SLICOTS: An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks

Reza Mohammadi, Reza Javidan, and Mauro Conti, *Senior Member, IEEE*

***Abstract*—Software defined networking (SDN) is a novel networking paradigm which decouples control plane from data plane. This separation facilitates a high level of programmability and manageability. On the other hand, it makes the SDN controller a bottleneck and hence vulnerable to control plane saturation attack. One of the key mechanism to achieve control plane saturation is via TCP SYN flooding attack. This is one of the most effective and popular denial of service attack, in which the attacker produces many half-open TCP connections on the targeted server in order to degrade its availability. Furthermore, when applied to SDN, TCP SYN flooding attack also introduces *control plane saturation attack*. In particular, the attacker generates a significant number of TCP SYN packets and imposes data plane switches to forward them to the controller. As a result, the performance of the controller degrades and the controller will not be able to respond genuine requests in acceptable time. In this paper, we propose SLICOTS, an effective and efficient countermeasure to mitigate TCP SYN flooding attack in SDN. SLICOTS takes the advantage of dynamic programmability nature of SDN to detect and prevent attacks. SLICOTS is implemented in the controller, it surveils ongoing TCP connection requests, and blocks malicious hosts. We implemented SLICOTS as an extension module of OpenDayLight controller and evaluated it under different attack scenarios. The experimental results confirm that, compared to the state-of-art, SLICOTS reduces the response time overhead up to some 50%, while ensuring the same level of protection.**

***Index Terms*—TCP SYN flooding, SDN, security, SYN flooding countermeasure.**

I. Introduction

THE CONTINUOUS growth of current networks such as enterprise networks, data centers and cloud in terms of size and complexity is leading to the rise of Software Defined Networking (SDN) as a novel technology for simplifying network management. SDN separates data and control planes and provides capabilities to deploy and manage networks dynamically [1]–[3]. In SDN, logically centralized controller has a global view of the network and can be programmed directly by network administrators [4]–[6]. OpenFlow (OF) protocol is one of the most widely adopted open standards for SDN architecture that provides a communication interface between data plane switches and controller [7]. Each OF-based switch has a set of flow tables which is managed by the controller. Each entry of the flow table is a forwarding rule that comprises a set of header fields and related actions to apply to matching packets. All incoming packets to OF switch should be compared against its flow table entries. If an OF switch does not find any matching rule for an incoming packet, it forwards the packet inside a packet-in message to the controller to ask further actions [8]. In this way, for any new incoming packet, the OF switch has to ask the controller to install new rules. This mechanism leads to serious issues such as scalability and security. For example, an attacker could launch Denial of Service (DoS) attack by generating a large number of anomalous packets with different header fields [9], [10]. In this way, the OF switches which receive the packets, forward them to the controller and subsequently cause control plane saturation [11]. Therefore, the controller cannot answer the new request for benign users in time.

TCP SYN flooding is one of the most widespread DDoS attacks and has been introduced by Ziegler [12]. In this type of attack, attackers send a lot of SYN packets to a targeted server and cause degradation of the availability of the server. In normal TCP connection, three-way handshaking is needed between source and destination. When a TCP client sends a SYN packet, TCP server enters a listening state and answers with SYN-ACK packet. Furthermore, the server resources are frozen waiting for ACK message from the client. If the ACK message does not receive by the server for a certain amount of time, the server will release the resources [13]. In SYN flooding, the attacker creates many half-open TCP connections without sending ACK message. This behavior exhausts the targeted server resources and makes the server unresponsive to legitimate traffic [14]. Moreover, SYN flooding introduces data to control plane saturation attack in SDN [15]–[17]. Because in SYN flooding, attackers produce a huge number of new packets, and OF switches have to contact the controller and request relevant actions. Sending SYN packets in high rate by the attackers throttles throughput and processing capacity of the controller.

Manuscript received February 12, 2017; accepted May 2, 2017. Date of publication May 8, 2017; date of current version June 9, 2017. Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission (agreement PCIG11-GA-2012-321980). This work is also partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061), the EU-India REACH Project (agreement ICI+/2014/342-896), and by the projects "Physical-Layer Security for Wireless Communication", and "Content Centric Networking: Security and Privacy Issues" funded by the University of Padua. This work is partially supported by the grant n. 2017-166478 (3696) from Cisco University Research Program Fund and Silicon Valley Community Foundation. The associate editor coordinating the review of this paper and approving it for publication was E. Lupu. *(Corresponding author: Reza Mohammadi.)*

R. Mohammadi and R. Javidan are with the Department of Computer Engineering and Information Technology, Shiraz University of Technology, Shiraz 715555-313, Iran (e-mail: R.mohammadi@sutech.ac.ir; Javidan@sutech.ac.ir; conti@math.unipd.it).

M. Conti is with the Department of Mathematics, University of Padua, 35122 Padua, Italy.

Digital Object Identifier 10.1109/TNSM.2017.2701549

Recently, Fichera *et al.* [18] proposed OPERETTA as a solution for mitigating SYN flooding attack against Web servers. OPERETTA is implemented in the controller and acts as a proxy for TCP connections. OPERETTA first validates the source of a TCP connection and then installs a pair of forwarding rules along the path between the source and destination in both sides. Unfortunately, while OPERETTA is beneficial in the general case, it introduces a long delay for benign TCP connections. Moreover, OPERETTA is unable to detect the attacks in which the attackers complete TCP handshaking but do not send any data.

In this paper, we propose SLICOTS which is implemented in the controller, monitors ongoing SYN connections and detects and blocks malicious request in an efficient and effective manner. SLICOTS acts as an extension for OpenDayLight controller, has low overhead and does not sacrifice other benign traffics.

In summary, we make following contributions:

- We propose SLICOTS as a novel solution for countermeasuring TCP SYN flooding attack in SDN.
- We perform a comprehensive evaluation for our solution, which confirms the effectiveness of SLICOTS against TCP SYN flooding attack in SDN. Moreover, our experiments show a significant reduction of the benign request response time when compared to OPERETTA.

The remaining of this paper is organized as follows. In Section II, we review some related works in the area of SYN flooding attack and its countermeasures. Section III describes the problem and adversary model. In Section IV, we describe our proposed countermeasure SLICOTS. Experimental results are evaluated and analyzed in Section V. In Section V-C, we discuss some optional extensions for SLICOTS. Finally, conclusions are described in Section VI.

II. Related Works

As we mentioned in Section I, SYN flooding is one of the most effective and popular attacks that represents 90% of DoS attacks worldwide [19]. Hence, a lot of solutions have been proposed in the literature to defend against this type of attack. In this section, we classify countermeasures into two groups as follows:

A. Countermeasures in Legacy Networks

Most of the proposed solutions for mitigating SYN flooding attack in legacy networks such as SYN cookies, SYN cache and SYN proxy [20], [21] rely on changing end host TCP implementation. The main limitation of these solutions is that they only detect the attack on the receiver side. In this way, network resources are wasted, because in SYN flooding attack a large number of packets are transmitted through the network. Other solutions are using firewalls and Intrusion Detection System (IDS). A firewall has a set of static rules and compares each incoming packet against rules, and rejects malicious packets which violate the rules. Although firewall might detect and prevent trivial SYN flooding attack, it cannot detect DDoS attack as well as the attacks in which the attackers use spoofed source IP or different TCP source ports [18], [22]. Similar to a firewall, IDS cannot detect DDoS SYN flooding attack especially when spoofed source IP or different TCP source ports [23], [24]. This is because in DDOS attacks, attackers generate huge number of SYN requests with different source port numbers and IP addresses which makes it more difficult to configure IDS in order to prevent the attack. Moreover, using middle-boxes such as firewall and IDS in the edge of targeted network cannot prohibit the saturation of backbone network.

