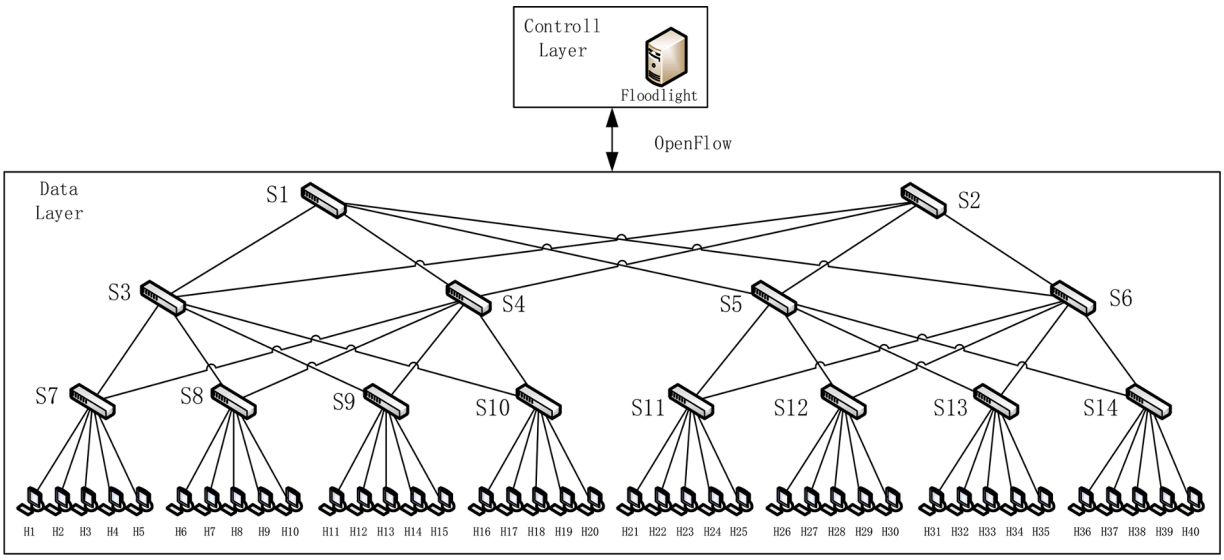


Fig. 15. Topology of multi-layer SDN-based data center network in the simulation experiment.

Table 4
Information of simulated background traffic.

Experiment time	600s		
Attack duration	60s		
Attack Type	Constant Rate		
Attack Traffic Type	SYN/UDP/TCP/ICMP		
Elephant statistics	Threshold of Elephant flow		
	1	10	30
Total number of elephants	2760	362	91
Average active elephants	1062	186	56
Average duration (s)	443.9	595.1	715.4
Traffic contribution (%)	77.6	47.1	28.3
Count contribution (%)	0.2	0.03	0.008
Upstream traffic			
Mean packet size (bytes)	32.46		
Mean inter-departure time (ms)	2.82		
Downstream traffic			
Mean packet size (bytes)	1014.31		
Mean inter-departure time (ms)	1.51		

Table 5
Descriptions of spoofing attacks.

Attackers' nodes	Normal packets (packet/s)	Attack packets (packet/s)	Attack type
H1	50	100	84.1%SYN flood, 8.9%UDP flood, 3.1%TCP flood, 0.6% ICMP flood
H6	50	200	
H11	500	300	
H16	500	400	
H21	2000	500	
H26	2000	600	
H31	5000	700	
H36	5000	800	

provide the results of average packet forwarding delay in different network conditions. Due to redundant processes of source address verification, the deployment of SDN-based SAVI has burdened the communication between the controllers and switches apparently in attacks of Type A and B, and D-SAVI could ease the situation. Refer to the network performance improvement for details in Table 9, D-SAVI has better efficiency (38.4% and 13.4%) when the network is operating

Table 6
Percentage of the spoofing packets in the simulated experiments.

