

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/320092071>

A holistic approach to mitigating DoS attacks in SDN networks

Article in *International Journal of Network Management* · September 2017

DOI: 10.1002/nem.1996

CITATIONS

0

READS

44

2 authors:



Dridi Lobna

École de Technologie Supérieure

2 PUBLICATIONS 3 CITATIONS

[SEE PROFILE](#)



Mohamed Faten Zhani

École de Technologie Supérieure

47 PUBLICATIONS 1,114 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:




WAT-SDN [View project](#)

All content following this page was uploaded by [Mohamed Faten Zhani](#) on 21 October 2017.

The user has requested enhancement of the downloaded file.

A holistic approach to mitigating DoS attacks in SDN networks

Lobna Dridi | Mohamed Faten Zhani 

École de Technologie Supérieure (ÉTS),
Montreal, Quebec, Canada

Correspondence

Mohamed Faten Zhani, Department of
Software and IT Engineering, École de
Technologie Supérieure (ÉTS), Montreal,
Quebec H3C 1K3, Canada.
Email: mfzhani@etsmtl.ca

Summary

Software-defined networking (SDN) has recently emerged as a new networking technology offering an unprecedented programmability that allows network operators to dynamically manage their infrastructures. However, despite these benefits, deny-of-service (DoS) attacks are considered a major threat to such networks, as they can easily overload the SDN controller and flood switch forwarding tables, resulting in a critical degradation of the network performance. To address this issue, we propose SDN-Guard, a novel holistic approach to protect SDN networks against DoS attacks. Software-defined networking-Guard leverages an intrusion detection system (IDS) to detect potential DoS attacks and then efficiently mitigate their impact by dynamically (1) rerouting malicious traffic, (2) adjusting flow time-outs, and (3) aggregating flow rules. This paper extends our previous work by proposing solutions to minimize the switch-to-IDS traffic without impacting the IDS accuracy. We hence propose to use sampling techniques and devise an integer linear program to find the optimal placement for the IDS and to determine the switches that should mirror the flows towards it so as to minimize network bandwidth consumption. Extensive experiments using Mininet show that SDN-Guard maintains network performance during DoS attacks and succeeds in reducing by up to 32% their impact on controller performance, usage of switch forwarding tables, and control plane bandwidth. Furthermore, our results show that carefully placing the IDS and selecting the switches mirroring, the traffic can reduce by up to 90% the switch-to-IDS traffic. They also show that the IDS accuracy remains at 100% by analyzing only 11% of the network traffic.

KEYWORDS

DoS attacks, IDS accuracy, IDS placement, network security, software-defined networking (SDN), traffic sampling

1 | INTRODUCTION

Software-defined networking (SDN) is a new paradigm transforming the way IT networking infrastructures are managed, controlled, and configured. The SDN perspective relies on the separation of the control plane (ie, the network intelligence) from the data plane (ie, packet forwarding). The control plane comes then under the responsibility of a centralized

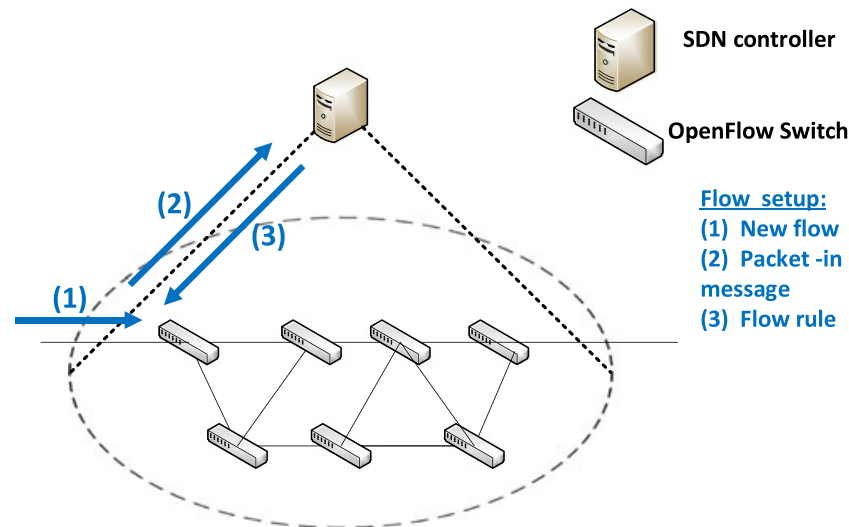


FIGURE 1 Typical software-defined networking (SDN) deployment

controller that takes all flow forwarding decisions in the network. The communication between the 2 planes is achieved through the OpenFlow protocol specified by the Open Networking Foundation.¹