B. SDN-Based Countermeasures

SDN-based solutions for mitigating SYN flooding attack can either be implemented on the data plane or the control plane. In this section, we first review data plane countermeasures, and then we review the control plane countermeasures. AVANT-GUARD [15] has been proposed as a solution for detection and prevention of SYN flooding attack. AVANT-GUARD uses connection migration mechanism in data plane switches as a proxy for incoming SYN packets and limits the effect of the control plane saturation attack. During the TCP handshaking process, each data plane switch acts as a SYN proxy and protects the controller and destination from attack, and informs the controller about the requests which have performed successful TCP handshaking. Although AVANT-GUARD is beneficial in a general case, it causes long delay for establishing new connection for legitimate TCP requests. Moreover, it introduces buffer saturation attack [25] and limits the number of connections that a data plane switch can proxy to the number of available TCP port numbers [23]. Moreover, using AVANT-GUARD requires upgrading data plane equipments to support AVANT-GUARD features.

LineSwitch [25] is an efficient solution which removes the limitations of AVANT-GUARD and tackles buffer saturation attack. Similar to AVANT-GUARD, LineSwitch uses SYN proxy technique in data plane switches. But LineSwitch implements probabilistic blacklisting of network traffic for mitigating the buffer saturation attack. Using probabilistic blacklisting decreases the size of needed memory for storing the state of ongoing connections. Because LineSwitch proxies a minimum number of connections, in compared to AVANT-GUARD, it reduces the limitation of the number of connections. LineSwitch is an effective solution against control plane saturation attack, but as well as AVANT-GUARD, it is required to upgrade data plane switches when applying LineSwitch mechanism.

Chin *et al.* [26] proposed a collaborative technique for SYN flooding attack detection and containment. They have developed some new components such as monitors and correlators in SDN architecture for mitigating DDOS attacks. The monitor continuously listens to ongoing traffic for detecting the SYN packets with different source IP addresses which denotes IP spoofing. When the monitor detects an attack, it informs the correlator by sending an alert message which contains a number of source IP addresses found in SYN packets. Upon receiving the alert message, the correlator issues a query of the switch flow table which the origin of the malicious traffic connected to it. If the correlator finds that the IP address in the alert message originated from a port that has had a different IP

address in the original database, it concludes that the traffic is malicious. As a result, the correlator informs the controller and the controller blocks the malicious host. Chin *et al.* [26] have mentioned this technique works only for SYN flooding attack which the attackers use spoofed IP addresses. Furthermore, this method requires connecting a monitor to each switch in the network which is not possible in general case.

SPHINX [27] is a framework which has been proposed to detect security attacks in SDN. SPHINX is implemented on the control plane and can detect SYN flooding attack by investigating the rate of packet-in messages which correspond to the new SYN requests. If the rate of new SYN request is above the administrator-specific threshold, SPHINX raises an alarm. However, setting a control threshold for detecting SYN flooding attack is a static solution and would likely generate false alarms [28]. The reason is that SPHINX does not investigate the SYN requests to distinguish the legitimates from illegitimate requests and only strictly controls SYN requests with a predefined threshold.

Another solution to SYN flooding attack named OPERETTA is proposed by Fichera *et al.* [18], and can be utilized in both centralized and delocalized controllers. OPERETTA is an OpenFLow-based approach which is implemented in the controller and investigates new SYN requests for detecting and rejecting malicious requests. Unlike AVANT-GUARD and LineSwitch, in which data plane switches acted as a proxy, OPERETTA is an installed module on the controller and acts as a proxy between the TCP client and server. In OPERETTA, the data plane switches send new SYN packets to the controller and the controller respond with a SYN-ACK to the client. If the client sends the ACK message, the controller sends an RST message to the host and installs a rule between the client and server in both sides. In this way, the first attempt to establish TCP connection has always been rejected even for legitimate requests. OPERETTA defines a counter of SYN requests for each host. This counter denotes to the SYN request which has not acknowledged yet. The aim of this counter is to limit the number of malicious requests, and if the number of SYN requests violates the predefined threshold, the controller will block the attacker. Although OPERETTA can detect and prevent SYN flooding attack, it rejects the first request for establishing any TCP connection. Moreover, an adversary might flood the controller with complete TCP handshakes and impose the controller to install new rule. After the installation of the new rule, the adversary can send its SYN request to the server. In summary, the adversary only needs to perform complete TCP handshaking to pass the OPERETTA.

In order to overcome the weaknesses of OPERETTA, this paper proposes SLICOTS which does not sacrifice the benign traffics and also reduces the response time of the legitimate TCP requests.

III. Problem Statement

In this section, we describe some background knowledge about SYN flooding attack and the adversary model for this attack in SDN.

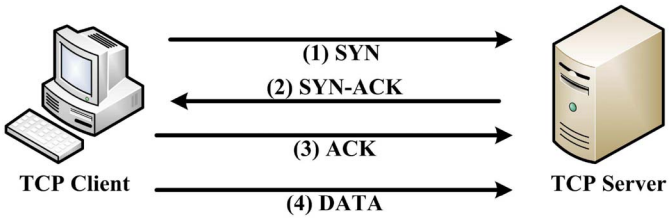


Fig. 1. Normal TCP Handshaking.

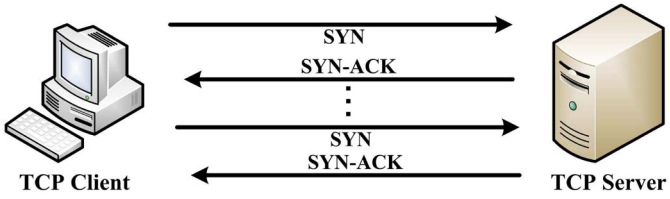


Fig. 2. SYN flooding Attack.

A. Background on SYN Flooding Attack

In a normal TCP connection, the client and server have to perform three-way TCP handshaking. Fig. 1 illustrates a successful establishment of TCP connection. According to Fig. 1, at first, the client sends a SYN packet to the server. Upon receiving the SYN packet, the server responds with SYN-ACK packet and allocates TCP stack for the new connection request and then enters to listening state. Such state called *transmission control block* (TCB) [13], and server retains in TCB state for a certain amount of time, even if the client does not respond with ACK. If the client answers with ACK, the TCP connection establishes successfully and as a result, the client and server can exchange their data [13].

However, this mechanism is vulnerable to SYN flooding attack in which the attackers creates a large number of half-open connection on the targeted server without sending ACK. Afterward, the resources of the server will be exhausted, and no new connection can be established for benign users. Fig. 2 shows the SYN flooding attack between a client and server.

To increase the effectiveness of the attack, in most cases, a large number of attackers launch DDoS attack and send a lot of SYN packets using spoofed source IP address and make difficult for targeted server to detect the source of the attack. In this way, the SYN-ACK packet will be delivered to the host which has the same IP address as spoofed IP.

B. Adversary Model

In this section, we present SYN flooding in SDN. According to Fig. 3, the attacker triggers SYN flooding attack by sending a new SYN packet (action (1) in Fig. 3). Once OF switch receives this SYN packet, which does not matched with any forwarding rule in its flow table, forwards it to the controller (action (2) in Fig. 3).

The controller calculates an appropriate route to the targeted server and installs new forwarding rule on the OF switch (action (3) in Fig. 3). Then, the OF switch forwards the SYN packet to the targeted server (action (4) in Fig. 3). The targeted server receives the SYN, allocates buffer to new request and

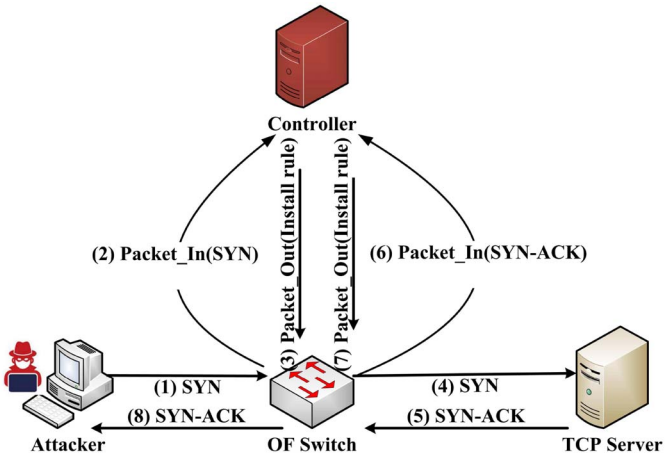


Fig. 3. SYN flooding in SDN.