Attack type	Percentage of spoofing packets (%)	Percentage of normal packets (%)
(A)	15	85
(B)	30	70
(C)	50	50
(D)	65	35
(E)	80	20

Table 7
Metrics of performance for the SDN specific-featured version of InSDN dataset.

Algorithm	Precision	Recall	F1-score	Training time (sec)
XGBoost	0.9783	0.9913	0.9848	11.958
DT	0.9712	0.9824	0.9767	25.284
SVM	0.8161	0.9943	0.8964	51.873
KNN	0.9837	0.9498	0.9664	78.512
RF	0.9793	0.5394	0.6954	4.283
NB	0.7459	0.9671	0.8422	2.137

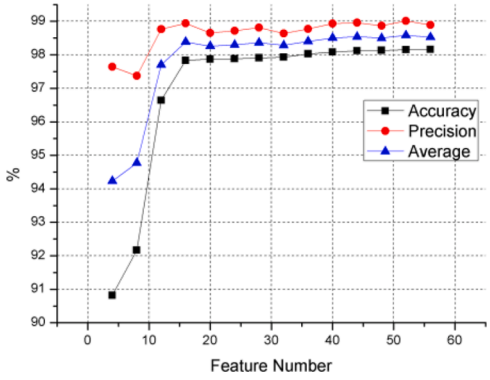


Fig. 16. Performances of features in the InSDN dataset.

normally with limited anomalies. Our D-SAVI is superficial because it avoids the unnecessary matching of the binding relationships for the verification before packet forwarding.

For a larger-scale attack under Type C, the impact on the network

Table 8
φ-Entropy of normal and spoofing attack traffic.

Normal traffic $\varphi = 0.3$	Average entropy	Standard deviation	Minimum entropy	Maximum entropy
sIP	2.359	0.0126	2.294	2.406
sPort	2.172	0.0139	2.062	2.281
dIP	2.231	0.0103	2.157	2.304
dPort	2.078	0.0093	1.972	2.189
Attack type A $\varphi = 0.3$	Average entropy	Standard deviation	Minimum entropy	Maximum entropy
sIP	2.695	0.0198	2.562	2.823
sPort	2.457	0.0143	2.324	2.575
dIP	2.076	0.0087	2.011	2.142
dPort	1.934	0.0064	1.893	1.966
Feature	Minimum entropy of normal traffic (MinN)	Maximum Entropy of Attack Type A (MaxA)		$\frac{MinN + MaxA}{2}$
sIP	2.406	2.562		2.484
sPort	2.281	2.324		2.303
Feature	Maximum entropy of normal traffic (MaxN)	Minimum Entropy of Attack Type A (MinA)		$\frac{MaxN + MinA}{2}$
dIP	2.157	2.142		2.150
dPort	1.972	1.966		1.969

performance by current SDN-based SAVI is reduced when the intensity of the spoofing attack increases. The main reason for such reduction is that when there are several IP spoofed addresses on the SDN-based network, the time cost of packet forwarding increases dramatically. Especially, in some extreme situations with serious attacks, the original OpenFlow environment without protection will lead to the collapse of the SDN controller, and the forwarding of normal packets will also be affected severely. Thus, compared to the original OpenFlow, there are improvements in the network performance of SDN-based SAVI and D-SAVI. Consider a circumstance in which an increasing number of spoofing packets requires urgent validation for detailed comparison under attack Types D and E. We can observe that both SDN-based SAVI and D-SAVI are superior to the original OpenFlow in quantitative indications. A possible conclusion is that the state partitioning and transitioning model is effective in reducing the frequency of candidates polling during address validation in systems.

However, it can be concluded from Table 9 that when the number of redundant verifications is limited, the advantage provided by D-SAVI reduces for attack Type E compared with attack Types A and B, which is a side effect of redundant anomaly analysis in D-SAVI. In general, compared with original OpenFlow and SDN-based SAVI, the average packet forwarding delay is significantly optimized by deploying D-SAVI.