Typically, in an OpenFlow-based SDN network, when a switch receives a new packet, it checks first its flow table (ie, called also TCAM table) to determine the output port (Figure 1). If the packet does not match an existing entry in the flow table, a packet-in message is transferred by the switch to the controller inquiring a new flow rule entry. The controller decides of the routing path and instructs all the involved switches with the rules to handle this new flow. Each flow entry is characterized by a hard time-out after which the switch automatically deletes the entry. Figure 1 illustrates a typical SDN deployment and the described steps.

As SDN technology is gaining momentum and several Internet providers are gradually embracing it, there is a growing attention to the security concerns that might arise when it comes to its real-world deployment.² In this context, deny-of-service (DoS) attacks are the most popular and inevitable threats to SDN networks, as they have always been to traditional networks. Usually, DoS attacks aim at overloading network links and the targeted servers by aggressively flooding them with the traffic until they fail to serve legitimate users.

Because of the centralized control of the SDN architecture, DoS attacks might have severe impacts on the network performance leading to cases where the entire control plane becomes completely crippled. More specifically, a DoS attack in an SDN network can lead to the following issues:

- (i) Overloading the SDN controller: If the controller is overloaded, packet-in messages will be stuck in the controller's queue, and no more routing decisions can be taken for the new incoming flows. In this case, flows with no forwarding entries will be stuck in the switches.
- (ii) Exhausting the control plane bandwidth (ie, switch-to-controller bandwidth): This problem is tightly related to the previous one. However, in this case, the switch-to-controller links are congested; some packet-in messages might then be lost, which will delay the decision regarding the waiting flows.
- (iii) Switch TCAM memory overflow: DoS attacks can purposely create a large number of new flows that might saturate the flow forwarding tables of the switches. When this happens, switches are forced to constantly add and remove flow entries and to send more packet-in messages towards the controller.

In this paper, we present SDN-Guard, a novel SDN application that protects SDN networks against DoS attacks and mitigates their impact on the SDN controller performance, the consumption of the control plane bandwidth, and the switch TCAM usage. Unlike existing works, SDN-Guard is designed to mitigate simultaneously these issues by dynamically managing flow routes, rule entry time-outs, and the aggregate flow rule entries based on the alerts generated by an intrusion detection system (IDS). This paper extends our previous work.³ Specifically, we extend SDN-Guard to allow it to reduce the switch-to-IDS traffic (ie, the amount of network traffic mirrored by the switches to be inspected by the IDS) and minimize bandwidth usage while maintaining the IDS performance and detection accuracy. To achieve this goal, we propose to use traffic sampling, and we also devise an integer linear program (ILP) able to find the optimal placement for

the IDS and to determine the switches that should mirror the flows towards the IDS so as to minimize bandwidth consumption in the network. Through extensive experiments using Mininet and Snort, we found SDN-Guard reduces by up to 32% the impact of DoS attacks on the controller performance, switch TCAM usage, and control plane bandwidth. We also found that the accuracy of the IDS when detecting DoS attacks remains at 100% by analyzing only 11% of the network traffic and that this amount can be further reduced by up to 90% when the IDS is located at the optimal location that minimizes the switch-to-IDS traffic.

The remaining of this paper is organized as follows. Section 2 provides an overview of previous works addressing DoS attacks in SDN networks. Section 3 presents the details of our proposed solution. Finally, we present experimental results in Section 4 and draw our conclusions in Section 5.

2 | RELATED WORK

With the increasing adoption of SDN technology, a growing body of work is addressing its security issues and investigating how they can be mitigated.² In this work, we mainly focus on research efforts that have recently addressed DoS attacks in SDN networks.^{2,4-8}

For instance, Shin et al⁴ and Kandai et al⁵ have analyzed the impact of DoS attacks on the network performance and showed how such attacks may impact on several parameters like the control plane bandwidth (ie, controller-switch channel), latency, switches flow tables, and the controller performance. However, they do not provide any solution to address these issues.

