Computer Networks 173 (2020) 107223

Contents lists available at ScienceDirect

Computer Networks

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

Slow denial-of-service attacks on software defined networks

Túlio A. Pascoal [a,1,*], Iguatemi E. Fonseca [b], Vivek Nigam [b,c]

[a] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg
[b] Informatics Centre, Federal University of Paraiba, Joao Pessoa, Brazil
[c] fortiss GmbH, Munich, Germany

ARTICLE INFO

MSC:
00-01
99-00

Keywords:
Distributed denial-of-service attacks (DDoS)
Software defined networking (SDN)
Slow-rate attacks
High-rate attacks

ABSTRACT

Software Defined Networking (SDN) is a network paradigm that decouples the network's control plane, delegated to the SDN controller, from the data plane, delegated to SDN switches.

For increased efficiency, SDN switches use a high-performance Ternary Content-Addressable memory (TCAM) to install rules. However, due to the TCAM's high cost and power consumption, switches have a limited amount of TCAM memory. Consequently, a limited number of rules can be installed. This limitation has been exploited to carry out Distributed Denial of Service (DDoS) attacks, such as Saturation attacks, that generate large amounts of traffic. Inspired by slow application layer DDoS attacks, this paper presents and investigates DDoS attacks on SDN that do not require large amounts of traffic, thus bypassing existing defenses that are triggered by traffic volume.

In particular, we offer two slow attacks on SDN. The first attack, called Slow TCAM Exhaustion attack (Slow-TCAM), is able to consume all SDN switch's TCAM memory by forcing the installation of new forwarding rules and maintaining them indeterminately active, thus disallowing new rules to be installed to serve legitimate clients.

The second attack, called Slow Saturation attack, combines Slow-TCAM attack with a lower rate instance of the Saturation attack. A Slow Saturation attack is capable of denying service using a fraction of the traffic of typical Saturation attacks. Moreover, the Slow Saturation attack can also impact installed legitimate rules, thus causing a greater impact than the Slow-TCAM attack. In addition, it also affects the availability of other network's components, *e.g.*, switches, even the ones not being directly targeted by the attack, as has been proven by our experiments. We propose a number of variations of these attacks and demonstrate their effectiveness by means of an extensive experimental evaluation. The Slow-TCAM is able to deny service to legitimate clients requiring only 38 s and sending less than 40 packets per second without abruptly changing network resources, such as CPU and memory. Moreover, besides denying service as a Slow-TCAM attack, the Slow Saturation attack can also disrupt multiple SDN switches (not only the targeted ones) by sending a lower-rate traffic when compared to current known Saturation attacks.

1. Introduction

Software Defined Networking (SDN) is a network paradigm that provides easier network management. It allows better network configuration, performance, monitoring and automation. SDN facilitates network management by decoupling the data plane from the control plane. While enabling new features, SDN also has its own security concerns, which may be exploited by attackers to carry out Distributed Denial of Service Attacks (DDoS).

A logically centralized controller at the SDN control plane is responsible for taking the decision of where packets should be forwarded, *i.e.*, defining routing actions, while the task of forwarding packets is left to SDN switches at the data plane. Notably, the controller is a single point of failure.

A second vulnerability regards the limited amount of memory SDN switches have to store forwarding rules. Whenever a packet arrives to a switch, it searches whether there is a matching rule installed for that packet. This search is efficient because of dedicated memories called Ternary Content-Addressable Memory (TCAM), where forwarding rules are stored. If no rule is applicable, the switch queries the controller for a decision about current packet. The controller may install a new rule or drop the packet.

However, TCAM is expensive resource and has high power comsumption [1]. Therefore, most current commercial SDN switches have a limited TCAM space [1–3] and can store only a limited number of rules

* Corresponding author.
E-mail addresses: tulio.pascoal@uni.lu (T.A. Pascoal), iguatemi@ci.ufpb.br (I.E. Fonseca), nigam@fortiss.org (V. Nigam).
[1] This work was done while at the University of Paraiba.

https://doi.org/10.1016/j.comnet.2020.107223
Received 14 October 2019; Received in revised form 15 February 2020; Accepted 15 March 2020
Available online 19 March 2020

(typically 1500 to 8000 rules) [1,3–6]. The limited amount of TCAM memory in SDN-switches has been exploited to carry out DDoS attacks, called Saturation attacks [1,3,5,7,8,8–14]. A Saturation attack forces the target SDN switch to install a great number of new rules, thus consuming switch's TCAM capacity. This not only disallows the installation of new flows for legitimate clients, but also causes the network controller to crash because of increased traffic between the switch and the controller.

A number of defenses have been proposed to mitigate Saturation attacks. The main underlying assumption of these measures is that attackers will send unique packets in a very high rate by, for example, spoofing IPs. This causes abrupt changes of network parameters, which easily trigger defense countermeasures.

However, the assumption that DDoS attacks can be launched by necessarily generating high traffic is not necessarily true. Indeed as witnessed by the class of slow-rate Application Layer DDoS attacks, such as Slowloris [15], attackers can deny service of a web-server or a VoIP server by sending a very low rate of requests to the target server [16–18]. Attackers can also carry out low-rate attacks using computationally weak devices [19,20] and exploit new vulnerabilities on application layer protocols in order to evade detection mechanisms, *e.g.*, SlowNext [21].

Inspired by slow-rate Application DDoS Attacks [15,16,20,22], this paper investigate slow DDoS attacks on SDN, which do not require very large amount of traffic. Since existing defense are triggered by monitoring traffic volume, these slow attacks can bypass such defenses.

We offer and investigate the following types of Slow attacks:

- A *Slow TCAM Exhaustion attack* (Slow-TCAM) denies service by sending unique-crafted packets to provoke a new flow rule installation in a target SDN switch. The attack follows by recruiting and coordinating a large enough botnet (typically of 1500 to 4000 bots). Each bot sends, in a disguised manner, an unique packet causing the target switch to install a forwarding rule thus exhausting the switch's TCAM and denying service to *new legitimate clients*. Finally, in order to avoid that the installed rules are dropped due to rule timeouts, the botnet periodically resends packets, thus reactivating each installed rule. As our experiments demonstrate, the Slow-TCAM attack can be carried out by generating a very low amount of traffic, up to 4 packets per second as opposed to more than 1000 packets per second in existing Saturation attacks [3,5,13,23]. This means that it can bypass existing defenses which assume high-rate attacks.
- A *Slow Saturation attack* combines the Slow-TCAM attack with a lower rate Saturation attack. It is able to deny service using a fraction of traffic of an usual Saturation Attack. Whereas Saturation attacks require traffic of more than 200 packets per second [24], the Slow Saturation affects service using only 100 packets per second. In contrast to the Slow-TCAM, which does not affect already installed rules, the Slow Saturation attacks also leads to the dropping of already installed rules by forcing their respective timeout expiration. This means that an attacker can also deny service to *clients using flows installed before the attack (existing rules in TCAM)*.
  We investigate two variants of the Slow Saturation attack. The first variation generates continuous traffic, while the second variation generates bursts of high traffic. We also measure the effects of the attack in the SDN network as a whole, *i.e.*, the effect of the attack in other network's components, but the targeted switch. Given the fact that the attack stress the network's controller, it also interferes and hampers the expected behavior of other devices linked to the affected controller.

We experimentally evaluate and discuss the features and the impact of these attacks assuming a reasonable system and threat model.

Paper Structure: After briefly revisiting the OpenFlow protocol used by SDN and its main workflows in Section 2, an introduction to existing attacks on SDN is presented in Section 3. Posteriorly, in Section 4, we introduce our proposed slow attacks, namely the Slow-TCAM Exhaustion and the Slow Saturation attacks, followed by Section 5, which extensively evaluates and discusses about existing defense mechanisms and possible counter-measures for the novel presented attacks. Then in Section 6, we describe our experimental setup, results and discussion. Finally, related work is discussed in Section 7 before concluding the work in Section 8.

Conference Paper: This extends and improves on the conference paper [25], which introduces the Slow-TCAM attack. For this manuscript, we have extended and improved the experimental results in [25] by using a more realistic client traffic model for the Slow-TCAM experiments. Furthermore, we propose and investigate a novel attack, called Slow Saturation attack, which was not present in the conference paper. We also added new experimental evaluation set up by assembling an new network topology consisting of more than one switch in order to measure the effects of the attack on other network's assets. Finally, we evaluated existing defense approaches against SDN-aimed attacks and their effectiveness against our proposed attacks. In addition, we also discuss possible countermeasures.

2. SDN fundamentals

While we assume that the reader is familiar with the OpenFlow protocol [26], the widely used SDN southbound protocol, which enables a controller to coordinate SDN switches forwarding process, we review some of the messages exchanged between a SDN switch and the controller. The installation of rules in SDN are based on two approaches: *proactive* and *reactive*. In the former, the SDN switch starts with some pre-defined rules, while in the latter new rules are installed according to queries (PACKET_IN messages) sent by the switch, in a dynamic way.

In the reactive mode, which is the most common approach [27], whenever a packet is received by a SDN switch, it checks whether there is a matching forwarding rule. If so, it applies the rule defined for that packet. However, if no rule is applicable, a *table-miss* is occurring and then the switch exchanges the following messages with the controller:

$$\text{Switch} \rightarrow \text{Controller} \quad : \quad \text{PACKET\_IN}(packet'sheader)$$
$$\text{Controller} \rightarrow \text{Switch} \quad : \quad \text{FLOW\_MOD}(idle\_timeout)$$

The PACKET_IN message contains the incoming packet's header, with its *buffer_id*, in_*port*, *payload* and other main information. FLOW_MOD messages contain the rule that should be installed by the switch and is always sent by the network's controller. The controller can specify a rule idle timeout. Given a rule timeout of T_o, a rule is uninstalled by the switch if it is not triggered for T_o time units. The use of timeouts is a mechanism to remove less used rules, thus freeing TCAM memory for other rules to be installed. Typically, the timeout T_o is a value between 9 and 11 s [10].

