SSG - A Solution to Prevent Saturation Attack on the Data Plane and Control Plane in SDN/Openflow Networks

Dang Van Tuyen[1,2], Truong Thu Huong[1]
[1] Hanoi University of Science and Technology, Hanoi, Vietnam
[2] People's Police University of Technology and Logistics, Ministry of Public Security, Bac Ninh, Vietnam
Correspondence: Truong Thu Huong, huong.truongthu@hust.edu.vn
Communication: received 3 November 2018, revised 25 January 2019, accepted 31 January 2019
Online early access: 28 February 2019, Digital Object Identifier: 10.32913/mic-ict-research.v2019.n1.833
The Area Editor coordinating the review of this article and deciding to accept it was Dr. Truong Trung Kien

Abstract: The SDN/Openflow architecture opens new opportunities for effective solutions to address network security problems; however, it also brings new security challenges compared to the traditional network architectures. One of these challenges is that the mechanism of reactive installation for new flow entries can make the data plane and control plane easily become a target for resource saturation attacks with spoofing techniques such as SYN flood. There are a number of solutions to this problem, e.g., the Connection Migration (CM) mechanism in Avant-Guard solution. Nevertheless, most of the solutions increase the load at commodity switches and/or split benign TCP connections, which can increase the packet latency and disable some features of the TCP protocol. This paper presents a solution, referred to as SDN-based SYN Flood Guard (SSG), which takes advantages of the OpenFlow's abilities to match TCP Flags fields and the RST Cookie technique to authenticate the three-way handshake process of TCP connections in a separated device from SDN/Openflow switches. Experiment results reveal that SSG solves the aforementioned problems and improves the SYN Flood attack tolerance compared to the existing solutions.

Keywords: *OpenFlow, SDN, DDoS attack, saturation attack, TCP SYN flood, SYN cookie, RST cookie.*

I. Introduction

Recently, Software Defined Networking (SDN) [1] has attracted interest in academia and industry as a promising network technology. The philosophy of separating control plane from data plane allows SDN researchers and operators to easily add new, creative, powerful network functions and protocols in order to optimize performance as well as reduce operational cost.

In an SDN network, the switches only handle the packet forwarding function based on the configuration by a controller. The network-control policies such as routing calculation, load balancing, and security rules are implemented by the controller and configuration information in the forwarding tables of the switches via a security interface.

OpenFlow [2, 3] is generally regarded as an SDN reference implementation that defines the structure of forwarding tables at the SDN switches and the messages exchanged between the switches and controllers. At an OpenFlow switch (OFS), network policies are specified into flow entries arranged in flow tables. In a flow entry (FE), there are criterion fields which will be matched with incoming packets for finding the corresponding appropriate policies. If an incoming packet is not matched with any existing flow rules, a *table-miss* event will be issued with a *packet_in* message sent to the controller to install a new FE for processing the packet. The FE will be kept in flow tables of the OFS for a period of time based on *idle-timeout* and *hard-timeout* settings or eviction commands by application running on controllers. With this mechanism of creating and maintaining flow rules, a *table-miss* consumes a specific amount of resources (includes of CPU, memory and bandwidth) in both data plane and control plane.

Exploiting this characteristic, attackers can implement Distributed Denial of Service (DDoS) attacks in order to flood, saturate OFSes and controllers by making a massive series of *table-miss* events and useless FEs. This may be easily carried out by sending packets with randomly forging some or all packet-header fields making them hard to match any existing FEs at a victim switch. If there are enough

new flows, creating a flood of table-miss events in a short time, the resource on data plane and/or control plane will be quickly exhausted.

SYN Flood attacks have been known early in the Internet development and they remain a common form of DoS and DDoS attack [4]. Attackers often spoof the source IP addresses in SYN packets, making it difficult to detect and mitigate this kind of attacks. In the SDN/OpenFlow context, by forging randomly source IP addresses to create flood of *table-miss* events and useless FEs, SYN floods can be exploited to attack OFSes (data plane) and controllers (control plane) [5–7].

Plenty of research has been performed to address the saturation of the data and control plane. However, most focus on collecting traffic information in the data plane and then applying analysis algorithms at the control plane either to produce packet policies or to enhance security processing ability in the data plane. One drawback of these techniques is that they increase in OpenFlow traffic. To address SYN Flood attacks, in Avant-Guard solution [8], S. Shin *et al.* integrate into OFSes the Connection Migration (CM) mechanism which applies SYN Cookie technique [9] to prevent Source-IP-spoofed SYN packets from requesting controller to install FEs.

CM mechanism makes OFS work as an SYN proxy. Upon receiving an SYN request from an Internet client, instead of forwarding the packet to the internal server, OFS handshakes with the client. After ensuring the connection to the client is successfully based on authenticating the three-way handshake (3HS) process, the OFS requests controller for a new FE to set up and maintain a TCP connection to the real internal server.

Although the CM mechanism prohibits attacked SYN packets from entering servers and decreases useless flow rules, it causes some problems [10] including: (1) splitting benign connections between Internet clients and internal servers, which consumes more time to process packets, disables some end-to-end functionalities of TCP protocol, and also requires the switch to maintain state information for each TCP connection; (2) an increase the processing load at the OFS, which makes the switch more vulnerable to the attack.

In this paper, we make two contributions. In the first one, we propose an SSG solution as a substitution for the CM mechanism in Avant-Guard. To protect the OFS from overloading, the SSG monitors the 3HS process of TCP connections on a separate security device. The OFS recognizes and transfers 3HS packets to the security device based on the ability to match TCP flag fields defined in OpenFlow 1.5 [3]. Moreover, to overcome the problem of splitting legitimate TCP connections, the SSG uses the RST Cookie technique to authenticate and verify the source IP address in SYN packets instead of SYN Cookie as in CM. These enhancements help the system solve the shortcomings of the CM mechanism.

In the second contribution, we build a test-bed and run a number of experiments to compare our solution against OpenFlow and the CM mechanism under SYN flood attack at different speeds. We also make an analysis on CAIDA dataset [11] to measure the increased traffic at OFS interfaces when a server operating with a real traffic is under SYN flood attacks. The analytical and experimental results show that under SYN flood attacks, the SSG improves the rate of successful connections, and decreases the average retrieve time with a lower resource consumption at both the OFS and the controller. The OpenFlow traffic between the OFS and the controller is also reduced with a small increase in the rate of traffic at the OFS interfaces. These improvements make the system more resistant under saturation attacks by SYN flooding.

The rest of this paper is organized as follows. Related works are introduced in Section II. Section III presents the principle of the CM mechanism of the Avant-Guard solution. Section IV describes our proposed network security architecture and the operating principles. The analysis of the proposed solution and experiment results are elaborated in Section V. Finally, conclusions are presented in Section VI.

II. Related Work

The SDN/OpenFlow architecture has attracted many research works including security issues [5–7, 12–18]. Some works analyze the SDN architecture and OpenFlow protocols in terms of security and identification of the threats of saturation attacks on data plane and control plane such as the study of Kreutz *et al.* [19], Kloti *et al.* [20] and Shin [16]. Solutions to mitigate this type of saturation attacks can be divided into three groups.

In the first solution group, processing functionalities are added to the controller. Traffic characteristics of flows passed across the system are collected and sent to the controller. The application running on the controller analyzes traffic characteristics and network topology to detect anomalies and through installing FEs into the OFS produce appropriate policies such as load balancing, dropping packets to mitigate attacks. The common drawback of these solutions is that the controller must collect sufficient information on each new flow on the entire network using statistical query messages and therefore the network traffic increases dramatically when an attack occurs. Some proposed solutions such as ONIX [21], HyperFlow [22], and ONOS [23] use logically centralized but physically distributed controllers to increase the responsiveness for

requests from the OFSs to the controllers. In [24], Anilkumar proposed SmartTime, an OpenFlow controller system that combines an adaptive timeout heuristic to compute efficient idle timeouts with the proactive eviction of flow rules, to utilize effectively TCAM space in the OFSs. The SDN-Guard solution proposed by Lobna *et al.* [25] combines dynamically rerouting potential malicious traffic, adjusting flow timeouts and aggregating flow rules while in FloodGuard [26], Wang *et al.* introduced two techniques: proactive flow rule analyzer and packet migration to reduce both the number of FEs on the OFSs and communication between the data and control planes.