Further, as mentioned SDN-based SAVI is not sensitive to real-time traffic in the network; therefore, we tested D-SAVI under similar circumstances. It can be concluded from Fig. 17 that the growth trend of average packet forwarding delay in a network with the deployment of D-SAVI is slower than the scenario with SDN-based SAVI (Fig. 2), which indicates that our D-SAVI is also more sensitive to real-time traffic especially when the packet capacity is larger. Conversely, as Fig. 3 depicts, when the percentage of spoofing packets is lower than 30%−40%, the average packet forwarding delay increases because of the repeated validation process. Thus, we tested D-SAVI under different spoofing packet percentages. By taking a closer look at our results (Fig. 18), we notice that our proposed framework is superior when the percentage of spoofing packets is limited. As we classify various candidates to reduce the polling number when the source address validation module is activated, these optimizations are to be expected. To summarize, along with the increased attack intensity, the rearrangement of the priority of binding rules in D-SAVI effectively alleviates the pressure of packet forwarding during source address validation, which are convincing demonstrations of our framework's effectiveness and feasibility. However, there is still room for future research to avoid false positives in anomaly analysis.

5.4. Security guarantee

As mentioned in Section 4.3, our D-SAVI has divided the suspected candidates into three states, and the security requirement of source address validation cannot be fully guaranteed automatically. To avoid this disadvantage, we added a random selection and polling module to realize real-time monitoring of the suspected anomaly candidates in the last time window. In this section, w, we concentrate on the D-SAVI

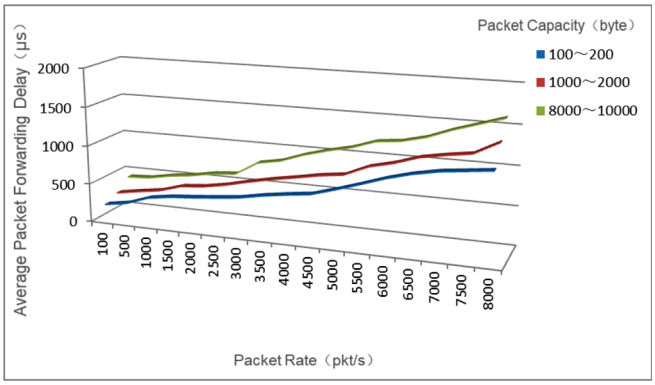


Fig. 17. Performances of the D-SAVI in different network conditions.

Table 9
Evaluation of source address validation frameworks.

TYPE	Benchmark	Average packet forwarding delay (μs)			Optimization percentage (%)	
		OPENFLOW	SDN-SAVI	D-SAVI	Over OPENFLOW	Over SDN_SAVI
A	Average	920	3661	2256	−145.217	38.376
	Standard deviation	117	224	251		
	Confidence interval (95%)	±83	±162	±180		
B	Average	1729	2154	1865	−7.865	13.417
	Standard deviation	159	178	92		
	Confidence interval (95%)	±114	±128	±66		
C	Average	2144	1419	1297	39.506	8.598
	Standard deviation	125	203	178		
	Confidence interval (95%)	±90	±146	±126		
D	Average	3298	1683	1255	61.947	25.431
	Standard deviation	213	198	156		
	Confidence interval (95%)	±153	±142	±122		
E	Average	4037	2977	2842	51.895	4.537
	Standard deviation	259	176	223		
	Confidence interval (95%)	±186	±126	±160		

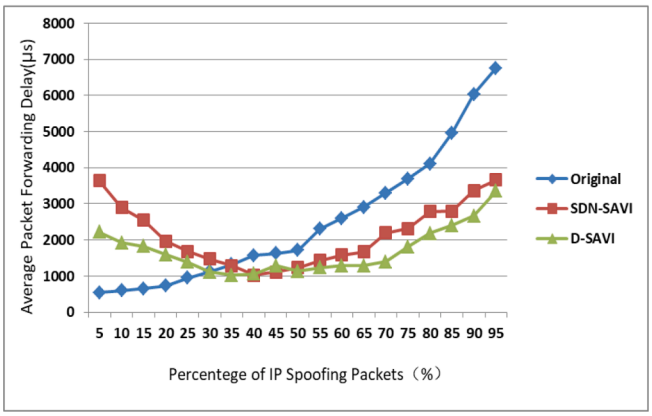


Fig. 18. Average packet forwarding delay in different IP spoofing conditions.