L. Wei and C. Fung⁶ proposed FlowRanger, a scheme that allows to detect and mitigate DoS attacks. FlowRanger is implemented at the controller side and consists of 3 components: (1) the trust management component that calculates a trust value for each packet-in message based on its source, (2) the queuing management component that places the message in the priority queue corresponding to its trust value, and (3) the message scheduling component that process messages according to a weighted round-robin strategy. FlowRanger can reduce the impact of DoS attacks on network performance by guaranteeing that legitimate flows are served first in the controller. However, unlike SDN-Guard, it does not prevent flooding the controller and switch CAM tables overload.

Dao et al⁸ present a solution to protect SDN networks against distributed DoS attacks based on IP filtering technique. The proposed scheme analyzes user behavior and uses it to assign the time-outs for the flow entries. Short time-outs are assigned for malicious users flows, and long time-outs are used for trusted ones. This solution forces entries of malicious traffic to be quickly removed from switches CAM tables. However, this may lead to new packet-in messages to be sent to the controller if the flow duration is higher than the set time-out. Furthermore, this solution drops all malicious traffic, which may be problematic for false positive flows.

Sahay et al⁷ leverage the centralized design and the programmability offered by SDN technology and propose a self-management scheme in which an ISP and its customers cooperate to mitigate DoS attacks. In this scheme, the ISP collects threat information provided by customers to use it to enforce security policies and update flow tables in the network accordingly. If a flow is considered legitimate by the customers, the ISP controller will tag it with a high priority value so that it takes a path with higher quality. However, if there is a doubt about the legitimacy of the flow, the ISP controller will assign a low priority to the flow and direct through the path designated for malicious flow. This proposal reduces the impact of the DoS attack on the network performance by balancing the load across different paths, but it does not mitigate the risks of overloading the controller and flooding flow tables in the switches.

FloodGuard⁹ is a defense platform against DoS attacks in SDN networks that aims at avoiding overloading switch TCAMs, the control channel bandwidth, and the controller. To do so, the authors propose a special module, called proactive flow rule analyzer module, that imitates the controller and handles the PACKET_IN messages on behalf of the controller during the DoS attack. They also propose to limit the throughput of the messages forwarded to the controller so as to avoid overloading it. Unfortunately, FloodGuard may not be always efficient because the proactive flow rule analyzer module may not be able to accurately reason the runtime logic of the controller. Furthermore, limiting the control traffic rate may delay the processing of control messages and hence delay the rules' setup times.

AVANT-GUARD¹⁰ is another framework that improves the resilience of the control plane against TCP-SYN flooding. To do so, AVANT-GUARD extends the data plane to complete the TCP handshake with the TCP source and only communicates with the controller. If this handshake is successful, the data plane communicates with the destination and allows to establish the TCP connection between the source and the destination. This solution clearly adds a significant delay

TABLE 1 SDN-Guard vs existing solutions

Approach	Inspected Traffic Management		Attack Mitigation by Minimizing			
	Traffic sampling	Network BW	Controller workload	Control plane BW	Flow table size	Network BW usage
SDN-Guard	✓	✓	✓	✓	✓	✓
L. Wei and C. Fung ⁶	✗	✗	✓	✗	✗	✗
Dao et al ⁸	✗	✗	✓	✓	✓	✗
Sahay et al ⁷	✗	✗	✗	✗	✗	✓
Wang et al ⁹	✗	✓	✗	✓	✓	✓
Shin et al ¹⁰	✗	✗	✓	✓	✓	✗
Mohammadi et al ¹¹	✗	✗	✓	✓	✓	✓
Androulidakis et al ¹⁵	✓	✗	✗	✗	✗	✗
Kawahara et al ¹⁶	✓	✗	✗	✗	✗	✗
Ha et al ¹⁷	✓	✗	✗	✗	✗	✗

Abbreviation: SDN, software-defined networking, BW, Bandwidth.

when establishing TCP connections. It is also not easily deployable in practice, as it requires to add new features to the data plane in the switches.