The second solution group deals with the problem by enhancing the functionality of switches. A typical function of OpenFlow 1.3 [2] is Rate Limit that restricts the rate packets processed at the switches. The DEFANE solution [27] proposed by Minlan et. al. adds topology discovery and FE distribution that allow the OFSs to easily enforce network access policies. Cutis et. al. proposed DevoFlow [28] that allows to detect elephant flows and to create and install a new FE by copying the same actions without refering to the controller. In [29], Mekky *et al.* proposed a solution to limit the communication between the data plane and the control plane by allowing packet forwarding based on application level information. By the same way, in [30], Kotani and Okabe reduced the data-to-control plane communication by adding two data structures for low-priority packets, Pending Rule Table (PRT) and Pending Flow Table (PFT). The packet are matched with FEs in the flow table and in the PRT and PFT. Only packets that are not matched with any FEs in the flow table can create *packet_in* events and be sent to the controller. However, this defense mechanism can be avoided by flooding attack packets with header modifications such as spoofed source IP fields. In such cases, the PRT and PFT filters will be ineffective and all the packets will be forwarded to the controller. The common downside of these solutions is that they require structure modification and higher capacity at the OFSs for storing additional detailed information and processing actions. With these modifications, OFSs must also have additional packet processing functionalities. Attackers can exploit this feature to increase the processing task at the switches.

The third solution group implements the packet handling rules to eliminate and prevent resource starvation attacks based on communication protocols. In Avant-Guard [8], S. Shin uses the SYN Cookie technique [9] to build the CM mechanism that prevents and minimizes impacts of SYN Flood attacks. As a result, the switch will censor all TCP connections and use SYN Cookies to authenticate the 3HS process. The OFS requires the controller to install FEs only for those connections that complete the 3HS process. Ambrosin introduced an improvement to the LineSwitch solution [10] in order to reduce the load of the OFS by ignoring the authentication of the 3HS process for SYN packets coming from trusted sources then sending a packet-processing request to the controller immediately at a given probability p. LineSwitch also immediately removes the incoming SYN packets if their source addresses are from untrusted sources (in the blacklist). Although LineSwitch reduces the load at the OFS and the connection processing time between the client and the server, the solution requires the switch to manage a large number of IP addresses for both trusted and untrusted source lists, so the search time in these lists will also increase. When a system is attacked by an SYN Flood with IP address spoofing, the number of IP addresses in the untrusted source list increases dramatically. Solutions in this group only address one type of attack. Thus far, the proposed solutions have only focused on SYN Flood attacks. Due to the popularity and influence of this type of attack [4, 31, 32], we study and propose the SSG solution based on the idea of Avant-Guard's CM mechanism. LineSwitch techniques can be fully applied to the SSG without compromising the performance of the OFS. To easily compare the performance and improvement, in Section III, the working mechanism and the shortcomings of the CM mechanism in Avant-Guard are summarized.

III. Introduction of Connection Migration Mechanism

The main idea of Avant-Guard is to introduce the CM mechanism into the OpenFlow architecture to ensure the legitimacy of an SYN-packet source by monitoring the 3HS process of TCP connections. For the TCP connections that complete the 3HS process, the OFS sends a *packet_in* message to the controller to request a new FE. The CM mechanism is integrated into the OFSs and using the SYN Cookie technique [9] with four phases as shown in Figure 1.

a) Classification Phase:

When the OFS receives an SYN packet (step 1 in Figure 1), it directly replies the client an SYN-ACK packet using the SYN Cookie [9] technique (step 2). Depending on whether an ACK packet is received from the client (CliACK) (step 3), the SYN Cookie technique checks the 3HS process and determines whether the SYN packet has a forged Source IP Address or not.

The SYN Cookie technique [9] was invented by Bernstein to combat SYN Flood attacks, where a server can verify a client SYN request without storing and managing connection status information in Transmission Control Blocks (TCB) while waiting for a CliACK packet replied from the client. When an SYN packet is received, the server generates a 32-bit cookie value, which is encrypted

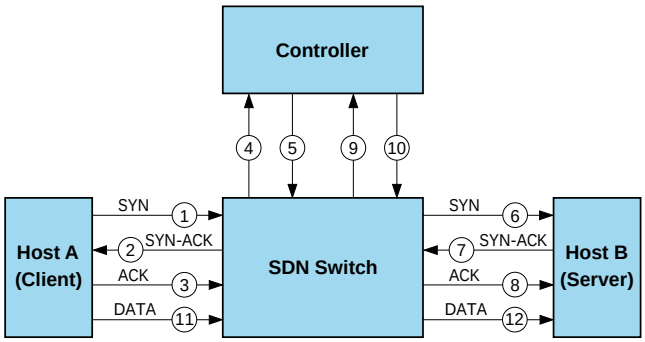


Figure 1. Avant-Guard connection migration mechanism.

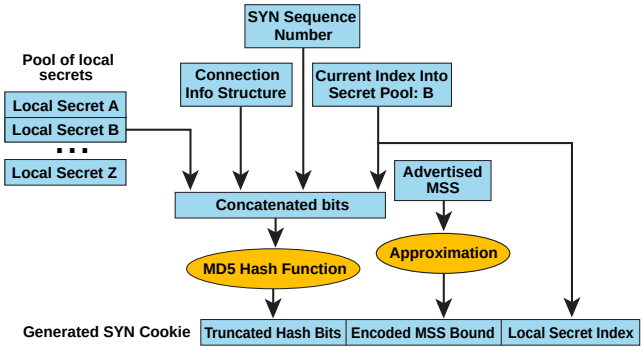


Figure 2. Cookie encryption process in SYN Cookie technique.

by the TCP connection parameters (including the source and destination IP addresses, source and destination ports, the initial *Sequence Number* of the SYN packet) and the server's secret parameters. This cookie value is used as the *Sequence Number* of the replied SYN-ACK packet (Figure 2). Upon receiving the acknowledgement packet CliACK from the client, the server encrypts cookie again and compares its value to the *Acknowledgment Number* in the received CliACK. If these two values match, the CliACK packet is validated and the 3HS process is assumed to be successful.

In Avant Guard, the CM applies the SYN Cookie technique directly into an OFS to authenticate the source IP address of an SYN packet by directly handshaking with the client. This allows the system to completely prevent SYN attack packets that spoof IP addresses from being forwarded to internal application servers without having to store the states of TCP connections in the OFS. Only authenticated SYN packets that complete the 3HS process will be passed to the Report phase.

b) Report Phase:

If the 3HS process of the SYN packet is completed, the OFS sends a *packet_in* message to the controller (step 4, Figure 1) to request to install a new FE that will be in charge of processing subsequent packets from the authenticated TCP connection from the client (step 5).

c) Migration Phase:

Upon receiving the controller's consent, the OFS establishes a TCP connection to the server (steps 6, 7 and 8 in Figure 1). Similar to the connection to the client, after the 3HS process with the server is successfully completed, the OFS also sends a *packet_in* message to the controller to request an FE for processing incoming packets of the connection from the server side (steps 9 and 10).

d) Relay Phase:

When both handshakes with the two sides are successful, the OFS maintains a communication session for exchanging packets between the client and the server by two FEs that have been created during the report phase and the migration phase.

A prominent advantage of the CM mechanism is that for only TCP connections that have completed the 3HS process, the OFS sends *packet_in* messages to the controller and the corresponding FEs are installed. SYN packets with forged source IP address will not complete the 3HS, hence their impact on the switch (data plane) and also on the controller (control planes) is mitigated. Moreover, the SYN Flood attack traffic is also prevented going to servers inside. Further more, applying the SYN Cookie technique helps the OFS reduce memory usage for TCBs especially when an SYN Flooding attack occurs.

However, the CM mechanism has three main disadvantages, as described next. First, CM uses the SYN Cookie, which is a host-based solution against SYN flood attacks applied directly into a server [33], which is a terminal entity of TCP connections. This approach is not suitable for deployment on intermediate devices such as OFSs because each benign TCP connection will then be split into two parts: one between the switch and the client while the other between the switch and the server. To link these two partial connections, the switch has to spend memory and computing resources for managing and maintaining state information as well as exchanging the *SEQ_Num/ACK_Num* value pair in packets for each TCP connection. After these value fields change, the packet's checksum must be calculated and updated from Layer 4 down to Layer 1 [34]. This reduces the system performance, increasing packet processing latency for benign TCP connections.

Second, the splitting of TCP connections between clients and servers due to the SYN Cookie also disables some optional features of the TCP protocol such as Maximum Segment Size (MSS), Selective ACKs, or TCP Window Scaling [34], etc during the 3HS process.