Once the message FLOW_MOD is received by the switch, it checks whether there is enough space in its TCAM memory for installing a new rule. If this is the case, the rule is installed and the packet is forwarded using it. Otherwise, the switch drops the packet and informs the controller that its TCAM memory is full (see Fig. 1) by sending the following message:

$$\text{Switch(dropspacket)} \rightarrow \text{Controller} \quad : \quad \text{TABLE\_FULL}$$

In case a PACKET_IN is sent to controller and the switch's *incoming buffer* is full[2], the whole packet content will be sent instead of packet's header only. This fact stresses the switch-to-controller communication channel, which can lead to denial of service, considering that a flooding of *new unique packets* can overload the controller and as a result compromise entire network proper functioning.

$$\text{Switch} \rightarrow \text{Controller} \quad : \quad \text{PACKET\_IN}(packet'swholecontent)$$
$$\text{Controller} \rightarrow \text{Switch} \quad : \quad \text{FLOW\_MOD}(idle\_timeout)$$

[2] Not to confuse the incoming packet buffer, which stores packets, with the TCAM that stores rules.

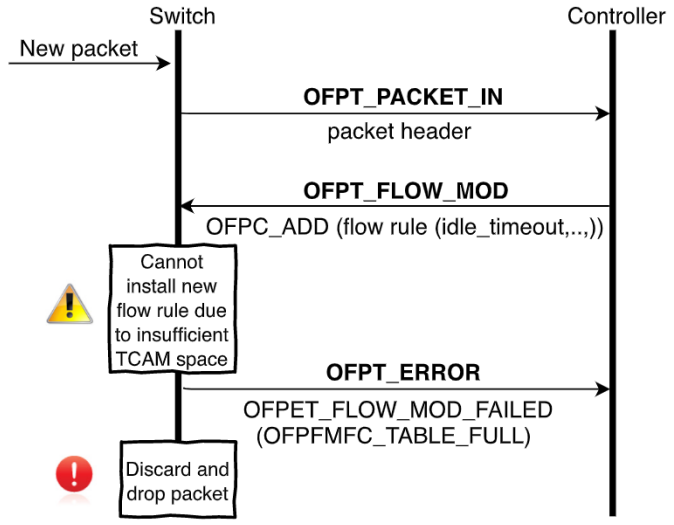


Fig. 1. Switch-controller behavior when TCAM table is full.

Additionally, always a rule is uninstalled due to its inactive time, the switch sends a OFPT_FLOW_REMOVED (OFPRR_IDLE_TIMEOUT) to the controller.

Switch → Controller : FLOW_REMOVED($IDLE\_TIMEOUT$)

Similarly, when a rule is deleted by a controller decision (by sending a OFPC_DELETE message), the rule is uninstalled in TCAM and the switch sends a OFPT_FLOW_REMOVED (OFPRR_DELETE) message to the network controller.

Controller → Switch : OFPC_DELETE(*rule*)
Switch → Controller : FLOW_REMOVED($DELETE$)

Moreover, we point out that the communication between a SDN switch and the controller is expensive as it builds a secure channel for their communication. Typically, the OpenFlow protocol is implemented on top of Secure Sockets Layer (SSL) or Transport Layer Security (TLS). Therefore, a defense should avoid switch-to-controller communication overhead.

Lastly, as mentioned in the OpenFlow specification [28], in case of connection's interruptions between a switch and controller, there are two possible modes in which the switch enters:

- **Fail Secure Mode:** In this mode, all following packets and messages destined to the controller are dropped. The remaining installed flow rules in the switch TCAM continue to expire according to their timeouts.
- **Fail Standalone Mode:** In this model, the switch starts to work as a legacy Ethernet switch or router, processing packets by its reserved port. This mode is only applicable for hybrid switches, which are switches that jointly support normal Ethernet switching and OpenFlow operations. This is not the case of virtual switches, *e.g.*, the Open vSwitch, which is one of the most popular implementations of OpenFlow switches [29].

3. Attacking software defined networks

SDN has been increasingly adopted by network managers and industry thanks to its better programmability and automation. SDN has also been used for denial of service attacks detection [30]. However, one of its main concerns is about its own security and vulnerabilities. For example, its centralized controller is a potential exploit vector that can compromise the entire network, forcing them to be well-configured in order to achieve scalability, high responsiveness and security. Another SDN's drawback is that SDN switches have a limited TCAM space that can be steadily overflowed.

To the best of our knowledge, with the exception of the Slow-TCAM attack [25], existing denial of service attacks on SDN are based on the sending of flooding traffic. Therefore, exploiting the limited TCAM space availability in SDN switches and overloading its switch-to-controller communication (thus, stressing SDN data and control planes). Recall that SDN switches are able to store between 1500 and 8000 rules [1,3–6] and also have a finite packet's incoming buffer. Therefore, there are attacks on SDN which attempt to (1) consume the TCAM memory of switches and (2) overload the switch-to-controller communication, these attacks are known by Saturation attacks [3,5,13,23].

These attacks are carried out by sending unique packets at a high traffic rate (typically greater than 500 unique packets per second), normally by spoofing IPs. This flooding behavior triggers the installation of a huge number of rules in switches' TCAM. Once the TCAM is exhausted, the targeted switch starts to drop packets leading to denial of service. Moreover, a Saturation attack goes even further by sending unique packets at an even higher rate consuming not only the switch's TCAM memory, but also the switch's incoming buffer. The switch, then, starts sending to the controller the whole packet instead of the packet's header only. Thus, overloading the controller and leading it to crash. Eventually, a controller crash or malfunctioning affects the whole network.

For purpose of illustration and comparison, we carried out Saturation attacks with different traffic rates on a switch with capacity of installing 1500 rules. (The detailed experimental setup is described in Section 6). Table 1 summarizes the different attacks. For lower rates (100 packets per second), the service is available to all legitimate clients. In addition, [24] shows that the performance of most commercial switches, such as Pica8 and HP Procurve, only starts to deteriorate when receiving 200 unique packets per second. This is in accordance to our own experiments depicted in Table 1.

As described in Section 7, defenses for the TCAM Exhaustion attack and the Saturation attack assume that the attacker necessarily sends a great number of unique packets, *i.e.*, flooding the switch. Existing defenses monitor parameters that are affected when receiving a large number of unique packets, *e.g.*, rule installation rate, CPU and memory consumption and number of unpaired rules, for example.

However, this assumption is not necessarily true. We identify that SDN is vulnerable to two new attacks which use lower traffic rates: the Slow TCAM Exhaustion attack (Slow-TCAM) and the Slow Saturation attack. In the following sections, we describe these two attacks. Then in Section 5, we discuss how one could mitigate theses attack and in Section 6, we demonstrate by simulation the effectiveness of these attacks.

4. Slow DoS attacks on SDN

4.1. System and threat model

Our system and threat model consider a scenario where there are a number of legitimate client hosts. In addition, there is also an attacker that has or controls some other hosts (bots), which are all linked to the targeted switch. We also assume that the OpenFlow protocol is operated

Table 1
Saturation attack: availability, controller's CPU and memory usage when under attacks of different intensities.

Attack traffic	Regular controller		
	Availability (%)	CPU (%)	Memory usage (MB)
Continuous (100 pkts/s)	100	24.7	40.5
Continuous (200 pkts/s)	24.2	45.4	214.1
Continuous (400 pkts/s)	13.4	50.7	266.1
Continuous (600 pkts/s)	10.8	50.6	269.6
Continuous (800 pkts/s)	10.3	51.8	265.6

Table 2
Slow DoS attacks' parameters and their respective description.

Attack parameter	Description
T_o	The inferred flow rules' timeout of the network.
N_r	The inferred number of the targeted TCAM table of the switch.
N_{ps}	The rate in which the attacker is sending new unique packets.
N_{pr}	The rate in which the attacker is re-sending packets.

using the most commonly applied rule installation mode, the reactive mode [27].

Apart from being connected to the targeted switch, the attacker does not need any prior knowledge about network topology or switches and controller configurations. The attacker can first infer the SDN timeout using existing SDN timeout probing approaches [31,32], such as SDN SCANNER [4], and then crafts its packets using the gathered information accordingly. Our system model assumptions have been replicated from previous existing works on attacking SDN networks [3,5,13,23,25,33].

A Slow-TCAM attack final goal is to overflow the TCAM table of the target switches by crafting and sending the packets in a disguised way. Once the switches' flow tables are full, no new rule can be installed and new packets will be dropped by the switches, *i.e.*, disrupting the SDN data plane. On the other hand, the Slow Saturation apart from sending Slow-TCAM traffic, it also disrupts the SDN control plane by launching saturation traffic at certain intervals, thus stressing and consequently interrupting switch-to-controller connection. When such communication failures take place, we assume that the switch will enter the **Fail Secure Mode** according to OpenFlow specifications and described at the end of Section 2.

4.1.1. Slow attacks' principles

Our proposed attacks are composed of two phases, namely (i) *Probing phase* and (2) *Launching phase*. The former allows the attacker to infer and identify network's settings, such as T_o, N_r, N_{ps} and N_{pr} described in Table 2. These parameters enable attackers to decide the optimal number of packets and minimal packet sending rate in order to trigger new flow rules' installation and refresh their timeouts, thus keeping the target switch's flow table continuously full. Once gathering network's configuration an adversary is ready to craft packet's payload and rate accordingly, therefore starting the latter phase that consists of sending the hushed traffic.

4.1.2. Probing phase

The main rationale behind the *Probing phase* is based on the fact that in any SDN network when a new packet that causes a *table-miss* in the switch arrives, it perceives a longer response time. This happens because as introduced in Section 2, the switch needs to query the network's controller for its decision on the rules that will be created to that specific packet. All next packets that match that rule, will experience a shorter responsive time because now the switch only needs to apply the forwarding rule set by the controller before.