Mohammadi et al¹¹ propose SLICOTS, a novel solution to mitigate TCP-SYN flooding attack in SDN networks. SLICOTS is a module plugged at the controller that monitors all ongoing TCP requests. For each TCP connection request, SLICOTS installs temporary forwarding rules between the client and the server to be used only for the TCP handshaking process. If the handshaking is successful, it installs then permanent forwarding rules. SLICOTS also blocks attackers that cause the creation of a large number of half-open TCP connections. The main limitation of this system is that all the TCP handshake packets should be forwarded to the controller, which might increase the control plane bandwidth consumption and overload the controller.

Furthermore, existing proposals leverage either a centralized IDS, ie, a single IDS that analyzes the traffic of the whole network, or a distributed IDS where multiple IDSs are spread across the network and sending alerts to a centralized server for further analysis.¹²⁻¹⁴ In this work, we focus on proposals that use a single IDS to analyze the traffic of the whole network and the solutions they proposed to reduce the amount of inspected traffic. For instance, Androulidakis et al¹⁵ proposed to analyze only small flows, as they assume that small flows are predominant in malicious traffic. Kawahara et al¹⁶ use sampled traffic to obtain flow statistics to detect traffic anomalies. Ha et al¹⁷ propose a strategy to determine the optimal sampling rate for the traffic to be inspected by the IDS to not exceed the capacity of the IDS. They hence formulated an optimization problem that determines packet sampling rate for each switch in the SDN network such that the IDS is not overloaded. The experimental results show that their solution keeps the detection accuracy around 100%. All these solutions focus on traffic sampling. However, they do not optimize the bandwidth consumed by the traffic mirrored by the switches towards the IDS to be inspected. They also not study the impact of the IDS location on the bandwidth consumption or deal with the attack mitigation.

Table 1 compares the proposed solution, SDN-Guard, to the existing ones with respect to their targeted performance objectives during DoS attacks. None of the existing solutions is able to simultaneously reduce the controller load, the switch-to-controller bandwidth, and also avoid flooding the CAM tables of the OpenFlow switches. In this work, we aim at achieving simultaneously these objectives to maintain an acceptable network performance during DoS attacks. Furthermore, the proposed solution minimizes the inspected traffic through sampling and leverages an ILP to minimize the switch-to-IDS traffic and the bandwidth consumed by such traffic in the network.

3 | PROPOSED SOLUTION

In this section, we present the design architecture of the proposed solution. We discuss its main components and major design rationales that were considered to achieve the targeted objectives.

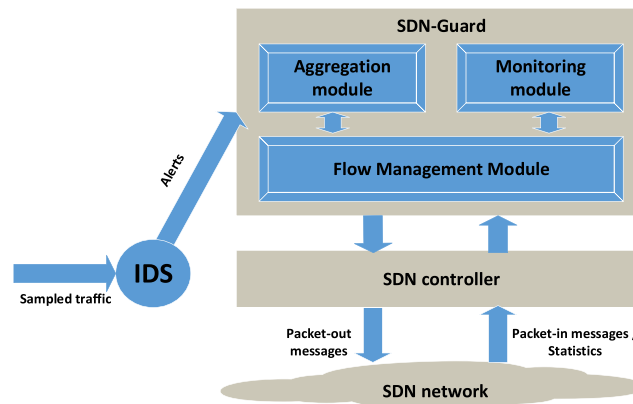


FIGURE 2 Solution architecture. IDS, intrusion detection system; SDN, software-defined networking

3.1 | Architecture overview

As depicted in Figure 2, SDN-Guard can be seen as an SDN application that can be plugged on top of any SDN controller. It relies on an IDS to analyze the network traffic and generate alerts whenever a malicious traffic flow is detected. Based on these alerts and the current network state, it takes the appropriate decision for each flow to minimize the impact of DoS attacks. SDN-Guard consists of the 3 following modules:

- **Flow management module** is responsible for selecting the routing paths for each of the flows and deciding the hard time-out of their corresponding TCAM entries based on the threat alerts sent by the IDS so as to mitigate the impact of the DoS attack. This module is also in charge of selecting the appropriate traffic sampling rate to be applied to the traffic before it is mirrored to the IDS to be analyzed. It also decides of the paths taken by the mirrored traffic towards the IDS in a way that minimizes link usage.
- **Rule aggregation module** is in charge of aggregating flow entries of the malicious flows having shared properties (eg, source and destination TCP port and IP addresses) to reduce the number of entries used in the switches TCAMs.
- **Monitoring module** is responsible for collecting multiple statistics about flows, switches, and links (eg, flow throughput, switch TCAM usage, and link bandwidth usage) to be used by other modules.