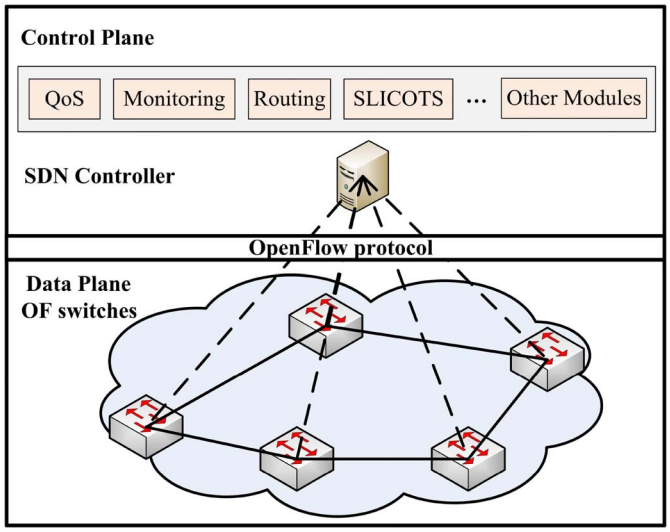


Fig. 4. The overall architecture of our solution.

responds with SYN-ACK (action (5) in Fig. 3). Upon receiving the SYN-ACK, the OF switch forwards it to the controller and ask forwarding rule (action (6) in Fig. 3). The controller calculates another route for backward path and installs related forwarding rule on OF switch (action (7) in Fig. 3). Finally, the OF switch forwards the SYN-ACK to the destination. In this scenario, if the attacker uses spoofed IP address, the controller installs second forwarding rule (action (7) in Fig. 3) in such a way the SYN-ACK delivered to the host which the attacker forged its IP. It is evident that in this way, SYN flooding leads to exhaustion of data and control plane resources and cause control plane saturation rather than the targeted server availability degradation. Fig. 3 shows the attack scenario for only one attacker, while in general, this type of attack has simultaneously being performed by many attackers namely DDoS SYN flooding [29].

IV. Proposed Solution

To address the problems discussed in Section III, we propose SLICOTS as an extension for securing SDN control plane. The main objective of SLICOTS is to detect and mitigate SYN flooding attack in SDN. In this section, we illustrate the overall architecture of our proposed countermeasure (Section IV) and then we present SLICOTS design in details (Section IV-B).

A. Overall Architecture

SLICOTS takes the advantages of reactive flow installation in SDN to detect and prevent the attack. It is worth to mention that SLICOTS is capable of detecting and preventing SYN flooding attack in situations in which the SDN controller acts in reactive mode. Therefore, SLICOTS is not applicable for the networks in which the SDN controller installs forwarding rules proactively. Moreover, SLICOTS does not consider flow rule aggregation and installs two paths between the client and server of each individual TCP session, i.e., one for forward path and the other for backward path. The conceptual architecture of SLICOTS is illustrated in Fig. 4.

As shown in Fig. 4, SLICOTS is a security module implemented in the control plane and works concurrently with other modules of the controller. SLICOTS surveils all ongoing TCP handshaking process between network users, and installs short-term forwarding rules on the data plane switches during TCP handshaking process. Due to temporary forwarding rules are eliminated after a certain amount of time, it immunes data plane switches from TCAM exhaustion attack [30]. Once a successful TCP handshaking completed between two users, the controller installs permanent forwarding rules for the client and server. Otherwise, the TCP connection between the requested host and the targeted server remains half-open. If the number of half-open TCP connection for a specific requested host exceeds a predefined threshold, SLICOTS realize that it is a SYN flooding attack and subsequently install a forwarding rule for blocking the requested host at the edge switch. Hence, the future TCP requests from malicious host will be blocked by the data plane, and the control plane will not be saturated anymore by the malicious host.

B. SLICOTS Detection and Prevention Mechanism

Fig. 5 shows the flow diagram of SLICOTS in details. As we mentioned in Section I, when an OF switch receives a new packet which is not matched with any of its flow table entries, it encapsulates the packet in a packet-in message and forwards to the controller. Upon receiving a new TCP packet by SLICOTS, it first checks the type of incoming packet. If the packet is SYN, SLICOTS extracts required information (source MAC, destination MAC, source TCP port, and destination TCP port) and stores these values with another field named status as a new record in a list. This list namely *pending_list* and contains the uncompleted TCP connections between the hosts in the network. The status field of each record in this list indicates the last situation of the TCP connection. This field value for the SYN request is set to "SYN" and means that only SYN packet is transmitted for the related record in the list. Since IP spoofing attack is more widespread than MAC spoofing attack [18], [31], we implemented SLICOTS to store MAC address in the *pending_list*

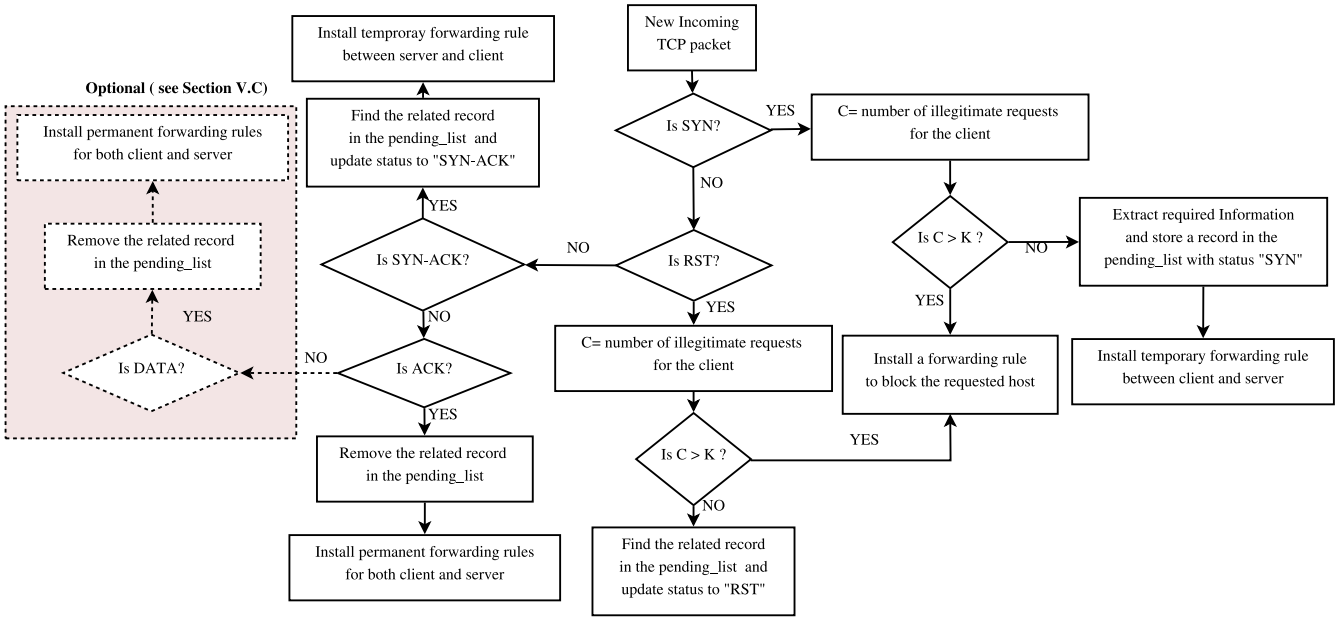


Fig. 5. The flow diagram of SLICOTS for a TCP request.

rather than IP address. Then, SLICOTS calls routing module to calculate an appropriate route toward the targeted server and then installs temporary forwarding rules for SYN packet on the OF switches along the route.