Third, the process of generating SYN-ACK packets, validating CliACK packets in the SYN-Cookie technique requires a number of certain computational resources including creating a packet, encrypting and generating cookies, calculating checksums, and so on. The CM mechanism performs the 3HS process by the SYN Cookie in the OFS for all incoming SYN packets even if the destination port on the server is closed. This wastes computing resources on the switch, making the switch vulnerable and easily become a target of SYN flooding and scan port attacks. In fact, integrating the additional hardware - CM module into OFS loses the original nature of an SDN switch that was designed to function as forwarding plane only. Therefore, it will be difficult to apply this type of solution to commercial OFSs that are designed as the original OFS (i.e. working in the data plane only).

As an enhanced solution of Avant-Guard, Lineswitch [10] supplements algorithms in order to reduce the usage of connection proxy at the OFS, through managing lists of trusted and untrusted source IP addresses and comparing them with the addresses of incoming packets. However, Lineswitch still uses the CM mechanism in a similar manner to Avant-Guard so the above shortcomings still prevail.

IV. SSG - Proposed Solution to Prevent Saturation Attack using SYN Flood on the Data Plane and Control Plane

1. Research Scope and Methodology

SDN-based SYN Flood Guard (SSG) is proposed to protect the OFSs (data plane) and controllers (control planes) of a small to medium size data centers from saturation attacks using TCP SYN Flood. The internal application servers are supposed to be trusted, not infected or under control of malwares. The techniques discussed in this paper only work for TCP SYN flood, not for other attack types.

With the aim of overcoming the disadvantages of CM mentioned in Section III, the SSG solution, first, minimizes resource and memory consumption at the OFSs by moving the function of monitoring the 3HS process from the switches in CM to a separated security device that is connected reliably with the controller. This helps SSG increase attack tolerance, eliminate the risk of becoming a target for saturation SYN Flood attacks, and make the solution feasible for all OpenFlow-supported commercial switches.

Second, SSG does not change OpenFlow packet-processing procedures when the system is free of SYN Flood attacks, in order not to affect the connection time or the delay of TCP connections. To do this, SSG must be equipped with a mechanism to detect the attack for each internal server and change the corresponding packet handling policies when server's state changes.

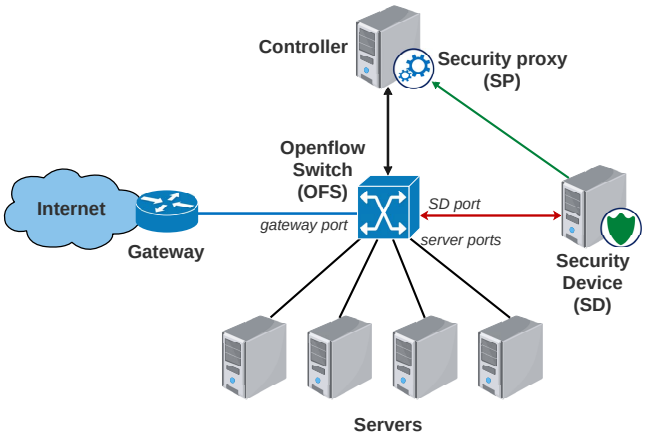


Figure 3. SSG system architecture.

Finally, SSG uses the RST Cookie technique to verify the source IP address of untrusted SYN packets instead of using the SYN Cookie in Avant-Guard. This allows direct TCP connections from the client end to the server end without being split, and therefore it is easy to discard SYN requests to closed ports on internal application servers. Not using the SYN Cookie technique at the intermediate device also enables optional features in the TCP protocol during the 3HS process.

2. System Architecture and Operation Principle

With the outlined methodology above, Figure 3 shows the SSG system architecture. Figure 4 describes the detailed structure of main components of the entities and the interactions between them. It is proposed to implement the SSG at the edge network of a data center. This architecture is inherited from our previous work [35] in which the Internet traffic is passed through a gateway to servers via an OFS.

The OFS takes charge of (i) capturing packets during the 3HS process of TCP connections and forward them to a Security Device (SD) for monitoring and (ii) maintaining FEs for exchanging data of each TCP connection after it accomplishes the 3HS process. Making use of the ability to match TCP Flags fields [3] and the packet processing pipeline, SSG arranges proactive and reactive FEs with suitable actions into flow tables in the OFS to handle these tasks. The OFS distinguishes three separate port types: gateway port to receive and process incoming and outgoing traffic from the Internet, SD port to connect to SD, and other normal ports (server ports) to connect to servers of the data center. This distinction is based on the information of the *in-port* matching field specified by the OpenFlow protocol and allows the system to distinguish the packets

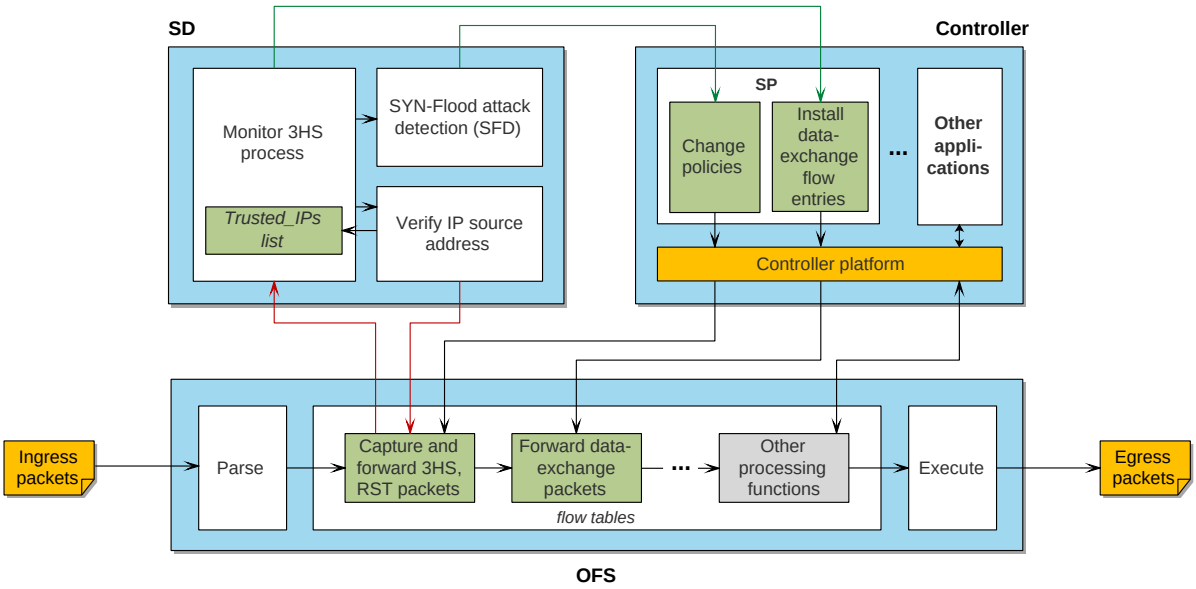


Figure 4. Detailed structure and interactions between main system components.

from clients, servers or SD and to have appropriate packet processing policies.

The SD is a stand-alone device which analyzes traffic forwarded from the OFS in order to: (i) detect SYN Flood attacks, (ii) monitor and authenticate the 3HS process of TCP connections between the outside clients and the inside servers, (iii) verify the source IP address of incoming SYN packets by using the RST Cookie technique when a server is under an SYN Flood attack, and (iv) request the controller to modify, install FEs for related packet processing rules. The SD uses packet processing tools such as DPDK [36] instead of relying on common network protocols at general terminals.

The Security Proxy (SP) is an application running on the controller, which receives the requests from the SD and interacts with the controller platform to (i) modify FEs for changing 3HS packet-forwarding policies when the attack state of a server changes and (ii) set up FEs for exchanging data between two ends of a TCP connection which is accomplished the 3HS process.

The SSG operation is based on the following principles. First, alll TCP connections from external clients must be controlled to ensure that their original source IP addresses are real and not spoofed by checking the 3HS process at the SD. The OFS just installs FEs for subsequent data-exchange packets of TCP connections of which the 3HS processes are accomplished.

Second, as an alternative to the CM mechanism, a 3HS process is carried out from the client end to the server end without being split. If there is no SYN Flood attack occurred, 3HS packets are sent to the SD for monitoring. Depending on the number of Half Open Connections (HOCs), the SD detects the presence of SYN Flood attack for each server and changes its state to either *"under SYN flood attack"* or *"attack free"*.