3.2 | IDS placement and switch-to-IDS traffic management

Software-defined networking-Guard resorts to an IDS that should analyze the traffic flows of the whole network and generate alerts whenever a malicious traffic is detected. The SDN-Guard receives these alerts and take the appropriate decisions to mitigate the attacks. One of the biggest challenges when deploying the IDS is to minimize the traffic sent by the switches to the IDS, as it might consume a tremendous amount of bandwidth and may overload the IDS.

In practice, there are 2 options to deploy intrusion detection systems able to analyze all the network traffic. The first option is to deploy multiple IDSs with one IDS connected to each switch. Each IDS analyzes then the traffic crossing its associated switch, which is a portion of the network traffic. However, the limitation of this solution is that some attacks like distributed DoS attacks may not be easily detected because each IDS is not aware of all the network traffic. In this case, more sophisticated solutions (eg, coordination and synchronization between IDSs) should be elaborated to allow detecting all types of attacks. The second option is to deploy a single IDS that analyzes all the traffic of the network. In this work, we adopt this solution because it has the advantage of detecting all types of attacks (eg, distributed DoS). However, there are 2 challenges to implement this solution. First, it is not practical to mirror all the network traffic to a single location, as this would consume a large amount of the network bandwidth. Second, the IDS may have limited resources and may not be able to process all the network traffic with an acceptable packet-processing time. To address these issues, we propose 2 solutions: (1) optimal IDS placement and traffic mirroring and (2) switch-to-IDS traffic sampling. In the following, we provide more details about these 2 solutions.

- *Optimal IDS placement and traffic mirroring:*

We should find the optimal location for the IDS and determine the switches that should mirror the flows in a way that minimizes the amount of the switch-to-IDS mirrored traffic and the amount of bandwidth consumed during this

operation (ie, by minimizing the number of links crossed by the mirrored traffic). In the following, we propose an ILP to model the IDS placement problem.

Let $G = (N, L)$ be a graph representing the network, where N is the set of switches and L is the set of links connecting them. We define $p_{n\bar{n}}$ as the cost of the shortest path from switch $n \in N$ to switch $\bar{n} \in N$, which corresponds to the number of hops between the 2 switches. Let $i \in I$ denote a flow crossing the network. We denote by f_i the throughput of the flow i . Define $r_{in} \in \{0, 1\}$ as a boolean variable that is equal to 1 if the flow $i \in I$ crosses the switch $n \in \bar{N}$. It is worth noting that the values of all the variables defined so far (ie, $p_{n\bar{n}}$, f_i and r_{in}) are known to the controller, as it is aware of the network topology and regularly receives flow statistics. Furthermore, a flow i cannot be forwarded from a switch n if it does not cross this switch. Hence, we have

$$x_{in} \leq r_{in} \quad \forall n \in N \forall i \in I. \quad (1)$$

We also define the decision variable $x_{in} \in \{0, 1\}$ as a boolean variable that indicates whether the flow i is mirrored from the switch n towards the IDS. Each flow i is mirrored one, and only one time, to the IDS. We therefore must satisfy the following constraint:

$$\sum_{n \in N} x_{in} = 1 \quad \forall i \in I. \quad (2)$$

The cost of mirroring all the flows to an IDS connected to switch $\bar{n} \in N$ corresponds to the amount of mirrored traffic forwarded from all switches to the IDS. It can be computed as

$$C_{\bar{n}} = \sum_{i \in I} \sum_{n \in N} x_{in} p_{n\bar{n}} f_i \quad \forall n \in N. \quad (3)$$

Finally, our ultimate objective is to find the switch $\bar{n} \in N$ that minimizes the mirroring cost:

$$\min_{\bar{n} \in N} C_{\bar{n}}. \quad (4)$$

The proposed ILP provides the location of the IDS (ie, \bar{n}) as well as the switches that should forward the traffic to the IDS (ie, using the decision variable x_{in}).