The forwarding rule for the edge OF switch which the requested host connected to it contains two actions: one action for forwarding the packets to the next hop and another for forwarding a copy of the packet to the controller. The installed forwarding rules on other OF switches along the route, has only one rule for forwarding the packets toward the targeted server. The temporary forwarding rule can be produced by setting *hard_timeout* field to a certain amount of time *T*. In this paper we set this time value to three seconds which is the timeout value for waiting SYN-ACK packet [32]. Therefore, the temporary forwarding rules will be eliminated from the OF switches after three seconds. The targeted server responds with SYN-ACK after receiving the SYN packet, and because the edge OF switch connected to the server does not have any forwarding rule for the first SYN-ACK packet, it forwards the SYN-ACK to the controller.

Upon receiving the SYN-ACK, SLICOTS extracts the required information, finds the related record in *pending_list* and changes its status from "SYN" to "SYN-ACK". Then similar to the previous step, it calls the routing module to calculate the reverse route and installs temporary forwarding rules on the OF switches along the route.

In addition to this, the forwarding rule for the edge switch the targeted server connected to it, contains two actions for forwarding the incoming TCP packet along the route and a copy of the packet to the controller. Once the requested host receives SYN-ACK, it responds with ACK. Since the edge OF switch has two actions, it forwards the packet toward the targeted server and also sends a copy of the packet to the controller. SLICOTS receives the ACK; it realizes that this is a legitimate TCP connection. Therefore, it finds the related record and then removes the record from the *pending_list*. Fig. 6 shows SLICOTS functionalities for a benign user.

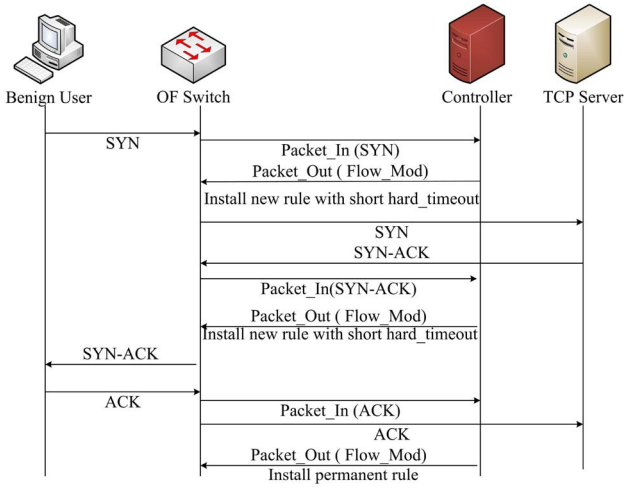


Fig. 6. SLICOTS functionalities for benign users.

One of the most important advantages of SLICOTS is that it can also detect the SYN flooding attack in which the ACK packet takes different route from the one taken by the SYN packet. Because SLICOTS installs the routes after completion of TCP handshaking, there will not be cases in which an ACK packet that takes different route, is identified as a malicious behavior.

In some cases, the targeted server might respond the SYN request with RST packet [13], i.e., sending SYN request to a closed TCP port at the targeted server. In such cases, SLICOTS changes the status field of the related record in *pending_list* to "RST".

Any record in *pending_list* with the status of "SYN","SYN-ACK" or "RST" is an illegitimate request. As shown if Fig. 5,

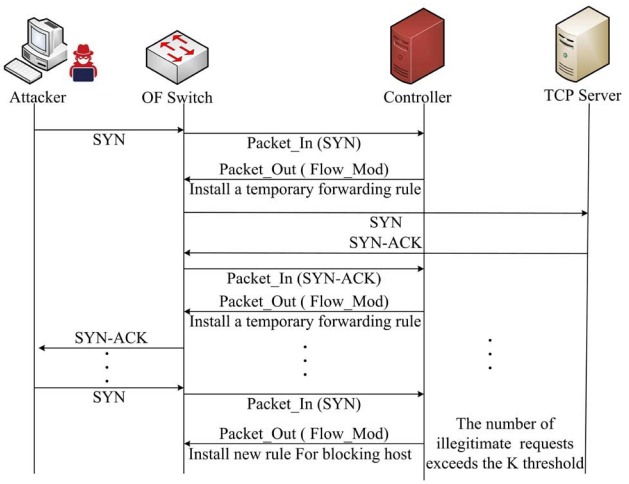


Fig. 7. SLICOTS functionalities for malicious users.

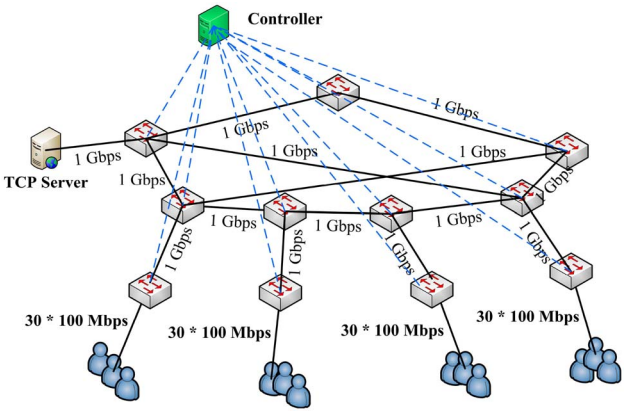


Fig. 8. The test-bed network.

upon receiving a new SYN packet, SLICOTS counts the number of illegitimate records. If the number of illegitimate requests for a specific requested host exceeds the threshold *K*, SLICOTS installs a forwarding rule on the edge OF switch the requested host connected to it and blocks it based on the MAC address. Similar to OPERETTA and SPHINX, SLICOTS uses the threshold *K* to prevent malicious hosts. But unlike SPHINX, SLICOTS has the ability to distinguish the legitimates from illegitimate requests and applies the threshold *K* only for the illegitimate requests. Blocking based on MAC address leads to prevent the attackers that use spoofed IP addresses. Fig. 7 shows SLICOTS functionalities for a malicious user.

The time and space complexity of SLICOTS depend on the threshold *K* and number of hosts in the network. Since SLICOTS stores at most *K* records for each host into the *pending_list*, the maximum size of the *pending_list* in the worst case is *n.K*, where *n* is the number of hosts in the network. As the result, the time and space complexity of SLICOTS is *O(n.K)* that shows the quickness and efficiency of SLICOTS.

As we mentioned, when a user sends a TCP request to a closed port on the targeted server, the server responds with RST. Due to in SLICOTS, any record with "RST" status indicates to an illegitimate request, it can also detect TCP port scanning [33] when the number of "RST" exceeds the *K* threshold as well as SYN flooding attack.

V. Evaluation and Discussion

In this section, we conduct a comprehensive simulation study and analyze the results to evaluate the performance of SLICOTS against OPERETTA and normal SDN. In normal SDN, there is no defense mechanism against SYN flooding attack, and we will show that how SLICOTS can improve the security of normal SDN.

A. Experimental Setup

We implement SLICOTS as a security module upon the OpenDayLight controller [34], a well-known and popular SDN controller platform which is written in Java language. To emulate the data plane of test-bed network, we use mininet [35], a popular SDN emulation tool. We run our experiments on a PC with CPU Intel Core i7-4700MQ-2.4 GHz and 6GB RAM. As we mentioned in Section II, OPERETTA is one of the state-of-art solutions to tackle SYN flooding attack which has been implemented in control plane upon the POX controller. Since SLICOTS is also implemented in control plane, we compare it against OPERETTA. We implemented both SLICOTS and OPERETTA on OpenDayLight controller. In order to make a fair comparison, we have tried to consider the implementation details of OPERETTA on OpenDayLight. In addition to this, we designed the same experiments and settings as used for OPERETTA which is shown in Fig. 8. Nonetheless, the authors of OPERETTA have not provided more details about simulation parameters such as time interval between SYN requests.