Notably, an adversary can smartly create and send packets to the network, and monitor the packet's response time. As a result, by applying statistical tests on the collected data, such as *t-tests* [4], the attacker can infer the network's timeouts. Similarly, attacker can also craft packets in order to find packet's match fields that trigger flow rule installation [33]. As the particularities and intuition of these approaches are not novel, we opt to refer the readers to [4,31–33] for a more detailed explanation and reasoning of such approaches in SDN.

4.1.3. Launching phase

Once gathering network's information from previous phase, the attacker is now able to define the N_{ps} and N_{pr} metric rates, such that they not only cause minimal impact on the network's bandwidth, but also continuously keep malicious rules installed in the targeted flow table [25]. After setting all parameters for the attack, the attacker plots the attack by crafting and sending the packets accordingly.

As each attack has its particular features, we explain their respective Launching phase separately in the following sections.

4.2. Slow TCAM exhaustion attack (slow-TCAM)

The *Launching phase* of the Slow TCAM attack is carried out as by following the steps:

1. Recruit a large enough number of bots, typically a number a bit greater than a half the rule capacity of the target switch (N_r). A number between 1500 and 4000 is enough considering existing SDN-enabled commercial switches [1–3]. This is feasible as the attacker can recruit a botnet using standard methods, *e.g.*, phishing or purchasing such botnet service. For example, the *"0x-booter"* is a type of Crime-as-A-Service that offers DoS-for-hire consisted of 20,000 bots [34]. Notice that the attacker is not spoofing IP addresses.
2. Each bot sends a unique packet to the target switch. Whenever the switch receives the first packet, a new rule is installed. Moreover, since there is no IP spoofing, two flows, an incoming and outgoing flow, are eventually installed.
3. The unique packet generation rate (N_{ps}) is controlled so that the rate that new rules are installed is not too high. The N_{ps} rate defines how fast an attacker is able to overflow the target switch flow table. In our experiments, the attacker generates a traffic of up to 40 packets per second, while typical flooding and Saturation attacks generate a traffic greater than 1000 packets per second [3,5,13,23].
4. Meanwhile, each bot keeps re-sending, at a N_{pr} low rate, packets to the switch within its rule idle timeout T_o. Recall that the T_o can be inferred by the attacker by trial and error using SDN SCANNER [4]. Consequently, once the switch's TCAM is full of bot rules and none of their rule is uninstalled, the TCAM will be always full. Thus, not being able to manage the installation of new rules.
5. Finally, by keeping the botnet behaving coordinated, the attacker is able to keep the target switch in the TABLE_FULL state indefinitely.

When a Slow-TCAM attack is carried out, the controller and the switch operate normally, but they are forced to serve only the *flow rules installed before the attack and the flow rules installed by the attacker*, thus denying service to new legitimate clients.

4.2.1. Slow-TCAM by example

In order to better understand how the Slow-TCAM attack works, we illustrate it by example. Assume a target switch with $N_r = 5$ and the rule timeout $T_o = 10$. Moreover assume that the attacker has already recruited a large enough botnet. In this case, a botnet of 3 bots is enough.

Assume for simplicity that a rule is represented by the following tuple:

$$\langle \text{rule}_i, n \rangle$$

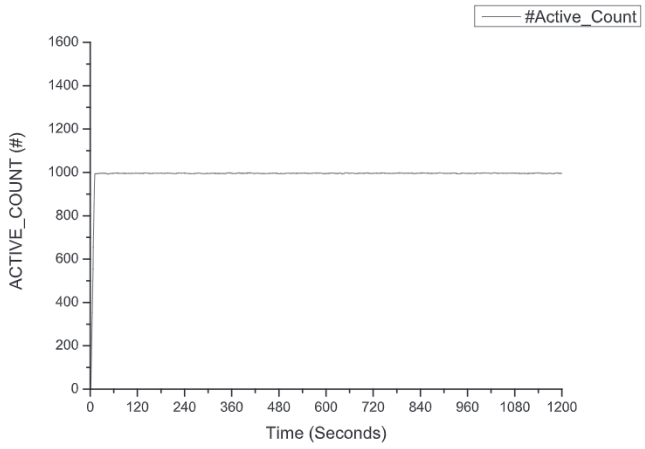
where rule_i stands for the rule identifier and n is the remaining seconds for timeout of the rule to be activated.

Once a controller is connected to the switch, a special rule representing their communication is created. The resulting switch flow table state is as follows:

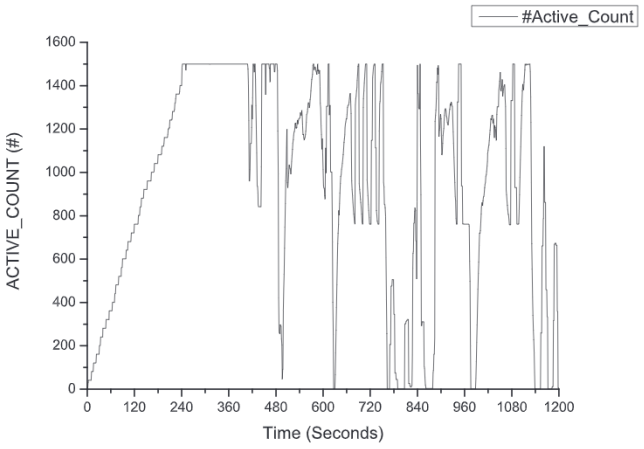
$$\text{Flow\_Table} = \langle \text{controller\_rule}, \infty \rangle$$

Now, suppose a bot attacker sends a packet to the switch. After the OpenFlow messages are exchanged and receiving the correspondent FLOW_MOD message, the switch installs the new flow rules. Moreover, since the bot is not spoofing its IP, the queried service will respond leading to the installation of an outgoing rule. Therefore, the switch's flow table has the following state:

$$\text{Flow\_Table} = \langle \text{controller\_rule}_1, \infty \rangle, \langle \text{inc\_rule}_1, 10 \rangle, \langle \text{outg\_rule}_1, 10 \rangle$$



(a) TCAM table during a Saturation attack.



(b) TCAM table during a Slow Saturation attack.

Fig. 2. TCAM table consumption: Comparison between a Saturation attack with intensity of 100 packets per second and a Slow Saturation attack (*i.e.*, a Slow-TCAM attack with intensity of 5.8 packets per second and a Saturation attack with intensity of 100 packets per second). The target SDN switch is capable of storing 1500 rules and the rule timeout is set to 10 s.

Assume two seconds elapse. Another bot also send its packet causing additional incoming (inc_rule) and outgoing (outg_rule) rules to be installed. At this point, the switch's TCAM is full with the following rules:

$$\text{Flow\_Table} = \langle \text{controller\_rule}_1, \infty \rangle, \langle \text{inc\_rule}_1, 8 \rangle, \langle \text{outg\_rule}_1, 8 \rangle, \langle \text{inc\_rule}_2, 10 \rangle, \langle \text{outg\_rule}_2, 10 \rangle$$

Any new flows cannot be installed as switch's TCAM does not have more room left to store new ones. If this happens, a TABLE_FULL message will be sent by the switch to the controller and the packets trying to be installed are dropped from the network, leading to the denial of service.

Additionally, during 7 s, the attacker's bots do not need to send any packets as the timeout of their rules will not expire before that. The state of the switch after 7 s is as follows:

$$\text{Flow\_Table} = \langle \text{controller\_rule}_1, \infty \rangle, \langle \text{inc\_rule}_1, 1 \rangle, \langle \text{outg\_rule}_1, 1 \rangle, \langle \text{inc\_rule}_2, 3 \rangle, \langle \text{outg\_rule}_2, 3 \rangle$$

At this moment, in order to keep their rules installed, the bots that generated the rules inc_rule_1 and inc_rule_2 will re-send a packet so to renew the timeout of their rules, respectively, as illustrated below:

$$\text{Flow\_Table} = \langle \text{controller\_rule}_1, \infty \rangle, \langle \text{inc\_rule}_1, 10 \rangle, \langle \text{outg\_rule}_1, 10 \rangle, \langle \text{inc\_rule}_2, 10 \rangle, \langle \text{outg\_rule}_2, 10 \rangle$$

Notice that the bots do not need to resend the same exact packet, but any packet that activates these rules will do the work. Indeed, in order to avoid simple monitoring defenses, the bot may choose packets with different payload or requests.

4.3. Slow saturation attack

The *Launching phase* of the Slow Saturation attack is carried out by combining a Slow-TCAM attack and a Saturation attack with a lower traffic rate. The Slow Saturation combines the power of the Slow-TCAM and the Saturation attack, *i.e.*, it denies service by using less traffic than usual Saturation attacks, and it allows to deny service to clients that are served by rules previously installed in the SDN switch.

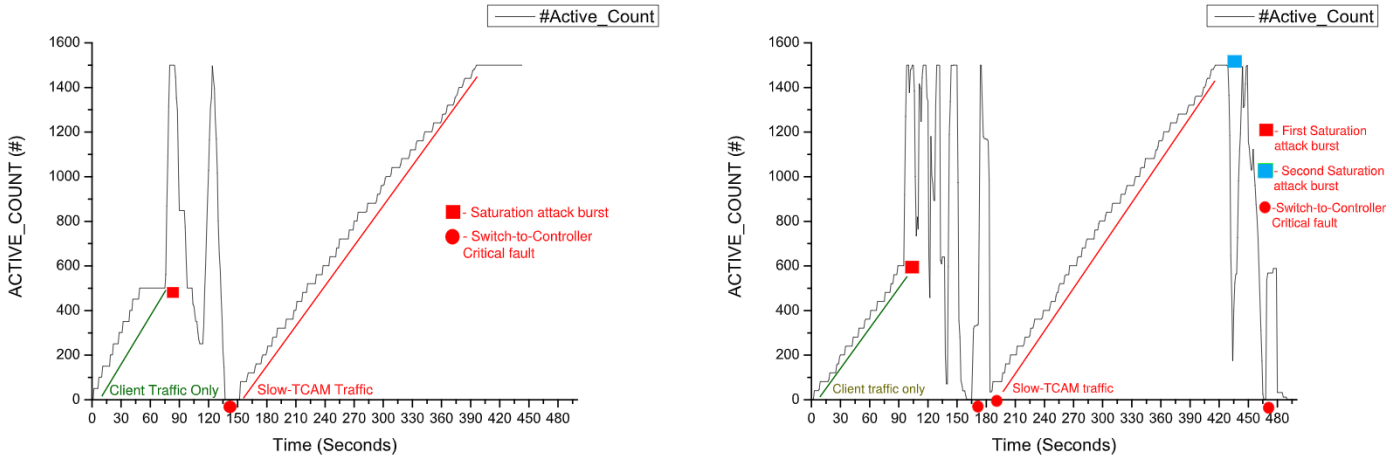
We illustrate and discuss these features in the Fig. 2a and b.