A well-known similar problem is the facility location problem that consists of selecting a factory location that minimizes total weighted distances from suppliers and customers, where weights are representative of the difficulty of transporting materials.^{18,19} The IDS placement and traffic mirroring problem does not only select the best location for the IDS but also decides of the switches that should mirror each flow towards it. As a result, the IDS placement problem generalizes the facility location problem, which is known to be \mathcal{NP} -hard.^{18,19} It can therefore be solved in a timely manner only for small-scale networks and a limited number of flows. In this paper, we resort to the ILP solver *lpsolve*²⁰ to find the optimal solution. Proposing an algorithm able to find a near optimal solution for a larger scale is part of our future work.

It is also worth noting that, in practice, the IDS could be hosted in a virtual machine, and hence, we have the flexibility to instantiate it and migrate it dynamically. In this case, the IDS placement problem could be solved periodically to adapt the location of the IDS to the traffic fluctuations over time. However, if the IDS is a dedicated hardware, the placement is designed for a longer time frame, as we do not have the flexibility offered by virtual machines.

- *Switch-to-IDS traffic sampling:*

To further reduce the amount of traffic mirrored by the switches towards the IDS, we sample the traffic before it is forwarded to the IDS. The sampling should reduce the workload of the IDS and the amount of traffic without hurting its performance and accuracy. In the experimental results (Section 4), we further study the impact of the sampling rate to figure out the optimal sampling rate that would not hurt the IDS performance and accuracy.

3.3 | Threat-based routing

To mitigate the impact of DoS attacks on bandwidth consumption and queuing delays, SDN-Guard, and more precisely the flow management module redirects malicious traffic through the paths with the least-used links in terms of bandwidth consumption and switch TCAMs. These 2 parameters are known to the flow management module thanks to the statistics collected by the monitoring module. It is worth noting that the selected path using these parameters may not be the shortest path; however, it ensures a minimal impact of attack on the performance of legitimate flows. At the same time, malicious traffic will reach the destination (which is important in case of false positive malicious flows) where it can potentially be further analyzed by intrusion detection or prevention systems. We do not opt for dropping malicious traffic

TABLE 2 Flow management decisions

Flow type	Threat			Rule Aggregation
	Probability	Time-out	Path	
Legitimate	Low	Default	Shortest	Optional
Malicious	High	High	Least-used links	Mandatory

to make sure that false positive flows get their chance to reach the destination, even with higher delays. It is worth noting that some studies showed that false positive rate can go up to 40% in some cases.²¹⁻²³ As a result, we do not drop malicious traffic because if a flow is wrongly classified as malicious and then dropped, some users may not be able to transmit their data, and hence, this may impact the overall quality of service. To avoid this potential issue, we choose to reroute malicious traffic through the least-used paths in terms of bandwidth consumption and size of the switches' TCAMs to reduce their impact on the legitimate traffic. As to legitimate flows, they are always routed through the shortest paths between the source and the destination so as to ensure a minimal round-trip time (RTT) delays.

Hence, whenever an alert about a malicious flow is received, the flow management module takes the decision about its new route and the time-out for its corresponding entries in the switches' TCAMs. Two cases can be identified:

- **Case 1 – malicious flow:** Whenever the flow management module receives an alert about a malicious traffic, it assigns a large hard time-out value to its corresponding flow rule and selects the least-used links in terms of bandwidth consumption and switches' TCAMs to ensure that this flow does not compete with legitimate flows and impact their performance.

The aggregation module analyzes the generated rules for such malicious traffic and tries, when possible, to merge the rules to reduce their number, and thereby minimize the flow table usage.

- **Case 2 – legitimate flow:** When there are no alerts for a particular flow, this flow is considered legitimate. The flow management module routes then the flow through the shortest path and assigns it a regular hard time-out value.

Table 2 summarizes the main decisions taken by SDN-Guard depending on the type of the received flow.

3.4 | Time-out management

The flow management module assigns the time-out value of each of the flow rules according to the received alerts. As the switch will have to communicate with the controller whenever the hard time-out expires, a small hard time-out will result in much more communication traffic with the controller. This will not only increase the switch-to-controller bandwidth consumption but also overload the controller. Hence, if the incoming flow is considered malicious, SDN-Guard assigns its forwarding rules a high time-out. The rationale behind this decision is to ensure that the same flow does not trigger many communications between the switch and the controller.