According to Fig. 8, the test-bed network comprises 11 OF switch which are connected together via 1 Gbps links. The server is connected to the network using 1 Gbps link while each of other hosts uses a 100 Mbps link. In addition to server, there are 122 hosts which 120 hosts are potential attackers and other two hosts are benign users. We used *Curl* [36] as a tool to generate legitimate HTTP connection requests by benign users, and *Hping*, [37] to perform SYN flooding attack by attackers. Each attacker uses *Hping* to produce SYN packets with spoofed IP address in different rates. However, SLICOTS can detect and prevent SYN flooding attack for both spoofed and non-spoofed IP addresses scenarios. But in order to compare it with OPERETTA, in this section, we consider spoofed IP addresses scenario. Furthermore, each benign host requests a Web page from the server every four seconds. One of the main important parameter which affects the efficiency of SLICOTS in attack detection is *K* threshold. In fact, choosing a proper value for *K* threshold can improve the efficiency of SLICOTS, because setting a large value for *K* increases attack detection time. Contrary, setting a small value might also cause detecting a legitimate user as an attacker. We conducted simulations in four different scenarios as follows:

- Scenario *S1*. The number of attackers varies from 20 to 120 and each attacker sends three SYN packets with spoofed IP every second.

- Scenario *S2*. The number of attackers is set to 50 while the number of fake SYN packets sent by each attacker varies from 1 to 6 per second.

Since the main goal of this paper is to introduce SLICOTS as a new countermeasure for SYN flooding attack and to show its superiority over OPERETTA, the scenarios *S1* and *S2* are designed according to OPERETTA paper. Specifically, the following scenarios have been considered for scalability and sensitivity analysis of SLICOTS:

- Scenario *S3*. This scenario is considered for the sake of scalability of SLICOTS for different rates of SYN flooding attack. In order to assess the performance of SLICOTS under high rate of SYN packets, we have chosen malicious SYN packet rate from 50 to 350 packets per second. In this scenario, the number of malicious and benign hosts is set to 120 and 2, respectively. It is worth to mention that due to hardware limitation, in this paper, we could only test SYN packet rate maximum up to 350 packet per second for each malicious host.
- Scenario *S4*. As we mentioned in Section IV-B, SLICOTS installs at most K temporary forwarding rules for malicious hosts which causes a few *False Positive Rate* (FPR) in detection of SYN flooding attack. Therefore, we consider different values for threshold K in order to investigate the performance of SLICOTS in terms of FPR and *True Positive Rate* (TPR) . In this scenario, the number of malicious and benign hosts is set to 120 and 2, respectively; and threshold K varies from 10 to 100.

We evaluate the experimental results using following performance metrics:

- *HTTP response time:* denotes to time between requesting a Web page by a benign user and receiving it from the attacked TCP server.
- *Number of installed entries:* denotes the average number of forwarding entries which have been installed on the OF switches by the controller.
- *Attack detection time:* the time between sending a first fake SYN packet by an attacker and detecting the attack by the controller.
- *CPU utilization:* the average CPU consumption to the overall available resources.

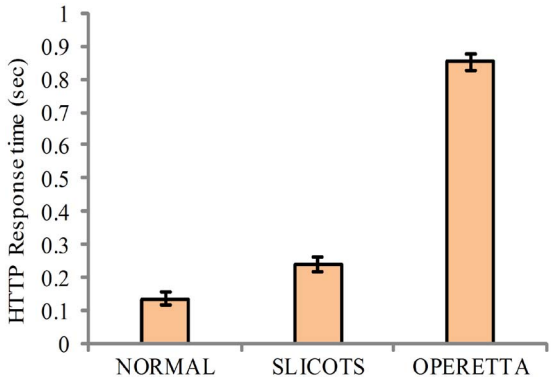


Fig. 9. HTTP response time in a non-attack scenario.

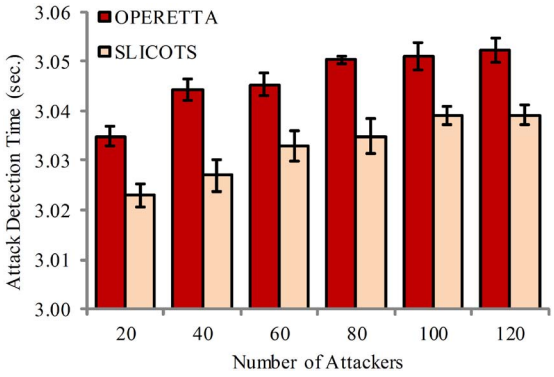


Fig. 10. Attack detection time vs. different number of attackers *(S1)*.

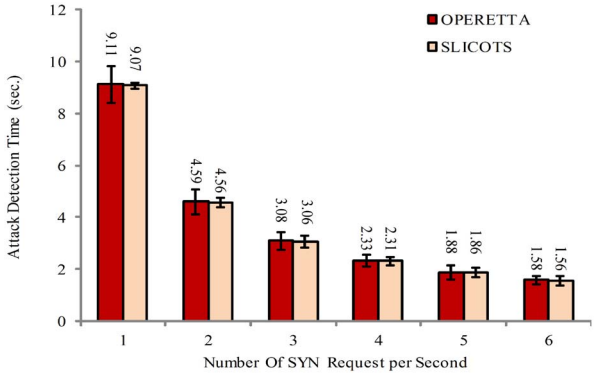


Fig. 11. Attack detection time vs. different SYN packet rates *(S2)*.

B. Experimental Results

In this section, we discuss and analyze the results for the two scenarios under different attack rates and attacker numbers. First of all, we investigate the HTTP response time in a scenario without any attacker. As shown in Fig. 9, it is evident that SLICOTS has a considerable low overhead in compared to OPERETTA in the non-attack scenario. This is because OPERETTA always causes rejection of the first attempts to create a TCP connection. Contrary, SLICOTS installs temporary forwarding rule for the TCP handshaking packets and does not cause any rejection.

Fig. 10 and Fig. 11 depict the average attack detection time for the two scenarios. Because the results for scenario-2 are very close together, we use bar chart rather than line chart to show the difference clearly. According to these two sets of results, SLICOTS outperforms OPERETTA in terms of attack detection time. The results are very close together, because the both solutions use the same constant value as a threshold for blocking the attackers. The reason that OPERETTA has a little bit longer attack detection time is that OPERETTA acts as a proxy and sends SYN-ACK to respond SYN packet of the attackers. Hence, this mechanism causes a short delay to detect the attacker. According to Fig. 11, the detection time decreases as the rate of sent SYN packet increases. This is due to the fact that SLICOTS and OPERETTA both detect the attack when receive the *(K+1)-th* malicious SYN packet. Therefore, it is evident that in low rate attacks, the detection time is longer than high rate attacks.

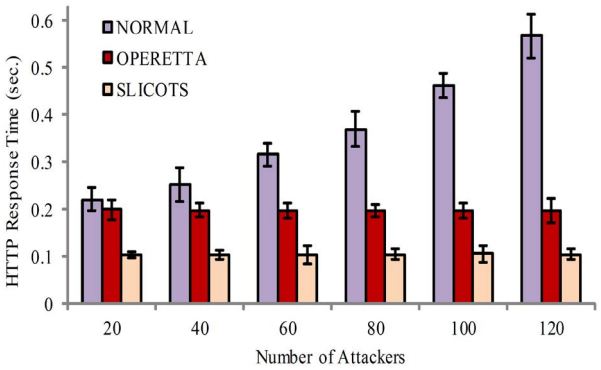


Fig. 12. HTTP response time for benign requests vs. different number of attackers *(S1)*.

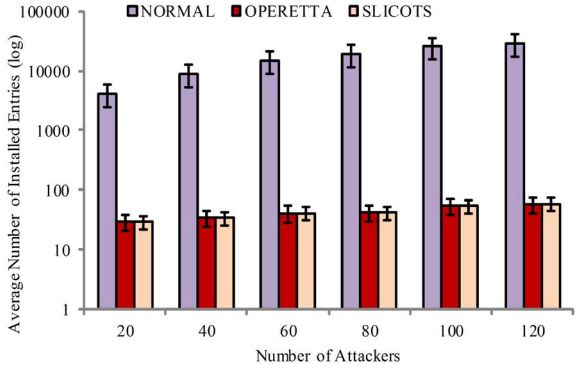


Fig. 14. Average number of installed entries vs. different number of attackers *(S1)*.