- **Lower Traffic Rate:** Fig. 2 illustrates the difference between a Saturation and a Slow Saturation attack using the same traffic rate of 100 packets per second, on a target SDN switch with $N_r = 1,500$ rules and $T_o = 10$ s. As shown in Fig. 2a, the Saturation attack is not able to successfully deny service (as also shown in Table 1). It does occupy 1000 of the 1500 of the available table rules.
  On the other hand, as shown in Fig. 2b, the Slow Saturation successfully disrupts service by using the same traffic rate *Nps* of 100 packets per second. This is because the Slow-TCAM component of the attack is able to keep their rules alive and installed for longer periods, in conjunction with the rules from the saturation component of the attack. It is important noticing that saturation traffic is not able to renew their rule's timeouts as it always send unique packets.
- **Dropping Previously Installed Rules:** Furthermore, as also illustrated by Fig. 2b, the Slow Saturation attack is able to disrupt the target switch, thus leading to a dropping of active rules. Around 480 s after the attack starts, the switch begins to suffer from communication failures with the controller, thus dropping stored rules (due to their timeout expirations - recall the fail secure mode definition from Section 2). Note that there are moments during the experiment in which there is no rule installed. This is due to the fact of longer (above 10 s) periods of switch-to-controller communication failure, leading to a complete dropping of installed rules.
  In addition, there are situations in which the switch keeps around 800 to 1000 rules installed, the key insight of this evidence is that while the rules installed by the saturation component of the attack are dropped by the rule timeout, the rules installed by the Slow-TCAM component of the attack are kept alive more easily (as they are rejuvenated by the attacker). Hence, this effect turns possible for the attack to consume all the switch's TCAM at certain points in time.

A Slow Saturation attack affects, therefore, not only new legitimate clients (by disrupting SDN data plane with the Slow-TCAM component of the attack), but also legitimate clients that are served by previous existing SDN rules (by the Saturation component of the attack, which slightly and punctually stress the SDN control plane, causing switch-to-controller communication failures that leads to flow rule's timeout expiration).

We consider two variants of this attack:

- **Continuous:** In the Continuous Slow Saturation attack, both the Slow-TCAM and the Saturation attack are carried out simultaneously and during the duration of the attack, *e.g.*, Fig. 2b.



(a) Slow Saturation with one Burst. (b) Slow Saturation with two Bursts.

Fig. 3. Slow Saturation: Number of installed rules in the target SDN switch during a Slow Saturation attack with punctual burst(s) of 1000 packets per second and Slow-TCAM intensity of 5.8 unique packets per second.

- **Bursts:** In the Burst Slow Saturation attack, the Slow-TCAM attack is carried out during the whole duration of the attack, but the Saturation attack alternates between two active periods (the *bursts*), when both the Slow-TCAM and the Saturation traffics are sent; and *sleeping* periods, in which the attack is only sending Slow-TCAM traffic.

The goal of the Burst Slow Saturation attack is to reduce the average rate of packets, hence increasing the chances of bypassing defenses that monitor the average traffic load.

Fig. 3a and b illustrate how Burst Slow Saturation attacks deny service. In this experiment, we first populate the switch's TCAM table with 500 flow rules installed by legitimate clients. Then, we stars two different variations of the Saturation attack, one with only one burst, and another one with two bursts.

In Fig. 3a, the switch's TCAM is first populated with 500 flow rules installed by legitimate clients. At the 75th second, an Saturation attack with only one burst starts (red square in the figure). From that moment on, several switch-to-controller connection interruptions take place due to the stress caused by the attack. These interruptions lead to the uninstalling of flow rules due to their timeout expiration. For example, at the 135th second, all the installed flow rules have been expired (red circle in the figure). Thus, denying service to all previously served legitimate client's flow rules. Finally, only at the 150th the communication is reinitiated, enabling an attacker to launch a Slow-TCAM traffic, therefore being able to load the TCAM table with malicious rules.

Similarly, Fig. 3b illustrates a sequence of two bursts of Saturation attacks. The first burst was able to disrupt switch-to-controller communication until the 185th second. From that moment, a Slow-TCAM traffic launched by an attacker could reach the TCAM table full capacity at the 427th second. Then, at the 430th a second saturation burst is started (represented by a blue square in the figure) leading to the removal of all currently installed rules at 485th. At this moment, an attacker could have started launching a Slow-TCAM traffic to fulfill the switch flow table once more.

In addition, it is important noticing that an attacker can freely choose when to start or release bursts of saturation traffic. This can be perceived and used as a means to not only drop all current installed rules in a target switch, but also to ensure that when the switch-to-controller communication is back alive, the most number of new installed rules come from the Slow-TCAM's traffic part of the attack.

In summary, this experiment shows the practicability of a Saturation attack when not only dropping previously installed rules (with the saturation part of the attack) but also when overflowing switch's TCAM table capacity with its Slow-TCAM component.

5. Mitigating TCAM-aimed attacks

The existing approaches against TCAM-aimed attacks imply that attackers send unique packets at a very high rate, mixing IP spoofing and flooding techniques. This behavior causes abruptly changes in the network that easily triggers countermeasures. The main approaches to avoid these attack are based on: (1) setting rule timeouts (*idle timeout*) embedded in the OpenFlow, which is the standard mechanism for removing obsolete rules; (2) monitoring unpaired rules, *i.e.*, rules for which there is an incoming flow, but no outgoing flow. When spoofing IPs, attacker rules are not able to have outgoing flow rules as the spoofed IP may be not achievable; (3) monitoring of rule installation rate, which is used to trigger countermeasures when the rate of new rules are high, such a case of a Saturation attack, and (4) CPU and memory monitoring of SDN controllers and switches, in which when under stress they increasingly start to consume more computational resources in order to deal with the augmented traffic.

Considering the novelty of the Slow-TCAM attack, to the best of our knowledge, only *SelectIve deFense for TCAM* (SIFT) [25] has been proposed as a defense against Slow-TCAM attacks until nowadays. Similarly, such as a novel attack, no defense has been proposed to directly mitigate the hereby presented Slow Saturation attack. Table 3 summarizes the main current defenses used to mitigate Saturation attacks. We discuss why they are not capable of mitigating TCAM-aimed attacks.

Although SIFT mitigates the Slow-TCAM attack, it is not able to mitigate alone the Slow Saturation attack. The reason is the relatively high rate of packets generated by the Slow Saturation attack. We replayed all experiments presented in this work and we observed that SIFT cannot handle high-rate traffic. We believe that in order to mitigate the Slow Saturation attack, SIFT would have to work with other parameters, such as switch's CPU/memory consumption. There are also some approaches and alternative defense mechanisms that could be implemented in the network along with SIFT to effectively cease a Slow Saturation attack. It is also possible to merge some existing approaches that mitigate Saturation attacks (such as the ones presented in Table 3) with SIFT.

6. Experimental results

We implemented the Slow-TCAM and Slow Saturation attacks and carried out a number of experiments. We used two virtual machines,

Table 3
Existing defense approaches against TCAM-aimed attacks.