3.5 | Malicious flow rule aggregation

As malicious flows are assigned a large hard time-outs, such flow entries will remain for a long time in the TCAM table of switches. This might increase the number of used entries in the flow tables and might overload them. To address this issue, flow rule entries of malicious flows at a particular switch are automatically aggregated by the flow aggregation module if they have some shared properties (eg, same source and destination) and forwarded to the same outgoing link.

4 | EXPERIMENTAL RESULTS

In this section, we first start by describing the experimental setup used to emulate an SDN network and the tools used to generate normal traffic and DoS attacks. We then show the results showing the impact of the IDS location on the amount of mirrored traffic and the impact of traffic sampling on the IDS accuracy and performance. Finally, we evaluate the efficiency of SDN-Guard in mitigating DoS attacks compared to a baseline approach.

4.1 | Experimental setup

To set up our experiments, we use Mininet 2.3.0,²⁴ a popular SDN network emulators that allows to create a network of virtual hosts, virtual links and OpenVSwitch (OVS) switches. The created network is controlled by Floodlight 1.2, which is

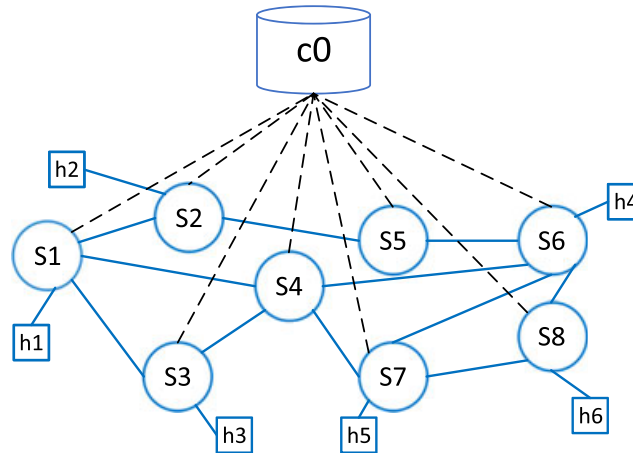


FIGURE 3 Emulated topology

a widely used java-based OpenFlow controller.²⁵ The communication between the switches and the Floodlight controller uses OpenFlow protocol version 1.3. We also use Snort 2.0.6²⁶ as an intrusion detection system that analyzes the traffic in the network and sends alerts to SDN-Guard. We conducted the experiments on a server running Ubuntu 14.04 with a CPU Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz \times 8 and 8 GB of RAM.

To evaluate our proposal, we emulated a topology consisting of 8 OpenFlow switches and 6 hosts as shown in Figure 3. The controller C0 is connected to switch S5, and all the control traffic between the switches and the controller is forwarded through the network (ie, via the switch S5). Furthermore, the hosts h_1 , h_2 , and h_3 are the attackers whereas host h_6 is the server targeted by the DoS attack.

We start the experiment by sending normal traffic made of TCP flows originating from all sources to all destinations. The DoS attack starts afterwards and lasts for 10 minutes during which the server h_6 is flooded with a large number of new TCP flows. The traffic returns to a normal behavior later on and only a normal amount of TCP flows is sent to the targeted server. To launch the DoS attack, we send TCP traffic using *hping3* network tool,²⁷ which floods the server h_6 with a large number of TCP-SYN, ICMP, and UDP packets with different IP source addresses. Such traffic emulates a distributed DoS attack coming from multiple sources.

The sampling technique was implemented using OpenFlow and the *tc* command at the Open vSwitch interface. Basically, the controller sends an specific OpenFlow packet (that we have created) where the sampling rate for the traffic is specified. The Open vSwitch uses this value to randomly drop packets from the traffic forwarded to the IDS according to the specified sampling rate.

4.2 | IDS accuracy and performance

In the first part of our study, we evaluate the accuracy of the IDS using sampled traffic, and we also analyze its performance in terms of packet-processing time when different sampling rates are used.