Third, when the system is under an SYN Flood attack, the SD verifies the source IP address of SYN packets before monitoring the 3HS process by using the RST Cookie technique. The monitoring process is only applied to the SYN packets with verified source IP addresses.

Finally, the change of processing rules for the 3HS packets depends on the state of the servers regarding to an attack. Whenever the state alters, the SD module will request the SP to modify the corresponding FEs for changing rules.

Details on packet processing at the OFS, the SD and the mechanism of the attack detection are described in the following subsections.

3. Forwarding TCP Packets at OFS

Using the ability to match packet headers with TCP flags specified in OpenFlow 1.5 [3], the ability to arrange flow entries with different priorities in multiple flow tables [2], in SSG, the OFS is set up and configured to match and capture the 3HS packets of TCP connections and route them to the servers or the SD based on the state of the internal destination server. If the state is "attack free", the packets are forwarded directly to the client and the server, and the copies are sent to the SD for monitoring the 3HS process. In case the destination server is under an SYN Flood attack, the SYN packet is sent to the SD for validating its source

TABLE I
ORGANIZATION OF FLOW TABLES AT THE OFS

Flow Table	Function	Flow entry type	When table miss occurs
FT1	Capture SYN packets from clients, SYN-ACK packets from servers and SD for directing and monitoring 3HS process, detecting SYN Flood attacks	Proactive (FEs exist permanently)	Direct to the flow table FT2
FT2	Direct data packets between clients and servers of TCP connections after 3HS completion	Reactive (FE exists until the TCP connection ends. The lifetime is based on idle-timeout and hard-timeout settings)	Direct to the flow table FT3
FT3	Capture CliACK, RST packets from clients, RST packet from servers for authenticating 3HS of a Half-Open Connection and processing the TCP connections	Proactive (FEs exist permanently)	Direct to the flow table FT4
FT4 and subsequent FTs	Contain FEs for other applications		

TABLE II
CHARACTERISTICS OF FLOW ENTRIES IN FLOW TABLE FT1 AND FT3 TO CAPTURE AND FORWARD 3HS PACKETS

Flow entry	In Flow table	Matched with packet	Packet appears on port	Actions
FE1x	FT1	SYN	gateway	– *If server is attack free:* pass to the corresponding server and a copy to SD – *If server is under SYN Flood attack:* pass to SD
FE2x	FT1	SYN	SD	– Pass to the corresponding server
FE3x	FT1	SYN-ACK	server	– Pass to the corresponding client through the gateway port – Create a copy and send to SD
FE4x	FT1	SYN-ACK	SD	– Pass to the client via gateway port
FE5	FT3	RST	gateway or server	– Pass to SD
FE6x	FT3	CliACK	gateway	– Pass to SD – *If server is attack free:* pass to the server

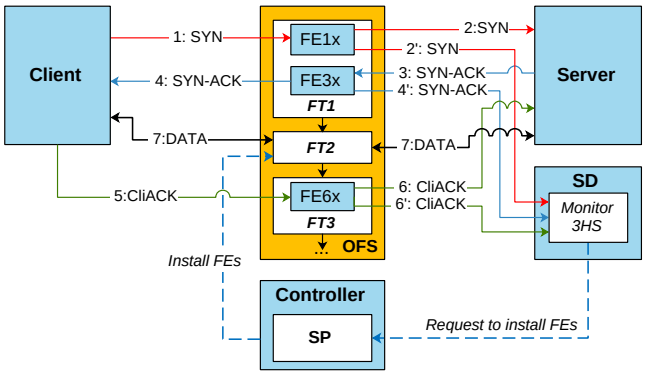


Figure 5. The process of capturing and directing packets of a legitimate TCP connection via FEs in OFS when the destination server is attack free.

IP address. Flow entries at the OFS are organized into flow tables which operate in the pipeline mechanism. The functions of the flow tables are shown in Table I. Table II describes the properties and actions of flow entries that capture and direct packets of the 3HS process.

Figure 5 is an example of the process of capturing and directing packets of a legitimate TCP connection via FEs at the OFS when the destination server is in the state of no SYN Flood attack.

4. Monitoring Three-way Handshake Process at SD

A normal 3HS process of a benign TCP connection must meet two requirements: (i) the client acknowledges the server by a CliACK packet upon receiving an SYN-ACK packet, and (ii) the *Acknowledgement Number (ACK_Num)* of the CliACK packet must be equal to the consecutive value of the *Sequence Number (SEQ_Num)* provided in the SYN-ACK packet. Monitoring the 3HS process is in order to identify and discard the SYN packets of which the source IP address is spoofed. The process of monitoring 3HS in SD is described in the flowchart of Figure 6. If there is no SYN Flood attack on the server, the monitoring process is applied to all incoming SYN packets. When the server is under a SYN Flood attack, the 3HS process of SYN packets from only trusted IP addresses are monitored. The trusted IP addressed are managed in a list named *Trusted_IPs* and

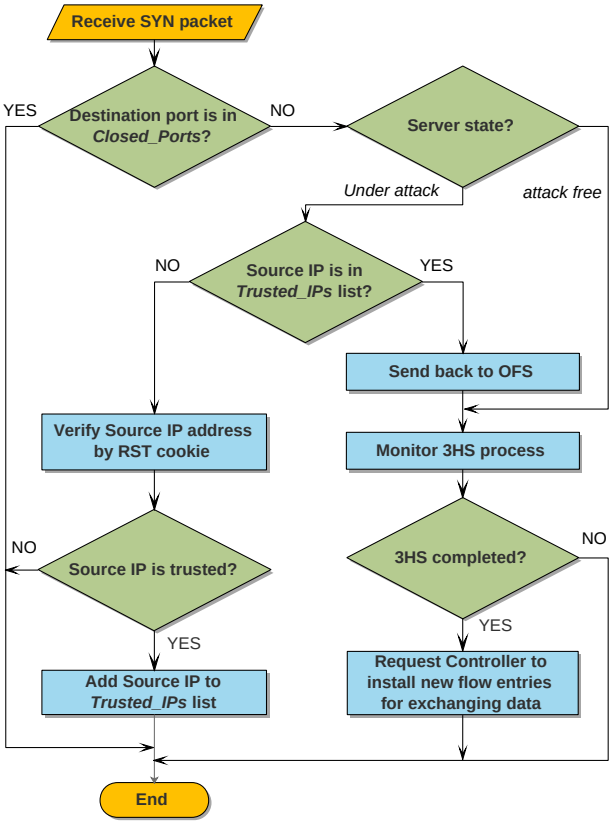


Figure 6. The flowchart of processing an incoming SYN packet in SD.

kept in the list for a period $T_{\text{IP-timeout}}$. SYN packets with the source address not in *Trusted_IPs* will be checked the source-address legitimacy by the RST cookie technique (the legitimacy check up will be described in subsection IV-E). If the source IP address is validated, it is added to the *Trusted_IPs* and the 3HS of the SYN request packet in the second time from the source will be monitored.

To limit port scanning attacks on the system, the SD also maintains a list *Closed_Ports* containing ports that have been recently closed in the servers. After receiving an SYN packet from a client, if the server returns a corresponding passive RST packet, the system understands that the port is currently closed. At this point, the port number is updated to the list and maintained for a period $T_{\text{port-timeout}}$. The SD discards the SYN packets that request to connect to destination ports on the server in the *Closed_Ports* list.

TCP connections are monitored for the 3HS process as described by the flowchart of Figure 7. To manage and monitor the 3HS, the SD maintains a list named *HOCs* containing half-open TCP connections. Each item in *HOCs* corresponds to a TCP connection and contains the information as shown in Figure 8. When the 3HS of a TCP connection is completed or the lifetime exceeds $T_{\text{HOC-timeout}}$, the corresponding item will be deleted.