Approach	Short description	Vulnerability against Slow-TCAM and Slow Saturation attacks
Yu et al. [35]; Katta et al. [2]	Propose architectural changes to TCAM and *OpenFlow* protocol in order to increase rule storage.	Their main goal is to enhance SDN perfomace, nevertheless their changes will only demand a larger *botnet* by the attacker. Their approaches are not also able to mitigate the Slow Saturation attack as they do not protect the switch-to-controller communication.
AVANT GUARD [23]	Validates TCP Handshake before installing rules.	As Slow-TCAM bots have legitimate IP, connection and so forth, they trespass the TCP Handshake testing. Furthermore, it has a vulnerability in its *cache module*, which can be exploited by attackers, leading to denial of service, as presented by [14].
SPHINX [3]	Monitors rule installing and creation rate.	As Slow-TCAM sends packets in a slow and "legitimate" way in addition to being configurable, the attacker can circumvent the limiars set in the defense. As during a Slow Saturation attack the adversary can control the rate and the interval of Saturation attack, the attacker can bypass the triggers of this defense.
Shin et al. [4]; Kandoi et al. [5]	Applies OpenFlow mechanisms, such as *Optimal Timeout* and textitFlow Aggregation in a dinamic way.	The traffic sent by a Slow-TCAM attack is very similar to client traffic, then hampering and increasing false positive matches during the detection. It is true that *Flow Aggregation* can maximize rule storage, however they need to be used very carefully as they can facilitate the incoming of malicious traffic to the network. Aside that, it is not capable of mitigating Saturation attacks, as mentioned in [36], consequently, they cannot mitigate the Slow Saturation attack.
FLOOD GUARD [13]	Monitors switch's CPU and memory usage.	Slow-TCAM has a little impact in these parameters while attacking, hence the defense is not to detect and start countermeasures against the attack. The attacker can perform a minimalist Slow Saturation attack, reducing the traffic rate and interval time. Thus, avoiding to consume switch's resources, trespassing the defense.
Wang et al. [8]	Applies traffic analysis along with neural networks and entropy techniques in order to decide about rule installation.	Slow-TCAM has a traffic similar to legitimate clients, therefore the defenses suffer from false positive detection. As the Slow Saturation mixes slow a flooding traffic it can difficult even more the accuracy of the defenses' analysis. In addition, it also has *cache* vulnerabilities as found in AVANT GUARD [14].
Jinan S. [9]	Applies a peer support strategy along switches in the network, where they share their idle storage TCAM space.	This defense is not able to detect the presence of the Slow-TCAM attack but it can retard the beginning of the denial of service, forcing the attacker to recruit a larger *botnet*. Furthermore, it is only concerned about the TCAM overflow, therefore, no countermeasures is taken against the flooding traffic of the Slow Saturation attack.
Xu et al. [37]	Uses the *Tocken Bucket* model to control PACKET_IN messages sent to the controller.	As Slow-TCAM attack traffic is slow and its rules are always reinstalled, it does not generate a high traffic of PACKET_INs in the network, consequently less switch-to-controller communication are performed. In addition, the Slow Saturation only send high traffic in certain intervals of time. Thus, the defense is not capable of detecting and mitigating the attack as it will never be triggered.
Shang et al. [27]	Uses a *cache system* to keep rules not found in the flow table (*"table-miss"* entries), thus sending PACKET_INs to the controller in a controlled way.	Slow-TCAM does not send a high traffic, running silently. Thus, not activating the approach countermeasures. In relation to the Slow Saturation attack, the adversary can model the attack in a way that it will not surpass traffic rate specified by the defense.
Ma et al. [38]	Propose a Moving Target Defense (MTD) approach, which posses a pool of interchangeable controllers that are shifted dynamically according to their stress level (due to flooding attacks). They also offer MTD approaches for reducing the success rate of attackers scanning the SDN network.	Their approach can only mitigates flooding-based attack, *i.e.*, Saturation attacks. Therefore, they cannot mitigate the Slow-TCAM component of our proposed attack. Although their MTD scheme of dynamic delays hinders the chances of attackers fingerprinting the network easier, an attacker still can carry out a Slow-TCAM or Slow Saturation attack using smaller interval periods.
SDN-Anti-DDoS [39]	Its mitigation module is responsible for detecting anomalies in PACKET_IN messages, such as unpaired rules (derived from IP spoofing-based attacks) as well as the rate of new incoming PACKET_INs.	As this approach is threshold-based, an attacker can adapt the traffic rates of both components of the Slow Saturation attack. In addition, the Slow-TCAM traffic component of the attack not only does not spoof IPs, but also generates a low rate packet sending, thus surpassing mitigation techniques proposed by this defense.
ReCON [40]	Applies MTD techniques in order to minimize and re-use under-utilized critical SDN network's resources, *i.e.*, OpenFlow Agent (OFA).	Similarly, this approach does not monitor TCAM table occupancy, as a consequence, it is not able to treat the Slow-TCAM component of our attack.

one running Mininet [41] along with Open vSwitch 2.5.0 [29], which are a well-known network emulator and open-source virtual switch, respectively. Another virtual machine executed the SDN controller Ryu [42] using OpenFlow 1.3 [26]. For the attack generation we used Hping [43] and Scapy python library [44].

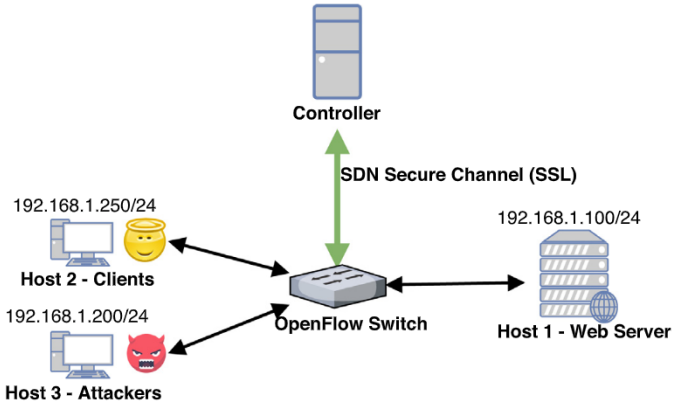
The Mininet machine was a Ubuntu 14.04 LTS, Intel i7-5500U CPU@2,40 GHz with 3 GB of RAM memory, while the Ryu machine was a Ubuntu 16.04.1 LTS, Intel i7-5500U CPU@2,40 GHz with 1 GB of RAM memory. The host machine was a Windows 10 - 64 bit, Intel i7-5500U CPU@2,40GHz with 8 GB of RAM memory.

In order to avoid resource limitation interference, we carried out the Saturation and Slow Saturation attacks on a Dell PowerEdge T430 server with dual 8-core Xeon E5-2620 CPU@2,10 GHz and 64 GB RAM, running the same virtual machines images (Mininet and Ryu) informed above. We set the SDN switch rule capacity to 1500 rules with rule timeout T_o of 10 s as recommended in the literature [10].

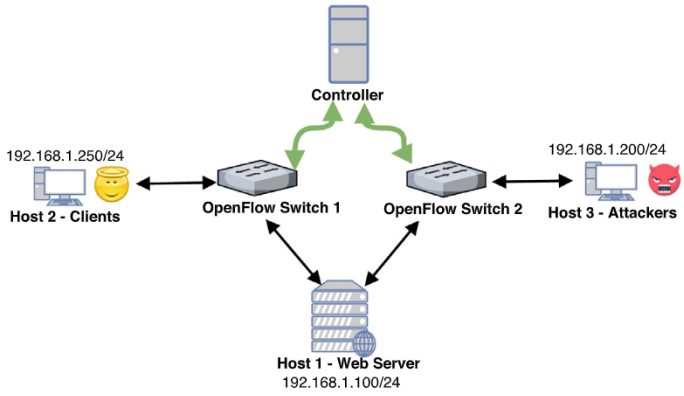
Fig. 4a shows the set-up of our experiments for the Slow-TCAM and Slow Saturation first experiments. We also performed additional experiments with the Slow Saturation attack in a scenario where there are two switches in the SDN topology. The goal of the latter experiments is to evaluate how a Slow Saturation attack affects other existing switches int the network, even when a non targeted switch belongs to the topology, but yet is not targeted by the attacker.

6.1. Slow-TCAM results

Legitimate client traffic (Host 2) consisted of 375 unique connections, which means the installation of 750 rules (incoming and outgoing rules) in a switch, *i.e.*, half the switch rule capacity. We implemented the Slow-TCAM attack where the attacker (Host 3) possesses a botnet with more than 760 bots and no more than 800 bots. Both legitimate and attacker's bots accessed the web-server (Host 1).

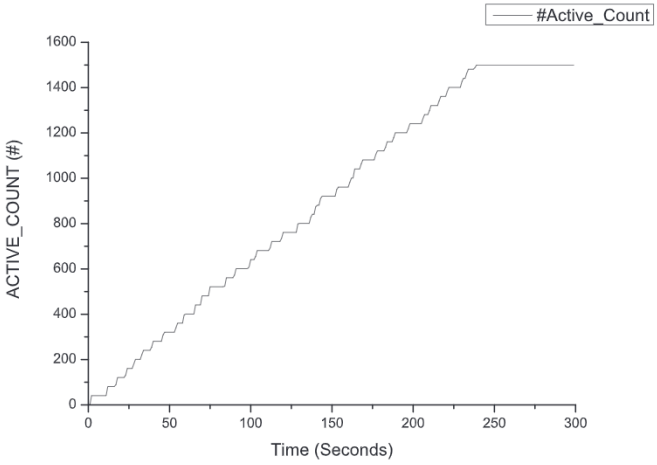


(a) SDN topology with one switch.

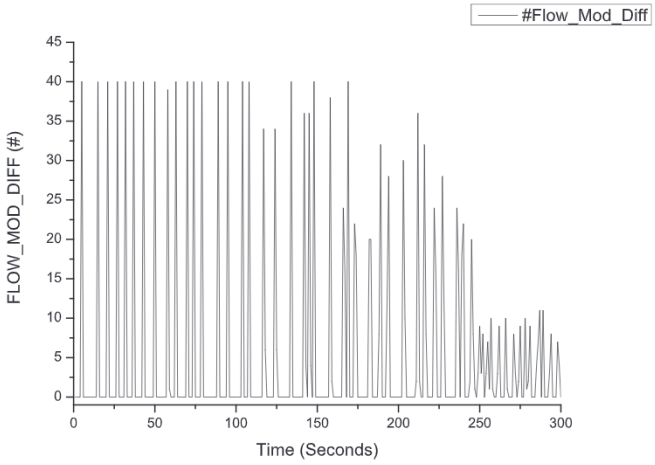


(b) SDN topology with two switches.

Fig. 4. Experimental set-ups.



(a) Number of installed rules in the target SDN switch during the attack.



(b) Number of FLOW_MOD messages sent per second by the controller during the attack.

Fig. 5. Slow-TCAM attack with intensity of 5.8 unique packets per second.

Table 4
Slow-TCAM: attack rate, availability, time to service, time to DoS, CPU and memory consumption. The value on Availability corresponds to the number of clients that are able to obtain a response after the attacker has carried out the attack and occupied all the TCAM memory.

Average attack rate	Availability (%)	TTS (ms)	Time to DoS (s)	CPU (%)	Memory usage (MB)
No Attack	100	12.6	–	1.1	36.9
3.2 unique pkts/s	0.0	∞	478	2.5	42.3
4.6 unique pkts/s	0.0	∞	324	3.8	43.0
5.8 unique pkts/s	0.0	∞	258	4.7	42.3
9.2 unique pkts/s	0.0	∞	162	4.9	42.5
13.6 unique pkts/s	0.0	∞	110	6.4	42.2
15.6 unique pkts/s	0.0	∞	96	7.2	41.9
23.6 unique pkts/s	0.0	∞	63	10.4	41.8
39.5 unique pkts/s	0.0	∞	38	10.9	42.3

In contrast to [25], we reproduced a new and more realistic legitimate client traffic by using the tool Scapy [44]. Clients now are able to renew their flow rules by accessing the web-server more than once, thus better simulating a real web-client traffic. We simulated 200 clients that are randomly created within periods of 5 and 10 s, sending 10 requests to the web-server (on Host 1) each.