security state by analyzing the software-defined security to guarantee its integrity and availability. To test the capability of providing a security guarantee, we made use of SDN's global view to locate the node that launched an attack and coordinate with the controller to update the flow table of the switch to mitigate the impact of the spoofing attack or elephant flows. Fig. 19 illustrates that the number of spoofing packets decreases dramatically under the protection of D-SAVI.

We also monitored the CPU utilization of the Process ID (PID) associated with the instance of the running controller. We launched multiple address spoofing attacks and increased the request rate during the attack. As Fig. 20 depicts, we tested the percentage of candidates of the *Normal Set* and CPU utilization of D-SAVI under attack Type E (80% spoofing packets) with different *N* numbers. The vertical axis is the percentage of detected candidates in the *Normal Set* and occupancy rate of CPU utilization. The horizontal axis is the *N* number. The percentage of detected candidates in the *Normal Set* decreased in tandem with the growing *N* number, and the CPU utilization increased in tandem with the growing *N* number. In our simulated multi-layer network, a 8–10 is a reasonable *N* value, which balance the percentage of candidates and CPU utilization.

Similarly, we tested the random selection percentage in the *Normal Set*, which is an effective method of tracking the candidates in the binding relationships against the potential risk that a candidate is in spoofing danger. Conversely, we tested the delay of anomaly detection in D-SAVI, which symbolizes the interval between an illegal host sending a forged packet until the system checks and filters it. Fig. 21 shows the results of the simulation, when the percentage reaches around 15%, D-SAVI could find the right balance between the delay of anomaly detection and CPU utilization. Regarding the security guarantee provided by D-SAVI, we tested the anomaly condition detection rate against different percentages of host launched attacks (Fig. 22). D-SAVI could detect at

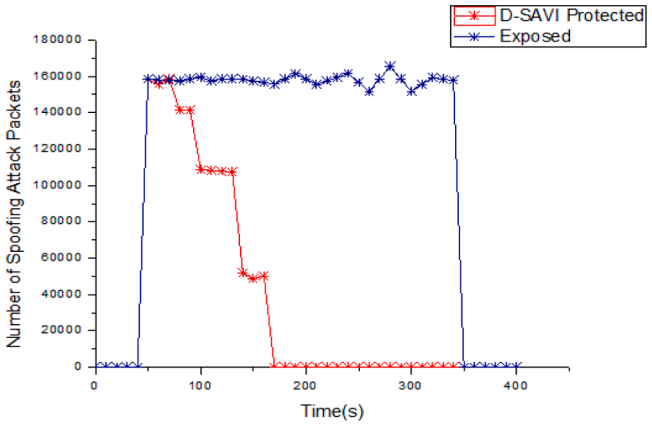


Fig. 19. Number of spoofing attack packets under the protection by D-SAVI.

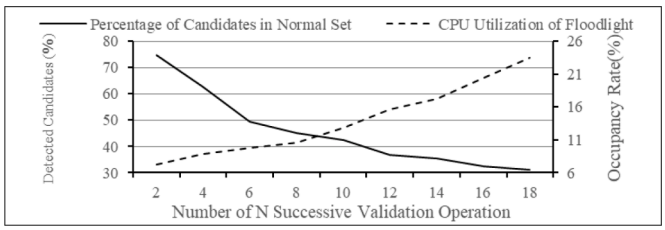


Fig. 20. Percentage of candidates in the normal set and CPU utilization of D-SAVI under attack type E.