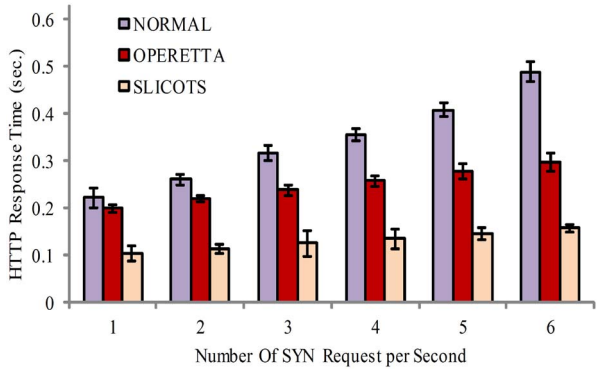


Fig. 13. HTTP response time for benign requests vs. different malicious SYN packet rates *(S2)*.

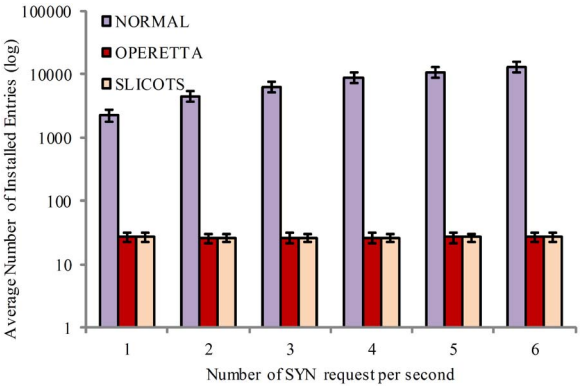


Fig. 15. Average number of installed entries vs. different malicious SYN packet rates *(S2)*.

Fig. 12 and Fig. 13 show the average HTTP response time for the benign users requests. As we explained in Section IV-B, SLICOTS installs the temporary forwarding rules during TCP handshaking and installs permanent forwarding rules after successful handshaking. This policy does not impose additional delay for benign requests. On the other hand, OPERETTA always rejects the first attempt and installs forwarding rules after receiving the ACK packet of the first attempt. This causes a significant delay for legitimate requests as shown in Fig. 12 and Fig. 13. The results confirm that SLICOTS considerably outperforms OPERETTA and does not sacrifice other legitimate users. Moreover, in both scenarios, HTTP response time increases as the number of attackers or rate of SYN packet increases in Normal SDN. Since in Normal SDN there is no security policy for mitigating SYN flooding, the resource of the targeted server will be exhausted and as a result, the response time increases.

Fig. 14 and Fig. 15 show the average number of installed forwarding entries. Because the difference between the results of SLICOTS and OPERETTA in compared to Normal SDN are significant, we depict the results in logarithmic scale. It is obvious due to in Normal SDN there is no policy for countermeasuring against attacks, the controller installs forwarding rules for each request and subsequently as shown in Fig. 14 and Fig. 15 the number of installed entries increases considerably. On the other hand, SLICOTS and OPERETTA block the attackers after detecting their malicious behaviors. Hence, the SYN requests of blocked attackers are blocked by the edge switches. Though SLICOTS installs forwarding rules for all TCP connection requests, the temporary rules for half-open connections are eliminated after a certain amount of time. For this reason, SLICOTS prevents the exhaustion of TCAM memory of the switches.

As we mentioned in Section IV-B, SLICOTS is a lightweight security module and can mitigate SYN flooding without imposing considerable overhead. Fig. 16 and Fig. 17 confirm SLICOTS outperforms OPERETTA and Normal SDN in terms of CPU utilization. In Normal SDN, there is no mechanism to prevent attackers and as a result, the controller will be saturated by SYN requests.

Fig. 16 and Fig. 17 show that SLICOTS and OPERETTA can significantly prohibit control plane saturation by preventing and blocking malicious users. Because OPERETTA responds the SYN request with SYN-ACK packets even for malicious users and also responds the ACK message of the first attempt with RST packet, it has a little more CPU consumption in compared to SLICOTS. Contrary, SLICOTS takes the advantages of temporary forwarding rules mechanism of OpenFlow in an efficient manner and has less CPU overhead.

In addition to Fig. 16 and Fig. 17, Fig. 18 also demonstrates that the high rates of malicious SYN packets saturates the control plane in normal SDN and as the result the CPU utilization percentage reaches close to 100%. The reason is that in normal SDN, the controller have to process each request

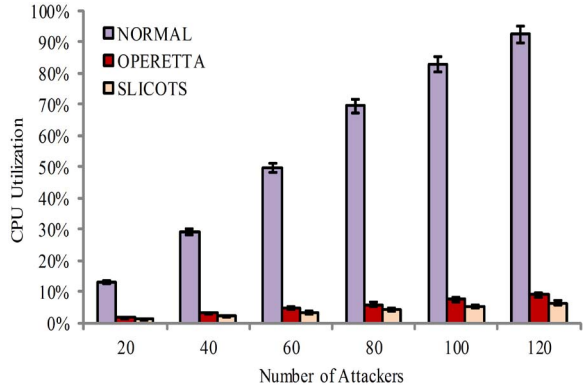


Fig. 16. CPU utilization vs. different number of attackers *(S1)*.

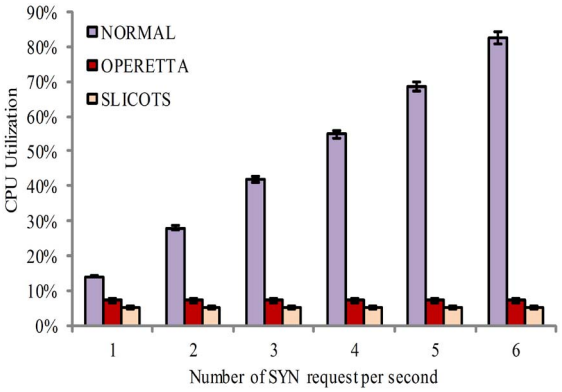


Fig. 17. CPU utilization vs. different malicious SYN packet rates *(S2)*.

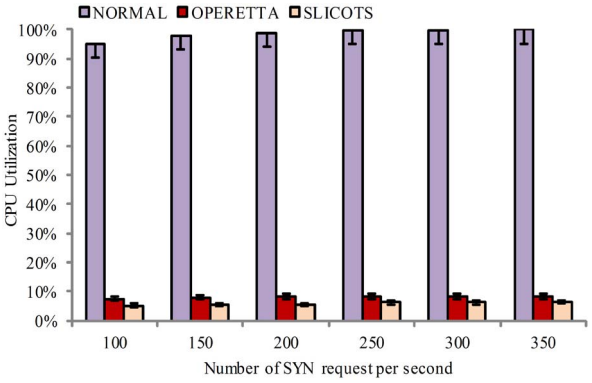


Fig. 18. CPU utilization vs. high rates of malicious SYN packet *(S3)*.

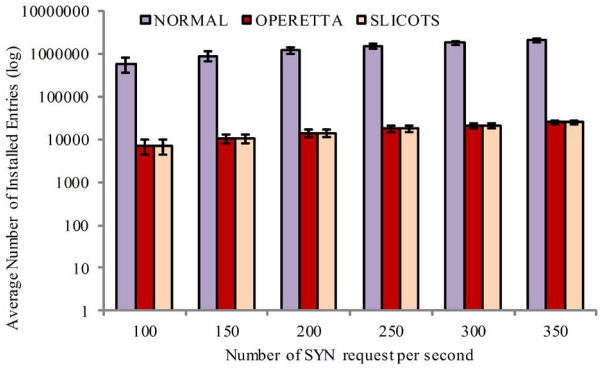


Fig. 19. Average number of installed entries vs. high rates of malicious SYN packet *(S3)*.