Table 4 summarizes our experimental results for the Slow-TCAM attack. We measured the legitimate client availability after the attacker has occupied all the TCAM memory, time to service (TTS), the time for the attacker to deny service, the controller's average CPU and memory usage.

Our experiments show that Slow-TCAM attack can be effective in denying service to legitimate clients accessing the network using a SDN switch. We carried out a number of experiments with different attack intensities from 3.2 unique packets per second to 39.5 unique packets per second. In comparison typical flooding attacks has a rule installation rate greater than 500 unique packets per second [3,5,13,23]. Once the attacker successfully occupied all the TCAM memory, every one of its

Table 5
Slow saturation: attack rate, availability, CPU and memory usage when under an attack of a variety of intensity. The slow part of the attack had a intensity of 5.8 unique packets per second.

Attack rate	Regular controller (no defense approaches embedded)		
	Availability (%)	CPU (%)	Memory usage (MB)
Continuous (100 pkts/s)	12.8	36.1	147.5
Continuous (200 pkts/s)	2.8	45.4	157.5
Continuous (400 pkts/s)	3.6	46.3	165.9
Continuous (800 pkts/s)	2.7	47.3	176.3
Bursts (100 pkts/s during 20 s every 20 s)	27.5	17.6	123.5
Bursts (100 pkts/s during 40 s every 40 s)	29.9	17.7	129.2
Bursts (100 pkts/s during 60 s every 60 s)	17.1	16.4	121.3
Bursts (200 pkts/s during 20 s every 20 s)	16.3	25.0	135.1
Bursts (200 pkts/s during 40 s every 40 s)	18.8	26.0	130.8
Bursts (200 pkts/s during 60 s every 60 s)	15.8	29.7	122.7
Bursts (400 pkts/s during 20 s every 20 s)	12.2	32.0	141.0
Bursts (400 pkts/s during 40 s every 40 s)	15.7	33.3	122.7
Bursts (400 pkts/s during 60 s every 60 s)	13.2	29.7	124.3
Bursts (800 pkts/s during 20 s every 20 s)	8.52	35.6	151.1
Bursts (800 pkts/s during 40 s every 40 s)	5.4	33.6	159.4
Bursts (800 pkts/s during 60 s every 60 s)	10.4	29.6	129.0

bots sends with periodicity of 3 s a packet to keep its corresponding rule active in the SDN switch.

Fig. 5a illustrates the TCAM consumption by the Slow-TCAM attack with intensity of 5.8 unique packets per second. It takes a bit more than 4 min to occupy all the rule capacity by installing 1500 rules. The remaining scenarios with different attack intensities had the same behavior. For our slowest attack, with intensity of 3.2 unique packets per second, the attacker can deny service even more silently in around 8 min with practically no impact on the controller's CPU usage. On the other hand, the attacker can also deny service more quickly in, 38 s, by carrying out a Slow-TCAM attack with intensity of 39.5 unique packets per second with still a very low impact on the controller's CPU usage. Notice that the attacker is able to keep the rules installed in the switch by avoiding their timeout to be fired. This can be observed by the fact that no rules are uninstalled. Once all 1500 rules are installed, there is no more room for new rules thus denying service to legitimate clients which require new rules to be installed.

We measured the number of FLOW_MOD messages sent by the controller (illustrated in Fig. 5b). As the attack is slow, it causes the controller to send a low amount of FLOW_MOD messages (less than 40 per second) and once the TCAM is occupied the number of FLOW_MOD messages reduces even further. For example, it were generated less than 1750 FLOW_MOD messages in 300 s of Slow-TCAM attack. In comparison, the Saturation attacks described in Section 3, 6000 messages were generated in 35 s. Moreover, the rate of rule installed when carrying out a Slow-TCAM attack is 10 times lower than when carrying out a Saturation attack. Notice as well that this number can also be reduced if the attacker is willing to carry out an attack with an even lower rate, *e.g.*, 3.2 unique packets per second. Finally, as the attacker is not spoofing IPs, all rules installed in the switch to handle his packets are paired, *i.e.*, have an incoming and outgoing rules.

More recently, a similar low-rate attack against SDN have been proposed in [33]. They evaluated their attack considering same topology as in our work, with the exception of that they used a commercial hardware-based switch EdgeCore AS4610-54T with a 1800 flow rule capacity, *i.e.*, a real SDN testbed was considered. As expected, they have found same findings and drawn same conclusions as encountered in our work. For this reason, we are confident that the Slow-TCAM attack has same efficiency when launched against TCAM-embedded switches.

6.2. Slow saturation experimental results

We carried out experiments with both versions of the Slow Saturation Attack, namely Continuous and Bursts. Since these attacks generate a greater amount of traffic, we used for these experiments the DELL PowerEdge server introduced in this section in order to avoid negative interference by lack of resources.

Table 5 summarizes our Slow Saturation experiments. The Continuous Slow Saturation attack was promptly able to considerably affect service (12.8%) when sending only 100 packets per second. In comparison, Saturation attacks with the same intensity was not able to deny service (see Table 1). Continuous Saturation attacks with greater intensity also denied service (web-server had a lower than 4% service availability). On the other hand, the controller's memory and CPU consumption during a Continuous Slow Saturation attack were similar to Saturation attacks.

On the other side, Burst Slow Saturation attacks considerably disrupted service availability, albeit less than similar-intensity Continuous Slow Saturation attacks. Note that during burst attacks, there is no saturation traffic for some intervals of time (20, 40 and 60 s - depending on the type of the experiment), thus making room for some legitimate client rules. However, the impact of Burst Slow Saturation to controller's CPU and memory was considerably lower, thus being more likely to bypass defenses that monitor such parameters.

6.2.1. Additional effects of the slow saturation attack

In order to measure the effects of the Slow Saturation attack in the network as a whole, we created a topology in which two switches are connected to the same network's controller and clients and attackers use different switches to access the web-server on Host 1 (see Fig. 4b). We measured the parameters for 200 clients (sending 10 requests each one of them) using the non targeted switch to access the web-server on Host 1. Note that the only attacked switch is the Switch 2, while Switch 1 only receives legitimate client traffic. However, as shown in Table 6, even not going through the targeted switch, the availability of the web-server for the legitimate clients decreased (availability varied from 78.9% to 90.4%). In addition, some of client packets has not been successfully answered by the non attacked switch due to the network instability caused by the attack.

This behavior is explained by the fact that the saturation component of the Slow Saturation attack is capable to disrupt the network's controller, consequently, both switch (even the not attacked one) suffers the denial of service effects. Therefore, a Slow Saturation attack is also able to disrupt other network's device components, *i.e.*, switches, and then the entire network. Thus, an attacker is capable of disrupting a target SDN-network even not having full knowledge about its topology.

These experimental results demonstrate that the attacker has a number of options when carrying out Slow Saturation attack, each with its own features: the type of Saturation attack (Continuous or Burst), its

Table 6
Slow saturation: attack rate, total sent vs. total unanswered packets by legitimate clients and availability when under an attack of a variety of intensity. The slow part of the attack had a intensity of 5.8 unique packets per second.

Attack rate	Non targeted switch (no defense approaches embedded)	
	Total sent / Unanswered packets	Availability (%)
Continuous (200 pkts/s)	1984 / 397	79.9
Continuous (500 pkts/s)	2000 / 373	81.4
Continuous (800 pkts/s)	1976 / 415	78.9
Bursts (200 pkts/s during 60 s every 60 s)	1976 / 189	90.4
Bursts (800 pkts/s during 20 s every 15 s)	1990 / 266	86.7
Bursts (800 pkts/s during 60 s every 60 s)	1989 / 230	88.4

intensity, intervals between bursts, and the intensity of the Slow-TCAM attack.

6.3. Possible countermeasures

As it have been shown in [25], the defense SIFT is capable of mitigating Slow-TCAM attacks (which only targets the SDN data plane). Nevertheless, given the number of its features, it is challenging to mitigate a Slow Saturation attack. For example, SIFT is not able to mitigate Slow Saturation attacks because it is not able to handle the attack's saturation component, which affects not only the SDN control plane, but also its data plane.

Therefore, as there is already a variety of defenses that mitigate saturation attacks, we speculate that combining SIFT with existing defenses for Saturation attacks (such the ones in Table 3) may be effective. In addition, we also think that applying Moving Target Defense (MTD)-based approaches [38,40,45–47] would offer a effective way of mitigating Slow Saturation attacks. The main rationale behind a MTD approach is that it randomly changes some networks settings, parameters or even topology, therefore, dynamically changing the attacker surface. Thus, imposing some advantages on the defender side.

For example, Ma et al. [38] applies a MTD technique that randomly changes packets' delays so that attackers cannot correctly infer SDN's timeouts. Besides that, they keep pool of healthy SDN controllers that replace controllers in critical stages (*i.e.*, being attacked). On the other hand, Gillani et al. [40] presented ReCON, a defense mechanism that offers a control agent that dynamically manages under-utilized resources within the SDN network's control plane and then increases resources capabilities of control plane's devices in critical situation, thus, minimizing damages on the SDN control plane. Moreover, Kampanakis et al. [45] offer IP randomization techniques in order to frustrate attackers launching scanning attacks. Similarly, Jafarian et al. [46] offer the same feature, however focusing on creating a more effective approach in terms of speed. Equivalently, MacFarland and Shue [47] not only aims at randomizing IP addresses, but also MAC address. Hence, decreasing chances of successful attacks, as attackers have to constantly find their proper targets.