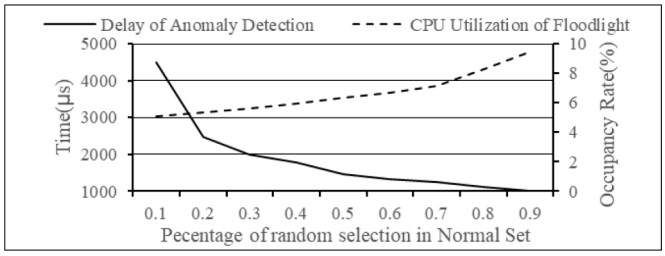


Fig. 21. Delay of anomaly detection and CPU utilization of D-SAVI under attack type E.

least 97% of the suspected anomaly candidates when several of the hosts in the SDN-based network launched spoofing attacks at a peak occupancy rate with a CPU utilization of 15% or less, which is the desired value in large-scale SDN-based networks [3]. In general, the average detection rate of D-SAVI reached 97.5%. As aforementioned the limited percentage of spoofing packets has a negligible influence on the performance of OpenFlow switches. Considering that D-SAVI significant optimizes in network performance compared with existing methods, the security guarantee of D-SAVI can be trusted.

6. Conclusion

In this paper, we proposed D-SAVI, which combines the requirements of optimized efficiency during source address validation and the quantity reduction in verification number when facing anomalous network states in a large-scale SDN-based environment. Our extensive research into solving source address validation based on SDN shows that D-SAVI has the characteristics of efficiency and lower resource consumption. The experimental results show that, the network performance of D-SAVI is superior to other methods, and it provides a satisfactory safety guarantee. Our future work will mainly focus on experimenting on an actual SDN-based data center network and thereby, further optimizing the performance of source address validation.

CRediT authorship contribution statement

Qizhao Zhou: Conceptualization, Methodology, Software, Formal analysis, Data curtion, Writing – original draft, Writing – review & editing. **Junqing Yu:** Resources, Writing – review & editing,

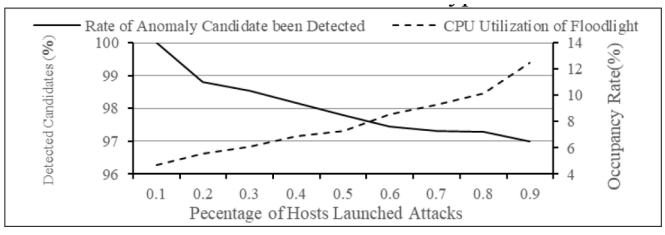


Fig. 22. Rate of anomaly detection and CPU utilization of D-SAVI under attack type E.

Supervision, Project administration, Funding acquisition. **Dong Li:** Validation, Methodology, Investigation, Visualization.

Declaration of Competing Interest

None

Acknowledgement

This work is supported by the National Network Security Key Research and Development Program of China under Grant No. 2017YFB0801703 and the CERNET Innovation Project under Grant No. NGII20170408.