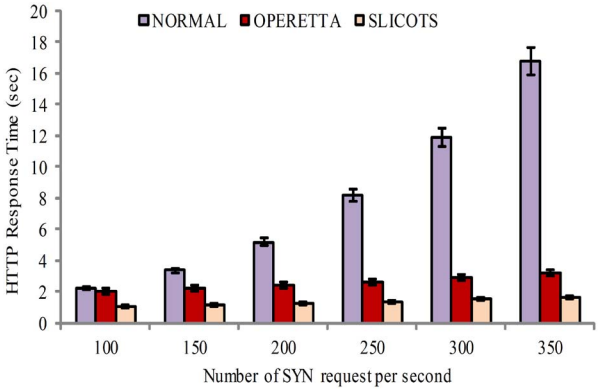


Fig. 20. HTTP response time for benign requests vs. high rates of malicious SYN packet *(S3)*.

and install forwarding rules for the request. On the other hand, OPERETTA and SLICOTS block the malicious hosts and subsequent requests of the malicious hosts will never reach to the controller.

As we aforementioned, one of the ill-effects of SYN flooding attack in SDN is to exhaust the *TCAM* memory of the data plane switches. Fig. 19 depicts that under high rates of malicious SYN packets, the number of installed entries in normal SDN is significantly higher than the OPERETTA and SLICOTS. This is because normal SDN does not care about the number of requests and installs forwarding rules for each new request. Contrary, SLICOTS and OPERETTA prohibit the malicious requests and install permanent forwarding rules only for benign requests. Therefore, the average number of installed entries for SLICOTS and OPERETTA are the same for different rates of attacks.

Fig. 20 shows as well the high rates of malicious SYN packet rates significantly increase the HTTP response time in normal SDN. As the matter of fact, due to lack of any defensive mechanism in normal SDN, the controller receives and processes each request. This behavior imposes additional delay and as a result, in high rate of attack causes longer delay for establishing forwarding rules, i.e., longer HTTP response time. Fig. 20 also depicts that SLICOTS outperforms OPERETTA in terms of response time for high rates of attack. The reason is that OPERETTA rejects the first request for establishing any TCP connection which causes additional delay.

As we aforementioned, attack detection time in OPERETTA and SLICOTS depends on the K threshold and the rate of attack. As shown in Fig. 21, for both OPERETTA and SLICOTS the attack detection time decreases as the rate of attack increases. This is because in a high rate of attack the K threshold exceeds sooner than a low rate of attack and therefore causes shorter attack detection time. It can be concluded from Fig. 18 to Fig. 21 that SLICOTS is a scalable countermeasure method and can detect and prevent SYN flooding attack even in the scenarios in which the attackers generate high rate of SYN packets.

TPR and FPR are two important metrics to assess the sensitivity and specificity of an attack detection algorithm. *Receiver*

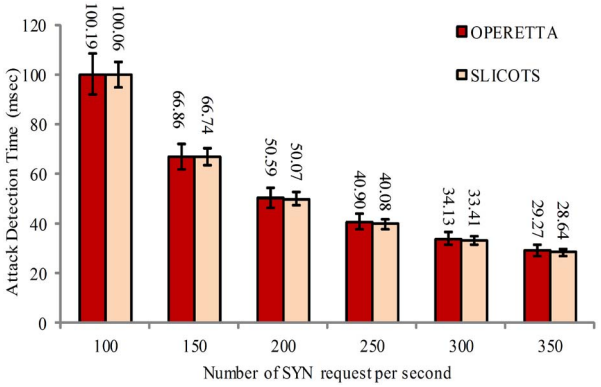


Fig. 21. Attack detection time vs. high rates of malicious SYN packet *(S3)*.

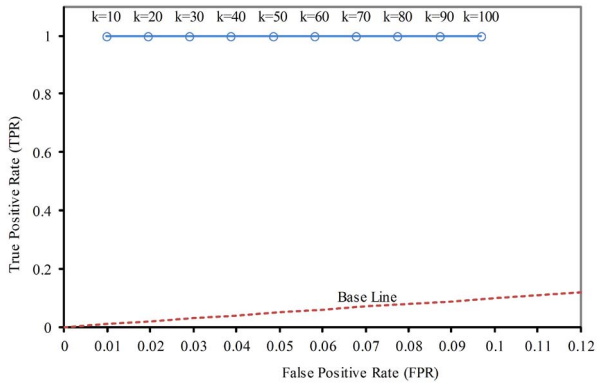


Fig. 22. Receiver Operating Characteristic for SLICOTS *(S4)*.

Operating Characteristic (ROC) chart is a useful tool to depict the tradeoff between the TPR and FPR of SLICOTS. In this paper, the FPR (1-specificity) is the percentage of malicious SYN packets which are incorrectly identified as benign SYN packets. Furthermore, the TPR (sensitivity) is the percentage of benign packets of the benign hosts which are correctly identified as benign packets. Since SLICOTS installs at most K temporary forwarding rules with duration of three seconds, each malicious host can send at most K malicious SYN packets to the targeted server. Obviously, this behaviour causes a little FPR. As shown in Fig. 22, the TPR of SLICOTS is the same for any value of K threshold, and is equal to one, which means SLICOTS can correctly detect benign requests regardless of K threshold value. Moreover, Fig. 22 shows that the FPR of SLICOTS increases smoothly with increasing K threshold value.

C. Discussion and Improvements

As illustrated in Section V, SLICOTS can effectively mitigate SYN flooding and also outperforms OPERETTA in terms of efficiency and attack detection time. Unlike OPERETTA, SLICOTS installs temporary forwarding rules for TCP handshaking process very fast and after validating the request installs permanent forwarding rules. Furthermore, in OPERETTA, it is necessary to change the end users TCP application as a way to request TCP connection for the second time after the failure of the first attempt.

As we discussed in Section II, there is another type of SYN flooding attack in which the attacker completes TCP three-way handshaking, but does not send any data packet afterwards [15], [25]. Due to OPERETTA uses migration procedure for TCP connection, it cannot address this type of attack. On the other hand, SLICOTS can tackle this attack in an efficient manner. To solve this issue, as shown in Fig. 5, we can add an optional step to SLICOTS (dotted box in Fig. 5). In this way, instead of waiting for the ACK packet, the controller waits until receives the first data packet. Upon receiving the first data packet from the client, the controller removes the pending request and installs permanent forwarding rules between the requested client and the targeted server.

Furthermore, we mentioned in Section IV-B that SLICOTS considers the current status of each request. Therefore, SLICOTS can detect TCP port scanning attack [30] as well as TCP SYN flooding attack. In TCP port scanning attack, an attacker sends a large number of SYN request with different TCP destination port to investigate open ports at the targeted server. If the requested port is close, the server responds with RST. In this case, upon receiving the RST packet, SLICOTS changes the status field of the related request to "RST" as shown in Fig. 5. Due to "RST" status indicates an illegitimate request, SLICOTS can detect the attack when the number of illegitimate requests exceeds K threshold.

VI. Conclusion

In this paper, we introduced SLICOTS as a lightweight extension for securing SDN against SYN flooding attacks. SLICOTS is implemented on control plane and monitors all ongoing TCP connections in network for detecting and preventing SYN flooding attack. For each TCP connection request, SLICOTS installs temporary forwarding rules during TCP handshaking process, and after validation of a request, installs permanent forwarding rules between the client and server. Moreover, SLICOTS blocks the attacker that makes a large number of half-open TCP connections. One of the main advantages of SLICOTS is that it does not sacrifice other legitimate requests. In this paper, we evaluated SLICOTS under different scenarios, and compared it against Normal SDN and state-of-art OPERETTA. Experimental results showed that SLICOTS is an efficient solution for mitigating SYN flooding attack and outperforms OPERETTA.

References

[1] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and openflow: From concept to implementation," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.