Therefore, we argue that by applying MTD-based methods, such as replacing faulty controllers with healthy ones, which are kept spare in a controller pool reserve, and randomly setting timeouts of flow rules is enough to mitigate the attacks proposed in this work. As a result, we plan to combine our previous work [25] with several anti-saturation attack defenses and MTD techniques, and experimentally evaluate it considering a variety of scenarios to effectively measure the effectiveness of such hybrid defense mechanism. However, a more detailed investigation is left for future work.

7. Related work

For the best use of SDN programmability, SDN switches utilize a special type of memory: *Ternary Content Addressable Memory* (TCAM). A memory which has a faster and wide query power than *Content Addressable Memory* (CAM) and *Random Access Memory* (RAM). However, all its benefits come with high prices and power consumption. Current SDN switches have approximately 1 Mbit to 2 Mbit of TCAM, where each 1 Mbit chip costs U$ 350 and consumes 15 Watt/1MBit in average [1]. Due to these factors, SDN switches have a limited TCAM and can store only a narrow number of flow rules [1–3], typically between 1500 and 8000 rules.

Some proposals [2,35] suggest modifications to the OpenFlow protocol used in SDN and in the structure of TCAM memory in order to improve memory management. Likely, a tagging approach is proposed by Banerjje et al. [48]. In their mechanism, TCAM entries utilize less bits to represent flow rules. For example, the tagged flows use only 24 bits while regular flow entries have 356 bits. Thus, increasing TCAM storing capacity.

Kandoi and Antikainen [5] and Shin and Gu [4] comment and propose the possibility of using Optimal Timeout technique to flow rules, seeking for better TCAM usage. In other words, their mechanism install rules setting their *idle_timeout* according to flow traffic of the network. This avoids that rules keep installed without being used, or uninstalled while being used. Consequently, improving TCAM usage. They also offer the Flow Aggregation mechanism which is a technique that generates more general rules defining macroflows, instead of using more specific rules, defining microflows. This strategy can increase the quantity of rules installed in a TCAM, but at the expense of leaving the network more vulnerable to other attacks, *e.g.*, Get-Flooding, allowing malicious traffic to use the network. Moreover, as pointed out by Wang et al. [13], Flow Aggregation is not capable of mitigating Saturation attacks such as the ones proposed by Curtis et al. [36].

The main goal of these proposals is to enhance SDN general performance, whereas we expose the TCAM limited space SDN vulnerability as a mean to deny its service. This means that with these approaches the attacker may need to require some more bots to deny service.

The aim of Saturation attacks is to force the switch to constantly install new rules. In the literature, this is accomplished by sending a high rate of unique packets, *e.g.*, using spoofing and sending UDP packets [3,5,7]. Furthermore, the Saturation attack [1,8,10–14] has as main objective to crash the controller by sending a large amount of traffic to a SDN switch occupying its incoming buffer. *This causes the switch to send to the controller the whole packet instead of only sending the packet header.*

Dhawan et al. [3] propose the detection of DoS attacks by monitoring the rate of rule creation by the SDN controller. If this rate surpasses a certain threshold, then mitigation actions are taken. Similarly, Xu et al. [37] use token bucket model in order to limit the attack rate. On the other hand, Shang et al. [27] propose FLOOD DEFENDER. It remodels the data plane flow table in order to maximize switches TCAM usage and utilizes deviation mechanisms to detour packets in order to save controller resources. Since the Slow-TCAM attack can be configured to set a particular rate of rule creation, these defenses are not effective in mitigating the Slow-TCAM attack. It is true that FLOOD DEFENDER

allows to store more flow rules; however the attacker will only need a larger, but still relatively small botnet.

The strategy AVANT GUARD [23] detects when a TCP-handshake is completed before creating rules in the network. It has been recently shown [14] that this defense is vulnerable to a modification of the Saturation attack capable to consume all AVANT GUARD's resources. As in a Slow-TCAM attack, the attacker's bots complete the TCP-handshake, AVANT GUARD's strategy cannot detect Slow-TCAM attacks.

Wang et al. [13] propose to monitor switch's buffer, controller's CPU and memory usage to mitigate Saturation attacks. As the Slow-TCAM attack has little impact to these parameters, the defense proposed by Wang et al. is not effective in detecting Slow-TCAM attacks.

Similarly, Wang et al. [8] recently proposed to use neural networks, entropy, and more sophisticated traffic analysis methods to help the controller to make the decision of adding a rule or not. However, their approach suffers the same deficiencies as the AVANT GUARD by storing a cache of rules. Moreover, since Slow-TCAM is a new attack and has similar characteristic of legitimate clients, it is not clear whether the trained neural network proposed by Wang et al. will be able to identify a Slow-TCAM attack.

Yuan et al. [9] proposes a peer support strategy in which SDN switches share their unused TCAM memory space among them when they are reaching its TCAM limit. This is done by installing flow rules in the attacked switch (they keep a reserved space in TCAM for that) in order to divert flows to other peer switches according to parameters such as TCAM usage, how close to the attacked switch, how busy is a switch, and how a switch connects to other switches. However they can only retard the attack and has the problem that when the majority or many switches are full they will divert traffic between them ending up in a vicious loop.

More recently, Moving Target Defense (MTD) approaches have been proposed as a means to mitigating and avoiding DDoS attacks on SDN networks [38,40,45–47]. The main goal of MTD is to frequently shift the attack surface, so that demanding higher efforts from attackers. For example, randomly setting different timeout values to several flow rules [45,46], keeping a pool of healthy controllers to be replaced with faulty ones [38], and dynamically changing not only network's controller IP addresses, but also MAC addresses [47]. Consequently, increasing overall SDN security.

One of most disguised DDoS attacks available are the so called Low-Rate Application Layer DDoS attacks, such as Slowloris [15]. Attackers can deny service of a web-server or a VoIP server by sending a very low rate of requests to the target server [16–18]. Attackers can also carry out Low-Rate attacks using rather weak devices such as mobile phones [19,20] and exploit new vulnerabilities on application layer protocols in order to evade detection mechanisms, *e.g.*, SlowNext [21].

After the proposal of the Slow-TCAM by Pascoal et al. [25], another independent and similar slow attack was proposed in [33]. Both approach aims to firstly infer some SDN configuration settings, *e.g.*, flow rule timeouts for assembling the strategy of the attack. Posteriorly, launching the attack in a disguised way, avoiding abrupt network's parameters changes and then evading existing countermeasures. The results in [33] shows the effectiveness of the attack in hardware-based SDN switches, which validates and consents with our results presented in [25] and in this paper. Based on our and their findings, we claim that our proposed attacks are feasible not only in laboratory-crafted scenarios, but also in real-word scenarios considering commercial SDN switches. In addition, we have extended both works [25,33] by offering a novel Slow Saturation attack and evaluating the attacks' effectiveness under other network's topology, such as in the multiple-switches scenario.

To the best of our knowledge, due to the novelty of the Slow-TCAM and Slow Saturation attacks, there are no proposed mitigation in the literature for them. Nevertheless, we have provided possible countermeasures and guidance in Section 6.3.

8. Conclusion

This paper proposes and investigates slow attacks on Software Defined Networks, namely the Slow TCAM Exhaustion attack (Slow-TCAM) and the Slow Saturation attack. These attacks exploit the fact that SDN switches can only store a limited number of forwarding rules. We demonstrate their effectiveness by performing an extensive experimental evaluation.

We show that Slow-TCAM attack can deny service by sending packets at a very low rate (3.2 packets per second), in contrast to existing attacks. It is also able to evade existing defense mechanisms due to its disguised traffic rate and similarity to legitimate clients behavior. The second proposed attack, Slow Saturation, is able to deny service using only a fraction of the traffic required by existing Saturation attacks. Additionally, it is also able to deny service to previous client (installed rules). Moreover, Slow Saturation attacks can affect and disrupt the SDN network as a whole, given the fact that it stress the network's controller, it also disrupt the well functioning of other network's components connected to it. As a result, it affects other network's components, *e.g.*, non-targeted switches, as shown by our results.

Since existing defenses for DDoS attacks on SDN assume that an attack necessarily floods the network, these defenses are not suitable for mitigating such our offered slow attacks. We believe that such defense can be constructed by combining SIFT [25] with existing approaches for the mitigation of Saturation attacks along with MTD-based techniques.

As future work, we aim at deploying our attacks considering a testbed with SDN commercial switches, and assembling an efficient defense mechanism able to tackle the particularities and intrinsicalities of such Slow DoS attacks, by using the presented guidelines.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Túlio A. Pascoal: Conceptualization, Methodology, Software, Validation, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Iguatemi E. Fonseca:** Funding acquisition, Project administration, Supervision, Writing - review & editing. **Vivek Nigam:** Supervision, Investigation, Writing - review & editing, Visualization, Conceptualization, Formal analysis, Validation, Funding acquisition.

Acknowledgments

This work was supported by the Fonds National de la Recherche Luxembourg (FNR) through PEARL grant FNR/P14/8149128, and partially supported by CNPq projects 425870/2016-2, 303909/2018-8 and FAPESP project 15/24516-1. This project received funding from the EUs Horizon 2020 research and innovation programme under grant agreement No. 830892.

References

[1] K. Kannan, S. Banerjee, Compact TCAM: flow entry compaction in TCAM for power aware SDN, in: International Conference on Distributed Computing and Networking, Springer, 2013, pp. 439–444.
[2] N. Katta, O. Alipourfard, J. Rexford, D. Walker, Infinite cacheflow in software-defined networks, in: Proceedings of the Third Workshop on Hot Topics in Software Defined Networking, ACM, 2014, pp. 175–180.
[3] M. Dhawan, R. Poddar, K. Mahajan, V. Mann, SPHINX: detecting security attacks in software-defined networks., NDSS, 2015.
[4] S. Shin, 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, ACM, 2013, pp. 165–166.