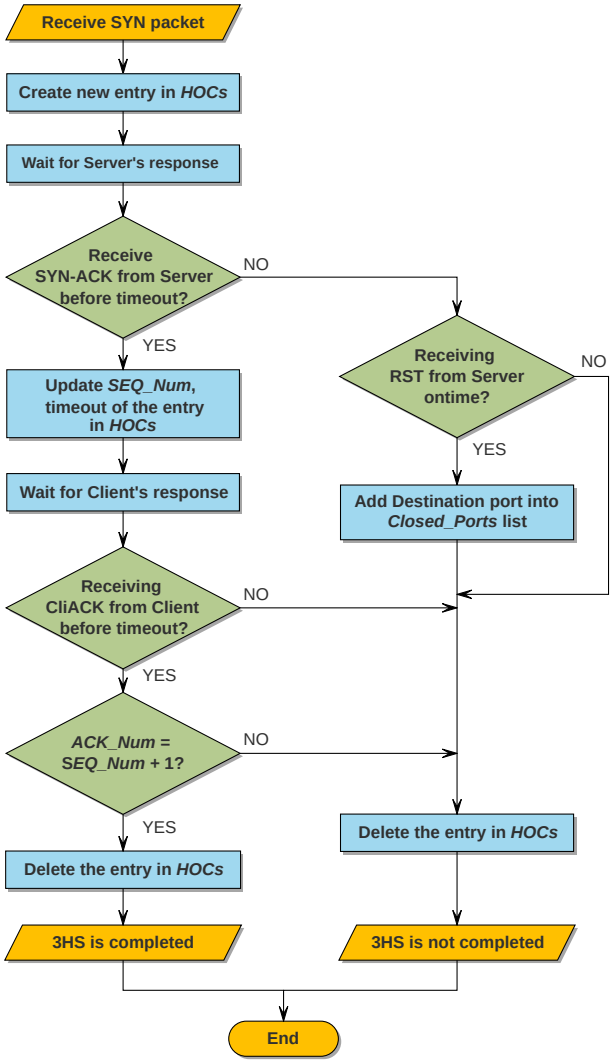


Figure 7. The flowchart of monitoring 3HS in SD.

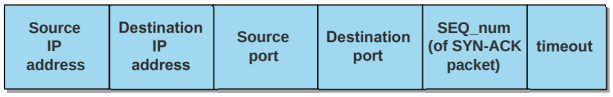


Figure 8. Structure of an item in *HOCs* list.

5. Verifying Legitimacy of SYN Packet Source IP Address

Verifying the source IP address of an SYN packet is the process to check if the address provided in the *Source IP Address* field is the real source which generates the packet. The RST Cookie technique is based on the principle of TCP reliable exchange between two entities (client and server) that each party initializes a 32-bit value for *SEQ_Num* and send to the other. A receiving packet from the opposite side is verified by the value of *ACK_Num*, which must be equal

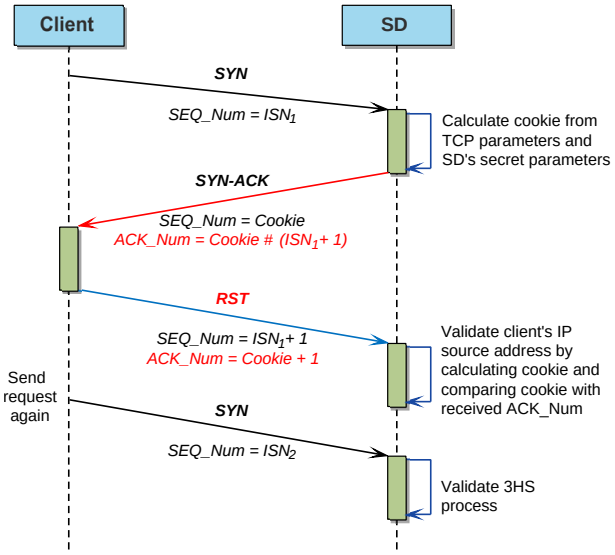


Figure 9. Sequence diagram of verifying the source IP address of an SYN packet using the RST cookie.

to the consecutive value of the sent *SEQ_Num*, that is,

$$ACK\_Num_{\text{receiving}} = SEQ\_Num_{\text{sent}} + 1. \quad (1)$$

If this rule is violated, the detecting side understands that an error has occurred and resets the connection. If this error is detected by the client, it will send an active RST packet and resend another SYN request.

Upon receiving an SYN request of which the *Source IP Address* is not in the *Trusted_IPs* list, the SD generates and replies the client with an SYN-ACK packet with the *ACK_Num* different from the consecutive value of the received *SEQ_Num* (Figure 9). At the same time, a cookie value generated by encrypting the TCP connection information and secret parameters is used as the initiated *SEQ_Num* for the direction from SD to the client. If the SYN packet is not spoofed, upon receiving this SYN-ACK, the client sends an RST with *ACK_Num* equal to the consecutive value of the cookie. The SD checks the validity of the RST packet by re-encrypting the cookie value and comparing with the received *ACK_Num*. If the values match, the SD understands that the SYN packet is generated from a real source and not being spoofed. In the case that SD does not receive a corresponding RST in time or the received *ACK_Num* is not correlated with the encrypted cookie, the field *Source IP Address* of the SYN packet has been forged for an SYN Flood attack.

6. Monitoring and Detecting SYN Flood Attacks at SD

To minimize the impact on the system performance and packet latency, SSG has different packet-handling policies for the cases of SYN Flood attack and attack free. The

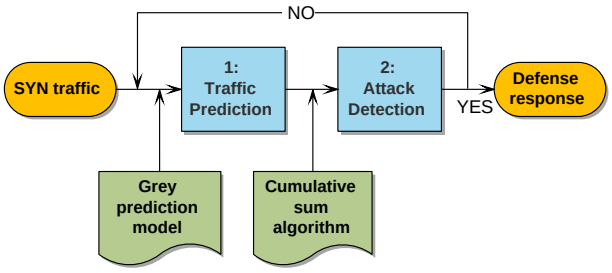


Figure 10. The model of SYN Flood detection used in SFD module.

SYN Flood attack Detection (SFD) module developed in SD detects the attack and asks the controller to distinguish these two states by using the corresponding packet handling policies. For the detection of the attack in SFD, the solution based on Grey prediction model [37–39] and Cumulative sum algorithm [40, 41] introduced by Wang *et al.* [42] is applied, Figure 10. When the system is under SYN Flood attack, the number of HOCs (k_{HOC}) increases quickly so this figure in each monitoring period T_{SFD} is chosen as the input parameter:

$$k_{\text{HOC}} = k_{\text{SYN}} - k_{\text{3HS-completed}}, \quad (2)$$

where k_{SYN} is the number of incoming SYN packets and $k_{\text{3HS-completed}}$ is the number of connections completing the 3HS process of a server in a monitoring period.

Depending on the classification result after each monitoring period T_{SFD}, when the attack-state of a server changes (from normal to attacked or vice versa), the SFD sends a message via the SP module to request the controller to modify corresponding flow entries FE1x, FE2x and FE6x for changing actions as described in Table II. With these modifications, subsequence SYN packets are processed by desired policies which are suitable for the current server state as described in subsection IV-3.

V. System Performance Evaluation

To compare the performance of the SSG with the CM mechanism of the Avant-Guard solution, we built a test-bed and implemented different SYN Flood attack scenarios by varying attack rates and analyzed the measured results. The test-bed is described in Figure 11.

The OFS is an OpenVswitch [43] version 2.7.90 soft switch built on a computer with five 1-Gbps ethernet cards, Intel Core$^{\text{TM}}$i3-2330M CPU @ 2.2 GHz, 500GB HDD, 2GB RAM.

The SD is a stand-alone PC running Ubuntu 14.04 with Intel Core$^{\text{TM}}$i3-2330M @ 2.2 GHz CPU, 500GB HDD, 2GB RAM. To adapt to SD functionalities, packets on the interface between the SD and the OFS are captured and

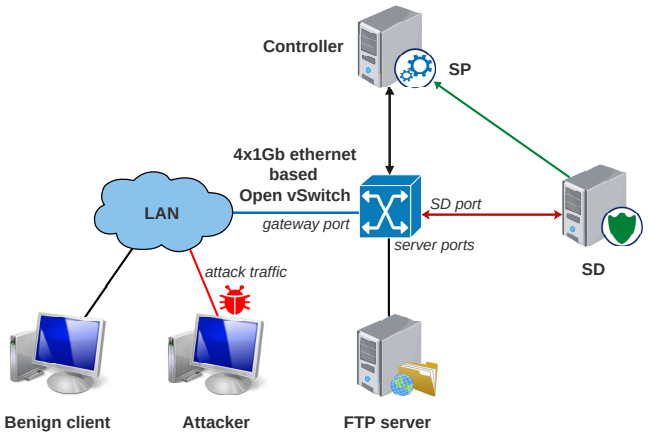


Figure 11. Structure of the testbed.

processed not according to the usual TCP protocol but a module developed by the DPDK tool [36].