References

[1] J. Kwon, D. Seo, M. Kwon, et al., An incrementally deployable anti-spoofing mechanism for software-defined networks, Comput. Commun. 64 (C) (2015) 1–20.
[2] G. Yao, J. Bi, A.V. Vasilakos, I.P. Passive, traceback: disclosing the locations of IP spoofers from path backscatter, IEEE Trans. Inf. Forensics Secur. 10 (3) (2015) 471–484.
[3] M. Lavanya, P.K. Sahoo, IP spoofing and its detection technique, Int. J. Adv. Comput. Techn. Appl. 4 (1) (2016) 167–169.
[4] Zhang C, Hu G, Chen G, Sangaiah A K, Zhang P, Yan X, Jiang W. Towards a SDN-Based Integrated Architecture for Mitigating IP Spoofing Attack. IEEE Access, 6: 22764-22777.
[5] A.R. Curtis, W. Kim, Y.P. Mahout, Low-overhead datacenter traffic management using end-host-based elephant detection, in: IEEE Infocom., IEEE, 2011.
[6] Wu, J. et al., 'Source address validation improvement (SAVI) framework,' Tech. Rep.:2013.
[7] J. Bi, J. Wu, G. Yao, et al., Source address validation improvement (SAVI) solution for DHCP, in: International Conference on Information Integration and Web-Based Applications & Services, ACM, NY, 2015, 77-77.
[8] R. Cziva, S. Jouët, D. Stapleton, et al., SDN-based virtual machine management for cloud data centers, IEEE Trans. Netw. Serv. Manage. 13 (2) (2014) 212–225.
[9] D. Adami, B. Martini, A. Sgambelluri, et al., An SDN orchestrator for cloud data center: system design and experimental evaluation, Trans. Emerg. Telecommun. Technol. 28 (11) (2017) e3172.
[10] N. Mckeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, J. Turner, OpenFlow: enabling innovation in campus networks, ACM Sigcomm. Comput. Commun. Rev. 38 (2) (2008) 69–74.
[11] B. Liu, J. Bi, Y. Zhou, Source address validation in software defined networks, in: 2016 ACM SIGCOMM Conference, ACM, NewYork, 2016.
[12] C. Li, et al., SDN-Ti: a general solution based on SDN to attacker traceback and identification in IPv6 networks, in: ICC 2019-2019 IEEE International Conference on Communications (ICC), IEEE, 2019.
[13] Chen, G., Hu, G., Jiang, Y., Zhang, C., 2016 SAVSH: IP source address validation for SDN hybrid networks.
[14] G. Yao, J. Bi, P. Xiao, VASE: filtering IP spoofing traffic with agility, Comput. Netw. 57 (1) (2013) 243–257.
[15] J. Li, J. Bi, J. Wu, Towards a cooperative mechanism based distributed source address filtering, in: International Conference on Computer Communications and Networks, IEEE, Nassau, 2013, pp. 1–7.
[16] B. Liu, J. Bi, A.V. Vasilakos, Toward incentivizing anti-spoofing deployment, IEEE Trans. Inf. Forensics Secur. 9 (3) (2014) 436–450.
[17] J. Li, J. Mirkovic, M. Wang, et al., SAVE: source address validity enforcement protocol, in: Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, New York, 2002, pp. 1557–1566.
[18] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, J. Zhu Mi. Zolla, U. Hölzle, S. Stuart, A. Vahdat, B4: experience with a globally-deployed software defined wan, in: Proceedings of the ACM SIGCOMM 2013, SIGCOMM '13, ACM, New York, 2013, pp. 3–14.
[19] L. Velasco, A. Asensio, J.L. Berral, et al., Towards a carrier SDN: an example for elastic inter-datacenter connectivity, Opt. Express 22 (1) (2014) 55–61.
[20] N.I. Mowla, I. Doh, K.J. Chae, An efficient defense mechanism for spoofed IP attack in SDN based CDNi, in: International Conference on Information Networking (ICOIN), IEEE, Cambodia, 2015, pp. 92–97.
[21] J. Bi, J. Wu, G. Yao, F. Baker, Source address validation improvement (SAVI) solution for DHCP, RFC (2015) 7513, https://doi.org/10.17487/RFC7513. MayAvailable: https://www.rfc-editor.org/info/rfc7513.
[22] G. Liu, T. Wood, Cloud-scale application performance monitoring with SDN and NFV, in: IEEE International Conference on Cloud Engineering, 2015.
[23] Ambrosin M, Conti M, Gaspari F D, et al. LineSwitch: Tackling control plane saturation attacks in software-defined networking. IEEE/ACM Transactions on Networking, 25(2):1206-1219.
[24] S.F. Papagalou, Application of anomaly detection algorithms for detecting SYN flooding attacks, Comput. Commun. 29(9) (2006) 1433–1442.
[25] Y. Xu, Y. Liu, DDoS attack detection under SDN context, in: IEEE Infocom 2016 - IEEE Conference on Computer Communications, IEEE, 2016.