[2] B. Görkemli, A. M. Parlakisik, S. Civanlar, A. Ulas, and A. M. Tekalp, "Dynamic management of control plane performance in software-defined networks," in *Proc. IEEE NetSoft Conf. Workshops (NetSoft)*, Seoul, South Korea, 2016, pp. 68–72.

[3] R. Mohammadi and R. Javidan, "An adaptive type-2 fuzzy traffic engineering method for video surveillance systems over software defined networks," *Multimedia Tools Appl.*, pp. 1–16, 2016. [Online]. Available: http://link.springer.com/article/10.1007/s11042-016-4137-0

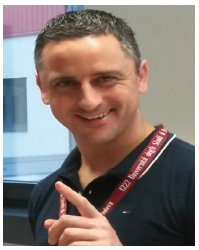
[4] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[5] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2211–2219.

[6] R. Mohammadi and R. Javidan, "An intelligent traffic engineering method over software defined networks for video surveillance systems based on artificial bee colony," *Int. J. Intell. Inf. Technol.*, vol. 12, no. 4, pp. 45–62, 2016.
[7] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[8] B. Pfaff, B. Lantz, and B. Heller, *Openflow Switch Specification, Version 1.3. 0*, Open Networking Foundation, 2012.
[9] L. Wei and C. Fung, "Flowranger: A request prioritizing algorithm for controller dos attacks in software defined networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., 2015, pp. 5254–5259.
[10] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 623–654, 1st Quart., 2015.
[11] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Trans. Rel.*, vol. 64, no. 3, pp. 1086–1097, Sep. 2015.
[12] B. Ziegler, "Hacker tangles Panix Web site," *Wall Street J.*, vol. 12, Sep. 1996.
[13] J. B. Postel, Ed., "Internet Protocol," Internet Request For Comments RFC 791, Sep. 1981.
[14] W. Chen and D.-Y. Yeung, "Defending against TCP SYN flooding attacks under different types of IP spoofing," in *Proc. Int. Conf. Netw. Int. Conf. Syst. Int. Conf. Mobile Commun. Learn. Technol. (ICNICONSMCL)*, 2006, p. 38.
[15] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Berlin, Germany, 2013, pp. 413–424.
[16] H. Wang, L. Xu, and G. Gu, "Floodguard: A DoS attack prevention extension in software-defined networks," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, Rio de Janeiro, Brazil, 2015, pp. 239–250.
[17] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*, Menlo Park, CA, USA, 2011, pp. 161–180.
[18] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, "Operetta: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD attacks against Web servers," *Comput. Netw.*, vol. 92, no. 1, pp. 89–100, 2015.
[19] Z. Li, Y. Gao, and Y. Chen, "HiFIND: A high-speed flow-level intrusion detection approach with DoS resiliency," *Comput. Netw.*, vol. 54, no. 8, pp. 1282–1299, 2010.
[20] W. Eddy, "TCP SYN flooding attacks and common mitigations," RFC 4987, Aug. 2007.
[21] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDOS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.
[22] F. A. El-Moussa, N. Linge, and M. Hope, "Active router approach to defeating denial-of-service attacks in networks," *IET Commun.*, vol. 1, no. 1, pp. 55–63, Feb. 2007.
[23] M. Nugraha, I. Paramita, A. Musa, D. Choi, and B. Cho, "Utilizing OpenFlow and sFlow to detect and mitigate SYN flooding attack," *J. Korea Multimedia Soc.*, vol. 17, no. 8, pp. 988–994, 2014.
[24] S. Manavi, S. Mohammadalian, N. I. Udzir, and A. Abdullah, "Secure model for virtualization layer in cloud infrastructure," *Int. J. Cyber Security Digit. Forensics*, vol. 1, no. 1, pp. 32–40, 2012.
[25] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks," in *Proc. 10th ACM Symp. Inf. Comput. Commun. Security*, Singapore, 2015, pp. 639–644.
[26] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect DoS flooding using software defined networking (SDN)," in *Proc. IEEE 35th Int. Conf. Distrib. Comput. Syst. Workshops*, Columbus, OH, USA, 2015, pp. 95–99.
[27] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "Sphinx: Detecting security attacks in software-defined networks," in *Proc. NDSS*, San Diego, CA, USA, 2015.
[28] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "An SDN-supported collaborative approach for DDoS flooding detection and containment," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Tampa, FL, USA, 2015, pp. 659–664.
[29] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, 2004.
[30] H. Hwang, A. Shingo, K. Yamamoto, K. Inoue, and M. Murata, "A new TCAM architecture for managing ACL in routers," *IEICE Trans. Commun.*, vol. 93, no. 11, pp. 3004–3012, 2010.
[31] Y. Gilad and A. Herzberg, *Lightweight Opportunistic Tunneling (LOT)*. Heidelberg, Germany: Springer, 2009, pp. 104–119.
[32] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's retransmission timer," IETF, Fremont, CA, USA, Tech. Rep. RFC 6298, 2011.
[33] S. Anandita, Y. Rosmansyah, B. Dabarsyah, and J. U. Choi, "Implementation of dendritic cell algorithm as an anomaly detection method for port scanning attack," in *Proc. Int. Conf. Inf. Technol. Syst. Innov. (ICITSI)*, Bandung, Indonesia, 2015, pp. 1–6.
[34] (Jan. 17, 2017). *OpenDayLight*. [Online]. Available: http://www.opendaylight.org
[35] (Jan. 17, 2017). *Mininet*. [Online]. Available: http://www.mininet.org/
[36] (Jan. 17, 2017). *CURL*. [Online]. Available: https://curl.haxx.se/
[37] (Jan. 17, 2017). *Hping*. [Online]. Available: http://www.hping.org/

Reza Mohammadi received the M.Sc. degree in computer networks from the Shiraz University of Technology in 2013, where he is currently pursuing the Ph.D. degree in computer networks. He has many publications in international conferences and journals regarding *Underwater Wireless Sensor Networks* (UWSNs). He currently focused on software defined networks (SDNs) as a new trend in computer networks. His major fields of interest are SDNs, heuristic algorithms, performance evaluation, UWSNs, ad hoc networks, and underground wireless sensor networks.

Reza Javidan received the M.Sc. degree in computer engineering (machine intelligence and robotics) and the Ph.D. degree in computer engineering (artificial intelligence) from Shiraz University in 1996 and 2007, respectively. He is currently a member of the Faculty and Lecturer with the Department of Computer Engineering and Information Technology, Shiraz University of Technology. He has many Ph.D. and M.Sc. students working together as a team. He has many publications in international conferences and journals regarding *Image Processing*, *Underwater Wireless Sensor Networks* (UWSNs), and *Software Defined Networks* (SDNs). His major fields of interest are artificial intelligence, image processing, UWSNs, SDNs, and sonar systems.



Mauro Conti (SM'14) received the Ph.D. degree from the Sapienza University of Rome, Italy, in 2009. He was a Post-Doctoral Researcher with Vrije Universiteit Amsterdam, The Netherlands. In 2011, he joined as an Assistant Professor with the University of Padua, Italy, where he became an Associate Professor in 2015. In 2017, he obtained the national habilitation as a Full Professor for Computer Science and Computer Engineering. He was a Visiting Researcher with George Mason University from 2008 to 2016, the University of California at Los Angeles in 2010, the University of California at Irvine from 2012 to 2014, Technische Universität Darmstadt in 2013, the University of Florida in 2015, and Florida International University from 2015 to 2016. His main research interests are in the area of security and privacy. He published over 170 papers in topmost international peer-reviewed journals and conference in the above areas. He was a recipient of the Marie Curie Fellowship by the European Commission in 2012 and the Fellowship by the German DAAD in 2013. He is an Associate Editor for several journals, including the IEEE COMMUNICATIONS SURVEYS & TUTORIALS and the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY. He was the Program Chair for TRUST 2015, ICISS 2016, and WiSec 2017, and the General Chair for SecureComm 2012 and ACM SACMAT 2013.