The controller is based on the Floodlight scheme [44] installed on the Ubuntu 14.04 PC with the following hardware configurations: Intel Core^{TM}i3-2330M @ 2.2GHz CPU, 500GB HDD, 2GB RAM.

SYN Flood attack traffic is generated from the BONESI [45] tool with attack rates varying from 100 pps, 200 pps to 5500 pps. To ensure the homogeneity for different configurations, the generated attack traffic was logged using the WireShark [46] tool and regenerated with the TCPReplay [47] tool. The emulated attack time is 500 s for each attack rate. Legitimate traffic is generated from a software running on a computer that continually requests connections and downloads files of the FTP service on the server at a rate of 30 connections per second. The traffic between the benign client and the FTP server is recorded using the WireShark tool for statistical analysis.

Attack scenarios are performed in the same way for three schemes: the standard OpenFlow, the CM mechanism of Avant-Guard, and the SSG solution. Measured results and analysis are presented in the following subsections.

1. Successful Connection Rate and Retrieval Time when System is under SYN Flood Attacks

The goal of the SYN Flood TCP attack is to make the system less capable of serving benign TCP connections, in other words, to reduce the successful connection rate *(SCR)* and increase the retrieval time of benign TCP connections. *SCR* is measured by the number of TCP sockets that download a desired file successfully over the total TCP sockets sent from the benign clients. The average retrieval time *ART* is calculated as the average time of the successful TCP sockets from the time a benign client sends an SYN packet until it receives the first data packet from the FTP server.

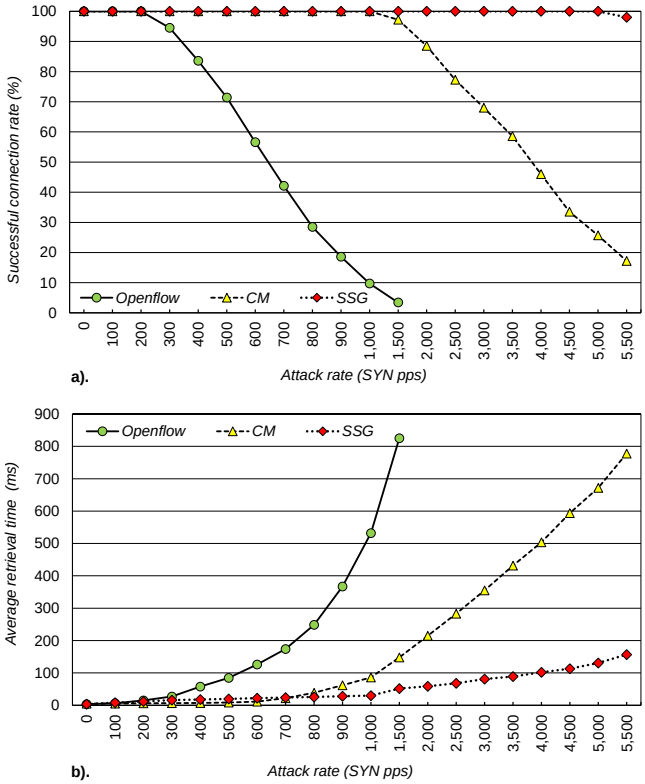


Figure 12. The *SCR* (a) and *ART* (b) of OpenFlow, CM and SSG schemes.

The measured parameters of the three schemes at different attack rates are presented in Figure 12. When the test-bed is attack free or under attack with low rate (below 200 pps), there is no considerable difference between the three schemes. In addition, under the increased attack rates, *SCR* of the OpenFlow scheme is quickly decreased to 3% at the rate of 1500 pps and the benign clients can not connect to the internal FTP server during the attack of 2000 SYN pps. The tolerance of CM mechanism is better as the *SCR* is maintained at 97% at a rate of 1500 pps and diminishes down to under 20% when the rate is 5500 pps.

Moreover, compared to the two other schemes, the connection ability to the server of the SSG solution is superior as the *SCR* remains stable at 100% and when the attack rate increases to 5500 pps, the rate just starts slightly decreasing to 98%, much higher than the figure of CM mechanism.

Finally, the results also point out that there is a correlation between the decline of *SCR* and the *ART* of TCP connections from benign clients to an internal application server for all of the three schemes. However, at low attack rates (from 100 pps to 700 pps), *ART* in SSG solution is a bit higher than that in the CM mechanism. The reason is that in SSG, a client must resend SYN requests when its IP address is not in the *Trusted IPs* list.

The testbed results show that the SSG solution significantly improves the system's attack tolerance over the OpenFlow and CM mechanism in Avant-Guard.

2. Resource Usage on the Switch and SD

Figure 13 presents the average amount of resource consumption in OFS and SD of the three schemes during states of attack free and under SYN flood attack. The resource consumption is defined using two parameters: the memory usage and the CPU utilization of OFS and SD processes on corresponding systems.

It can be seen that, when there is no attack, the memory occupancy and the CPU resource utilization of the SSG solution are approximately the same as the OpenFlow scheme with the respective values of over 150 MB and about 8%. Meanwhile, in the CM mechanism of the Avant-Guard solution, both these figures are higher, with values near 200 MB and 13%.

When the system is under SYN Flood attack, the resource consumption in the OFS of the OpenFlow scheme increases fastest and reaches nearly 1600 MB memory and over 50% CPU resource at an attack rate of 1500 pps. This is a demonstration of a saturation attack on the data plane, which occupies most of the switch's resource and causes the decrease in *SCR* as described in subsection V-A. The CM mechanism and the SSG solution help the OFS to reduce the impact of the attack and the resource occupancy rates just are under 300 MB of memory and about 30% of CPU at the same attack rate.

In comparison between the CM mechanism and the SSG solution, it can be seen that the resource usage in OFS of the SSG solution is always lower than of the CM mechanism. Especially, in the case of high attack rates (from 2500 pps to 5500 pps), the CM mechanism needs much more CPU resource than SSG which is the main reason for the *SCR* reduction presented in Section V-1.

The improvement in reducing resource usage, and packet processing capabilities of OFS in the SSG solution are due to transferring the function of 3HS monitoring from OFS to SD. The results in Figure 13 show that the resource consumption on the SD is not significantly affected by the increase in attack rate. Compared to the case of no attack traffic, at the highest attack rate in the testbed (5500 pps), the memory usage increases from 180 MB to 300 MB over 2 GB total memory and the CPU resource is increased from 12% to 23%. Choosing the right configuration for OFS processing capability, this combination helps the system increase its SYN Flood tolerance.

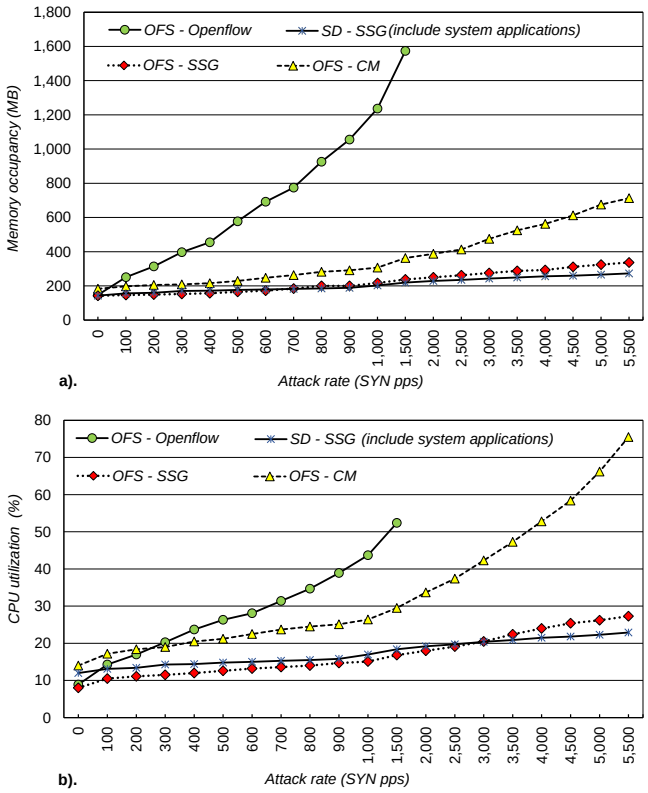


Figure 13. The memory occupancy (a) and CPU utilization (b) in OFS and SD.