[5] R. Kandoi, M. Antikainen, Denial-of-service attacks in OpenFlow SDN networks, in: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), IEEE, 2015, pp. 1322–1326.
[6] A. Vishnoi, R. Poddar, V. Mann, S. Bhattacharya, Effective switch memory management in OpenFlownetworks, in: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, ACM, 2014, pp. 177–188.
[7] R. Klöti, V. Kotronis, P. Smith, OpenFlow: a security analysis, in: 2013 21st IEEE International Conference on Network Protocols (ICNP), IEEE, 2013, pp. 1–6.
[8] M. Wang, H. Zhou, J. Chen, B. Tong, An approach for protecting the OpenFlow switch from the saturation attack (2016).
[9] B. Yuan, D. Zou, S. Yu, H. Jin, W. Qiang, J. Shen, Defending against flow table overloading attack in software-defined networks, IEEE Trans. Serv. Comput. 12 (2) (2016) 231–246.
[10] A. Zarek, Y. Ganjali, D. Lie, OpenFlow Timeouts Demystified, Univ. of Toronto, Toronto, Ontario, Canada, 2012.
[11] S. Hong, L. Xu, H. Wang, G. Gu, Poisoning network visibility in software-defined networks: new attacks and countermeasures, NDSS, 2015.
[12] X. Dong, H. Lin, R. Tan, R.K. Iyer, Z. Kalbarczyk, Software-defined networking for smart grid resilience: opportunities and challenges, in: Proceedings of the 1st ACM Workshop on Cyber-Physical System Security, ACM, 2015, pp. 61–68.
[13] H. Wang, L. Xu, G. Gu, Floodguard: a dos attack prevention extension in software-defined networks, in: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE, 2015, pp. 239–250.
[14] M. Ambrosin, M. Conti, F. De Gaspari, R. Poovendran, Lineswitch: efficiently managing switch flow in software-defined networking while effectively tackling dos attacks, in: Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ACM, 2015, pp. 639–644.
[15] Slowloris DoS application. Available at: https://github.com/llaera/slowloris.pl.
[16] Y.G. Dantas, V. Nigam, I.E. Fonseca, A selective defense for application layer DDoS attacks, in: JISIC 2014, 2014, pp. 75–82.
[17] M.O.O. Lemos, Y.G. Dantas, I. Fonseca, V. Nigam, G. Sampaio, A selective defense for mitigating coordinated call attacks, 34th Brazilian Symposium on Computer Networks and Distributed Systems (SBRC), 2016.
[18] Y.G. Dantas, M.O.O. Lemos, I. Fonseca, V. Nigam, Formal specification and verification of a selective defense for TDoS attacks, 11th International Workshop on Rewriting Logic and Its Applications (WRLA), 2016.
[19] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, Mobile executions of slow dos attacks, Logic J. IGPL 24 (1) (2016) 54–57.
[20] E. Cambiaso, G. Papaleo, M. Aiello, Slowdroid: turning a smartphone into a mobile attack vector, in: Future Internet of Things and Cloud (FiCloud), 2014 International Conference on, IEEE, 2014, pp. 405–410.
[21] E. Cambiaso, G. Papaleo, G. Chiola, M. Aiello, Designing and modeling the slow next dos attack, in: International Joint Conference, Springer, 2015, pp. 249–259.
[22] M.O. Lemos, Y.G. Dantas, I.E. Fonseca, V. Nigam, On the accuracy of formal verification of selective defenses for TDoS attacks, J. Logical Algebraic Methods Program. 94 (2018) 45–67.
[23] S. Shin, V. Yegneswaran, P. Porras, G. Gu, Avant-guard: scalable and vigilant switch flow management in software-defined networks, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, 2013, pp. 413–424.
[24] A. Wang, Y. Guo, F. Hao, T. Lakshman, S. Chen, Scotch: elastically scaling up SDN control-plane using vswitch based overlay, in: Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies, ACM, 2014, pp. 403–414.
[25] T.A. Pascoal, Y.G. Dantas, I.E. Fonseca, V. Nigam, Slow TCAM exhaustion DDoS attack, in: IFIP International Conference on ICT Systems Security and Privacy Protection, Springer, 2017, pp. 17–31.
[26] OpenFlow, Open Networking Foundation (ONF), https://www.opennetworking.org/. Accessed in: January 2nd, 2018.
[27] G. Shang, P. Zhe, X. Bin, H. Aiqun, R. Kui, FloodDefender: protecting data and control plane resources under SDN-aimed DoS attacks, in: INFOCOM 2017-IEEE Conference on Computer Communications, IEEE, IEEE, 2017, pp. 1–9.
[28] OpenFlow switch specification. Available at: https://www.opennetworking.org.
[29] OpenVSwitch, 2018. http://openvswitch.org/. Accessed in: April 22th, 2018.
[30] S. Rowshanrad, S. Namvarasl, V. Abdi, M. Hajizadeh, M. Keshtgary, A survey on SDN, the future of networking, J. Adv. Comput. Sci. Technol. 3 (2) (2014) 232.
[31] H. Cui, G.O. Karame, F. Klaedtke, R. Bifulco, On the fingerprinting of software-defined networks, IEEE Trans. Inf. Forensics Secur. 11 (10) (2016) 2160–2173.
[32] J. Leng, Y. Zhou, J. Zhang, C. Hu, An inference attack model for flow table capacity and usage: exploiting the vulnerability of flow table overflow in software-defined network, arXiv:1504.03095 (2015).
[33] J. Cao, M. Xu, Q. Li, K. Sun, Y. Yang, J. Zheng, Disrupting SDN via the data plane: a low-rate flow table overflow attack, in: International Conference on Security and Privacy in Communication Systems, Springer, 2017, pp. 356–376.
[34] DDoS-for-Hire Service Powered by Bushido Botnet. Available at: https://www.fortinet.com/blog/threat-research/ddos-for-hire-service-powered-by-bushido-botnet-.html.
[35] M. Yu, J. Rexford, M.J. Freedman, J. Wang, Scalable flow-based networking with DIFANE, ACM SIGCOMM Comput. Commun. Rev. 40 (4) (2010) 351–362.
[36] A.R. Curtis, W. Kim, P. Yalagandula, Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection, in: INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 1629–1637.
[37] T. Xu, D. Gao, P. Dong, C.H. Foh, H. Zhang, Mitigating the table-overflow attack in software-defined networking, IEEE Trans. Netw. Serv. Manag. 14 (4) (2017) 1086–1097.
[38] D. Ma, Z. Xu, D. Lin, Defending blind DDoS attack on SDN based on moving target defense, in: International Conference on Security and Privacy in Communication Networks, Springer, 2014, pp. 463–480.
[39] Y. Cui, L. Yan, S. Li, H. Xing, W. Pan, J. Zhu, X. Zheng, Sd-anti-DDoS: fast and efficient DDoS defense in software-defined networks, J. Netw. Comput. Appl. 68 (2016) 65–79.
[40] F. Gillani, E. Al-Shaer, Q. Duan, In-design resilient SDN control plane and elastic forwarding against aggressive DDoS attacks, in: Proceedings of the 5th ACM Workshop on Moving Target Defense, ACM, 2018, pp. 80–89.
[41] Mininet, 2018. http://www.mininet.org/. Accessed in: December 3rd, 2017.
[42] Ryu, 2018. https://osrg.github.io/ryu/. Accessed in: October 15th, 2017.
[43] Hping. Available at: http://www.hping.org/.
[44] Scapy project. Available at: http://www.secdev.org/projects/scapy/.
[45] P. Kampanakis, H. Perros, T. Beyene, SDN-based solutions for moving target defense network protection, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, IEEE, 2014, pp. 1–6.
[46] J.H. Jafarian, E. Al-Shaer, Q. Duan, An effective address mutation approach for disrupting reconnaissance attacks, IEEE Trans. Inf. Forensics Secur. 10 (12) (2015) 2562–2577.
[47] D.C. MacFarland, C.A. Shue, The SDN shuffle: creating a moving-target defense using host-based software-defined networking, in: Proceedings of the Second ACM Workshop on Moving Target Defense, ACM, 2015, pp. 37–41.
[48] S. Banerjee, K. Kannan, Tag-in-tag: efficient flow table management in SDN switches, in: Network and Service Management (CNSM), 2014 10th International Conference on, IEEE, 2014, pp. 109–117.

Túlio A. Pascoal is currently a Ph.D. candidate at the Interdisciplinary Centre for Security, Reliability and Trust at the University of Luxembourg. He received his B.Sc. and M.Sc. in Computer Science from the Federal University of Paraíba (UFPB), Brazil, in 2016 and 2018, respectively. He accomplished a 16-month scholarship during his bachelor's at the University of Toronto, Canada from 2013 to 2015. His current research interests include areas in Computer Networks, Privacy, and Security, focusing on topics as mitigation of Denial of Service attacks, techniques for the improvement of Software Defined Networks, and privacy-preserving methods for collaborative systems.

Iguatemi E. Fonseca is currently an associate professor at the Informatics Center of the Federal University of Paraíba. He received the electronics engineering degree from the Federal University of Campina Grande (UFCG), Brazil, in 1999, and the M.Sc. and Ph.D. degrees from the State University of Campinas (Unicamp), Brazil, in 2001 and 2005, respectively. His current research interests include areas in Communications Networks, working on topics as identification of traffic in computer networks, techniques for identification and mitigation of DDoS attacks, algorithms and protocols in Industrial Wireless Sensor Networks, WDM and elastic optical networks with QoS requirements.

Vivek Nigam is a research scientist leading the safety and security project group at fortiss. He is a Tenured Associate Professor at the Federal University of Paraíba, Brazil since 2012. He completed his Ph.D. studies in 2009 at the cole Polytechnique, France on the topic of Computational Logic. He has contributed with new foundational results, methods, and techniques in several topics of computer science, including formal methods, programming languages, protocol security, and denial-of-service attacks. He has collaborated in projects together with leading universities and with industries from several domains, such as avionics, automotive and networking.