[26] S. Behal, K. Kumar, Detection of DDoS attacks and flash events using novel information theory metrics, Comput. Netw. 116 (2017) 96–110.
[27] R. Li, B. Wu, Early detection of DDoS based on φ-entropy in SDN networks, in: IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), IEEE, 2020.
[28] M. Semerci, A.T. Cemgil, B. Sankur, An intelligent cyber security system against DDoS attacks in SIP networks, Comput. Netw. 136 (2018) 137–154.
[29] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, V. Maglaris, Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments, Comput. Netw. 62 (2014) 122–136.
[30] D. Li, C. Yu, Q. Zhou, J. Yu, Using SVM to detect DDoS attack in SDN network, in: IOP Conference Series: Materials Science and Engineering, 2018, p. 466.
[31] T. Chen, C. Guestrin, XGBoost: a scalable tree boosting system, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2016.
[32] Chen, T., Tong, H., Benesty, M., et al. Xgboost: extreme gradient boosting. 2016.
[33] M.S. Elsayed, N.A. Le-Khac, A.D. Jurcut, InSDN: A novel SDN intrusion dataset. IEEE Access (2020).
[34] E. Jo, D. Pan, J. Liu, et al., A simulation and emulation study of SDN-based multipath routing for fat-tree data center networks, in: Proceedings of the Winter Simulation Conference, IEEE, Savanah, 2014, pp. 3072–3083.
[35] B. Pfaff, J. Pettit, T. Koponen, et al., The design and implementation of Open vSwitch, Login 40 (2015) 12–16.
[36] Mei, A., Stefa, J., Swim, S.J. A simple model to generate small mobile worlds, 2009.
[37] Tools Yard. Hacking tools and penetration testing tools, Jun. 2012. Online, Available: http://toolsyard.thehackernews.com/2012/06/bonesi-ddos-botnetsimulator.html.
[38] S. Scholly, Akamai releases prolexic Q2 2014 global DDoS attack report, Database Netw. J. (2014).
[39] Y.C. Wu, H.R. Tseng, W. Yang, et al., DDoS detection and traceback with decision tree and grey relational. Analysis[C], in: International Conference on Multimedia & Ubiquitous Engineering, IEEE, 2009.
[40] D.L. Belgiu, Random forest in remote sensing: a review of applications and future directions, ISPRS J. Photogramm. Remote Sens. 114 (2016) 24–31 (-).
[41] Shichao, et al., Efficient kNN classification with different numbers of nearest neighbors, IEEE Trans. Neural Netw. Learn. Syst. (2017).
[42] S.C. Chu, T.K. Dao, J.S. Pan, et al., Identifying correctness data scheme for aggregating data in cluster heads of wireless sensor network based on naive Bayes classification, EURASIP J. Wirel. Commun. Netw. 2020 (1) (2020).
[43] H. Wang, J. Gu, S. Wang, An effective intrusion detection framework based on SVM with feature augmentation, Knowl. Based Syst. 136 (Nov.15) (2017) 130–139.

Qizhao Zhou received the B.S. degree from the School of Information and Communication Engineering at the Beijing University of Posts and Telecommunications in 2013. Then he graduated M.S. degree from the School of Information and Communication Engineering at the Beijing University of Posts and Telecommunications in 2016 and currently he is a Ph.D. student in Computer Science & Technology at the Huazhong University of Science and Technology. His current research areas include software defined network, network security, and data mining.

Junqing Yu has been working as Professor in the School of Computer Science & Technology at the Huazhong University of Science and Technology. He has been also working as director of Center of Network and Computation at Huazhong University of Science and Technology. He has authored more than hundred papers in international conference proceedings and refereed journals and has been actively serving as a reviewer for international journals and conferences. His current research areas include computer networks and network security, cybersecurity, Digital media processing and retrieval, multi-core computing and stream compilation.

Dong Li has been working as research fellow of Center of Network and Computation at Huazhong University of Science and Technology. He has authored Dozens of papers in international conference proceedings and refereed journals. His current research areas include computer networks and network security, technology of cloud datacenter and IPv6 Protocol.