3. Increase in Traffic Volume at OFS Interface

Compared to the CM scheme, moving the function of monitoring the 3HS process to SD increases the processed traffic at OFS interfaces. Here, the change in traffic at OFS interface is analyzed in both server states: free and under TCP SYN Flood.

a) When a server is not under SYN Flood attack:

By comparing the packet exchange between OFS and the other entities of SSG solution (Figure 5) and the CM mechanism (Figure 1), it can be seen that OFS in the CM scheme carries out 10 actions while in the SSG, the switch takes only 9 steps for the 3HS process of a benign TCP connection. Of these actions, 4 of CM and 1 of SSG are OpenFlow messages, the others are 3HS TCP packets. As defined in TCP protocols [34], 3HS packets usually are the smallest size TCP packets with the main information containing flag settings. Meanwhile, OpenFlow messages are TCP packets containing different information, so their size is much larger than 3HS packets. Therefore, taking account of the size of interaction packets, during the state of no attack, the traffic at the OFS interfaces of SSG solution is smaller than in the CM scheme.

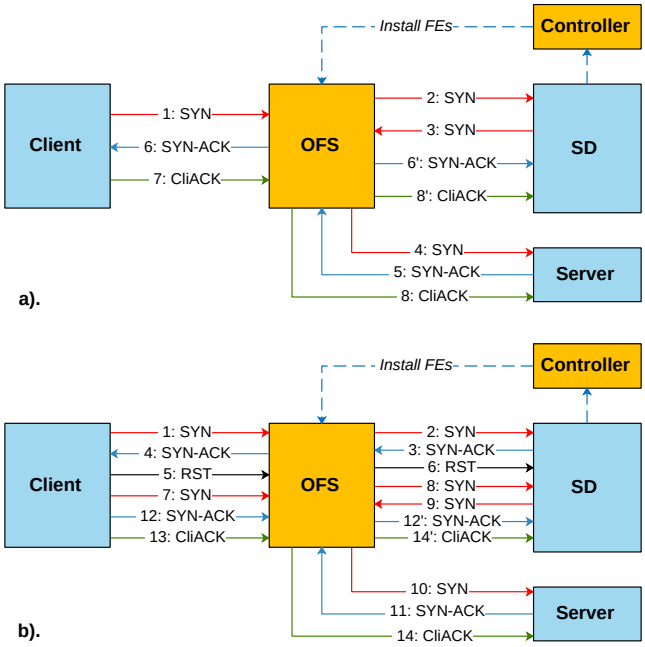


Figure 14. The interaction in SSG during 3HS process of a benign SYN packet with *Source IP address*: (a) in *Trusted_IPs* and (b) not in *Trusted_IPs* list

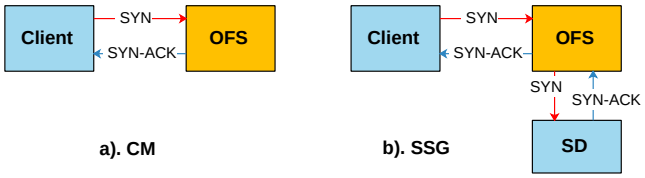


Figure 15. The interaction for processing an Source-IP spoofed SYN packet in: (a) CM and (b) SSG schemes.

TABLE III
STATISTICAL CHARACTERISTICS OF ATTACK-FREE TRAFFIC

Number of analyzed servers	100 servers
Analysis time	45 minutes
Number of TCP flows	3,947,883 flows
Number of new flows perserver per second	14.62 flows/server/s
Average total throughput from clients to each server	11,658,649.46 bytes/server
Average data rate from client to each server	5,181.62 bits/s/server
Average data rate in both ways per server	72,404,791.51 bits/s/server
Rate of SYN packets having the same source IP address with the previous packets ($T_{\text{IP-timeout}}$ = 5 minutes)	44,7 %

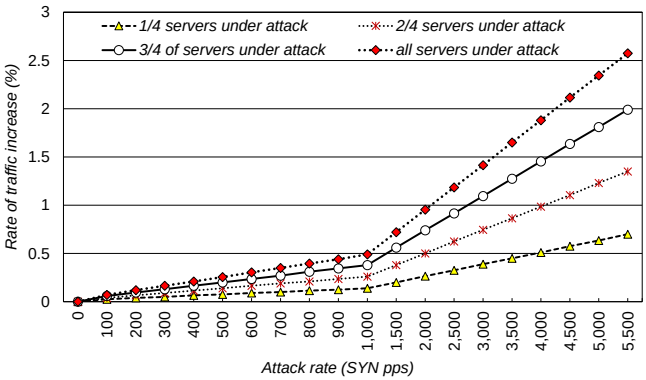


Figure 16. Traffic increase rate of SSG in comparison with the CM mechanism.

b) When a server is under SYN Flood attack:

Incoming SYN packets belong to one of three cases: (i) a benign packet with source IP address in the *Trusted_IPs* list, (ii) a benign packet with source IP address not in the *Trusted_IPs* list, and (iii) an attack packet with a spoofed source IP address. With packet processing mechanism in the SSG solution, OFS has to handle 9 actions in the 3HS process of a benign TCP connection for the first case (Figure 14.a) and 15 steps for the second case (Figure 14.b). In the CM mechanism, the interactions for 3HS process of SYN packets in the both cases are implemented in the same way through 10 steps as described in Figure 1. Concerning the third case, the number of OFS interaction steps in CM and in SSG are respectively 2 and 4 (Figure 15).

To evaluate the increase in traffic at OFS interfaces, the attack traffic is measured and analyzed in combination with attack-free traffic. For the attack-free traffic, logged packets from 100 randomly-selected application servers in the CAIDA 2013 dataset [11] are processed and analyzed. The statistical characteristics results of the traffic are described in Table III. The combined traffic is analyzed for four cases: 1/4, 1/2, 3/4 and all of internal application servers being attacked by SYN Flood with the source IP spoofing technique. The analyzed result in Figure 16 shows that when the system is attacked by SYN Flood, in comparison with the CM mechanism, OFS traffic increases with a small rate, just about 1.8% in case half of internal servers being attacked and over 2.5% when all the servers in the system are attacked with the rate of 5500 SYN pps each server.

4. Reduction in OpenFlow Traffic and Load on the Controller

Both the Avant-Guard's CM mechanism and the SSG solution operate based on monitoring the 3HS process of a TCP connection before requesting the controller to install FEs on the OFS. This not only prevents the resource consumption in the OFS by useless attack SYN packets but also protects the controller from being overloaded by messages of those attack TCP connections and reduces the OpenFlow traffic between the OFS and the controller.

Comparing the interaction between the controller and system entities in CM shown as Figure 1 and in SSG described as in Figures 5 and 14, it can be seen that for each TCP connection, SSG needs only one request to the controller but this figure is two in CM. With such a difference, the total number of messages exchanged with the controller to install FEs for legitimate TCP connections in SSG solution would be a half of the number in CM mechanism. This enhancement diminishes the load on the controller and so makes SSG more resistant to SYN Flood attacks than the Avant-Guard CM mechanism.

VI. Conclusion

Inspired by the CM mechanism in Avant-Guard scheme, the proposed SSG solution moves the SYN proxy, which monitors TCP 3HS connection processes, inside the OFS to locate it in a separated device, termed the SD. The ability to match the TCP Flag fields specified in OpenFlow 1.5 is applied to filter related packets in the OFS and forward them to the SD for monitoring the 3HS process. SSG uses the RST Cookie technique to authenticate Source IP address instead of SYN Cookie as in CM. Besides, by integrating SYN-Flood attack detection module, SSG processes incoming SYN packets depending on the attack state of the destination application server. Thanks to these improvements, SSG overcomes the shortcomings of the CM mechanism and can be used as an alternative solution that can be applied to all OpenFlow 1.5 supported switches without any modification. The experiment results show that during attack-free state, SSG does not affect packet exchange between the external clients and the internal application servers. When an internal server is under SYN Flood attack, SSG consumes less resources than CM with a negligible total traffic increase at the OFS interfaces. This shows that SSG is more resistant than CM under saturation attack by SYN Flooding on the data and control planes.

References

[1] Open Networking Foundation, "SDN architecture overview Version 1.0," 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf

[2] ——, "OpenFlow switch specification version 1.3.0 (Wire protocol 0x04)," 2012. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-spec-v1.3.0.pdf

[3] ——, "OpenFlow switch specification version 1.5.1 (Protocol Version 0x06)," 2015. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[4] T. Peng, C. Leckie, and K. Ramamohanarao, "Survey of network-based defense mechanisms countering the DoS and DDoS problems," *ACM Computer Survey*, vol. 39, no. 1, Article 3, 2007.

[5] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Proceedings of the IEEE SDN for Future Networks and Services (SDN 4FNS)*, Trento, Italy, Nov. 11-13 2013, pp. 1–7.

[6] S. Shin and G. Gu, "Attacking software-defined networks: A first feasibility study," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN'13)*, Hong Kong, China, Aug. 16 2013, pp. 165–166.

[7] R. Kandoi and M. Antikainen, "Denial-of-service attacks in OpenFlow SDN networks," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, ON, Canada, May 11-15 2015, pp. 1322–1326.

[8] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the ACM SIGSAC Conference on Computer & Communications security (CCS'13)*, Berlin, Germany, Nov. 04 - 08, 2013, pp. 413–424.

[9] Daniel J. Bernstein, "SYN cookies." [Online]. Available: https://cr.yp.to/syncookies.html

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

[11] CAIDA: Center for Applied Internet Data Analysis, "The CAIDA anonymized internet traces 2013 dataset." [Online]. Available: http://www.caida.org/data/passive/passive_2013_dataset.xml

[12] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for OpenFlow networks," in *Proceedings of the first workshop on Hot topics in software defined networks (HotSDN '12)*, Helsinki, Finland, Aug. 13 2012, pp. 121–126.

[13] L. Wei and C. Fung, "FlowRanger: A request prioritizing algorithm for controller DoS attacks in software defined networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, London, UK, Jun. 8-12, 2015, pp. 5254–5259.

[14] N.-N. Dao, J. Park, M. Park, and S. Cho, "A feasible method to combat against DDoS attack in SDN network," in *Proceedings of the International Conference on Information Networking (ICOIN)*, Siem Reap, Cambodia, Jan. 12-14 2015, pp. 309–311.

[15] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Towards autonomic DDoS mitigation using software defined networking," in *Proceedings of the Workshop on Security of Emerging Networking Technologies*, San Diego, CA, Feb. 8 2015, p. 7 pages.

[16] S. Shin, Y. Song, T. Lee, S. Lee, J. Chung, P. Porras, V. Yegneswaran, J. Noh, and B. B. Kang, "Rosemary: A robust, secure, and high-performance network operating system," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'14)*, Scottsdale, Arizona, USA, Nov. 03 - 07 2014, pp. 78–89.

[17] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, Feb. 8-11 2015, p. 15 pages.

[18] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "SPHINX: Detecting security attacks in software-defined networks," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, CA, Feb. 8-11 2015, p. 15 pages.

[19] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure

and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN '13)*, Hong Kong, China, Aug. 16 2013, pp. 55–60.

[20] R. Kloti, V. Kotronis, and P. Smith, "OpenFlow: A security analysis," in *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP)*, Goettingen, Germany, Oct. 07-10 2013, pp. 1–6.

[21] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10)*, Vancouver, BC, Canada, Oct. 4–6 2010, pp. 351–364.

[22] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proceedings of the Internet Network Management Conference on Research on Enterprise Networking (INM/WREN'10)*, San Jose, CA, Apr. 28-30 2010, p. 6 pages.

[23] P. Berde, W. Snow, G. Parulkar, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, and P. Radoslavov, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN'14)*, Chicago, Illinois, USA, Aug. 22 2014, pp. 1–6.

[24] A. Vishnoi, R. Poddar, V. Mann, and S. Bhattacharya, "Effective switch memory management in OpenFlow networks," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems (DEBS'14)*, Mumbai, India, May 26-29 2014, pp. 177–188.

[25] L. Dridi and M. F. Zhani, "SDN-guard: DoS attacks mitigation in SDN networks," in *Proceedings of the 5th IEEE International Conference on Cloud Networking (Cloudnet)*, Pisa, Italy, Aug. 17 2016, pp. 212–217.

[26] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS attack prevention extension in software-defined networks," in *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Rio de Janeiro, Brazil, Jun. 22-25 2015, pp. 239–250.

[27] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," in *Proceedings of the ACM SIGCOMM 2010 Conference*, New Delhi, India, Aug. 30 - Sep. 03 2010, pp. 351–362.

[28] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM conference*, Toronto, ON, Canada, Aug. 15 - 19 2011, pp. 254–265.

[29] H. Mekky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. Lakshman, "Application-aware data plane processing in SDN," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking (HotSDN'14)*, Chicago, Illinois, USA, Aug 2014, pp. 13–18.

[30] D. Kotani and Y. Okabe, "A packet-in message filtering mechanism for protection of control plane in OpenFlow switches," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 3, pp. 695–707, 2016.

[31] C. Schuba, I. Krsul, M. Kuhn, E. Spafford, A. Sundaram, and D. Zamboni, "Analysis of a denial of service attack on TCP," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 04-07 1997, pp. 208–223.

[32] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Computer Networks*, vol. 44, pp. 643–666, 2004.

[33] Eddy Wesley M., "Defenses against TCP SYN flooding attacks," *The Internet Protocol Journal*, vol. 9, no. 4, pp. 2–16, 2006.

[34] Postel, J., "Transmission control protocol, DAPRA internet program - protocol specification, rfc 793," 1981. [Online]. Available: https://tools.ietf.org/html/rfc793

[35] D. V. Tuyen and T.T. Huong, "A multi-criteria based software defined networking system architecture for DDoS-attack mitigation," *REV Journal on Electronics and Communications*, vol. 6, no. 3, pp. 50–60, 2017.

[36] The Linux Foundation Projects, "Data plane development kit (DPDK)." [Online]. Available: https://www.dpdk.org

[37] J. L. Deng, "Introduction to grey system theory," *The Journal of Grey Systems*, vol. 1, no. 1, pp. 1–24, 1989.

[38] T. Bohlin, *Practical Grey-box Process Identification: Theory and Applications (Advances in Industrial Control)*. Berlin, Heidelberg, Germany: Springer-Verlag, 2006.

[39] E. Kayacan, B. Ulutas, and O. Kaynak, "Grey system theory-based models in time series prediction," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1784–1789, 2010.

[40] D. Zhang, H. Wang, and K. G. Shin, "Change-point monitoring for the detection of DoS attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 4, pp. 193–208, 2004.

[41] J. L. Hellerstein, F. Zhang, and P. Shahabuddin, "A statistical approach to predictive detection," *Computer Networks*, vol. 35, no. 1, pp. 77–95, 2001.

[42] S. Wang, Q. Sun, H. Zou, and F. Yang, "Detecting SYN flooding attacks based on traffic prediction: A demonstration of the security communication networks class file," *Security and Communication Networks*, vol. 5, no. 10, pp. 1131–1140, 2012.

[43] The Linux Foundation Collaborative Project, "Open vSwitch." [Online]. Available: https://www.openvswitch.org

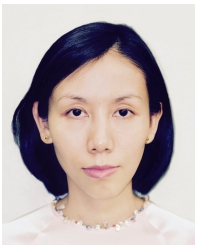
[44] Project Floodlight, "Floodlight OpenFlow controller." [Online]. Available: http://www.projectfloodlight.org/floodlight

[45] M. Goldstein, "BoNeSi: the DDoS botnet simulator." [Online]. Available: https://github.com/Markus-Go/bonesi

[46] W. Foundation, "Wireshark." [Online]. Available: https://www.wireshark.org

[47] Fred Klassen, "TCPreplay-PCAP editing and replaying utilities." [Online]. Available: https://tcpreplay.appneta.com

Dang Van Tuyen obtained his Bachelor degree in Electronics and Telecommunications in 1999 and Master degree in Telecommunication Engineering in 2008 from Hanoi University of Science and Technology (HUST), Vietnam. He is a lecturer at the Faculty of Electronics and Telecommunications, People's Police University of Technology and Logistics, Ministry of Public Security, Vietnam. Currently, he is pursuing the PhD program in the field of Telecommunication Engineering of HUST. His research interest includes: network security, SDN network, quality of service, wireless sensor network.



Truong Thu Huong is Associate Professor of the School of Electronics and Telecommunication, Hanoi University of Science and Technology (HUST). She is also Vice Director of Elitech, a project to promote all elite education programs of HUST. Truong's educational, research, and development work is oriented toward next generation networks, protocols and mechanism, traffic analyses, QoE/QoS measuring, green networking and deployment of new integrated multimedia services into fixed and mobile networks, network security and applications in the Internet of Things. She has been serving various international research conferences in different roles such as TPC member, publicity chair, organizer, track chair and serving international journals as reviewer. She is also active in capacity building; research and education development.