In our experiments, we generate 3 types of DoS attacks, namely, TCP-SYN flooding, UDP flooding, and ICMP flooding for 30 minutes. We run several experiments with different sampling rates. A sampling rate of $p\%$ means that $p\%$ of the mirrored traffic is randomly dropped at the switches before being forwarded to the IDS. The accuracy is measured by the percentage of detected attacks, which is computed as the number of attacks successfully detected when sampled traffic is analyzed divided by the total number of attacks detected when all traffic is analyzed.

Figure 4 shows the accuracy of the IDS using different sampling rates. It clearly shows that the accuracy of snort remains at 100% for a sampling rate lower than 89% and then starts decreasing when the sampling rate exceeds 89%. This means that the IDS is able to detect DoS attacks with the same accuracy with only 11% of the overall network traffic. According to our results, Snorts provides the same alerts (ie, true and false positive and negative alerts) when using all the traffic or sampled traffic. The veracity of the alerts depends on the quality of the rules used by Snort, which is out of the scope of this paper.

Another important advantage of sampling is that it reduces the IDS workload and hence leads to a lower packet-processing time at the IDS. The packet-processing time is the time spent by Snort to analyze a packet.

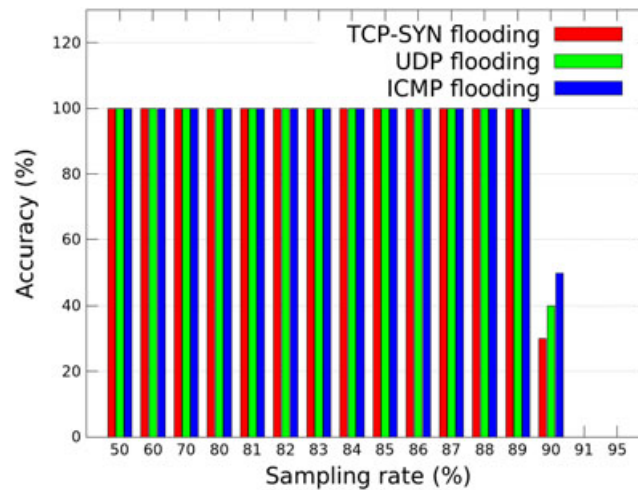


FIGURE 4 Snort accuracy versus sampling rate for each type of deny-of-service attack

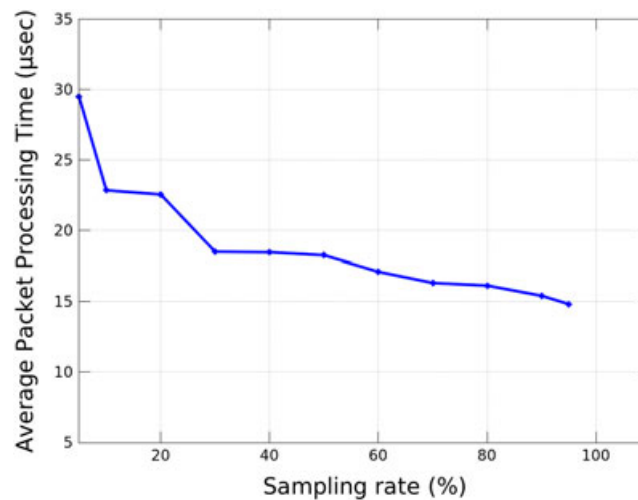


FIGURE 5 Average packet-processing time

Intuitively, it depends on the number of security rules checked by Snort for each packet and the Snort workload. Figure 5 shows the average packet-processing time in Snort with different sampling rates. Without traffic sampling (0% sampling rate), Snort receives all the network traffic and the average packet-processing time is around 30 microseconds. It then gradually decreases up to 16 microseconds as the sampling rate goes up to 91%. From both Figures 4 and 5, we can conclude that a sampling rate of 89% not only reduces the amount of traffic mirrored to the IDS but can also cut down the packet-processing time up to 50% while maintaining the IDS accuracy at 100%.

4.3 | IDS optimal placement

We have also investigated how to find the best placement for the IDS, that is, the best location that minimizes the amount of traffic mirrored to the IDS location and hence reduces the amount bandwidth consumed by such traffic. To do so, we run several experiments while varying at each time the location of the IDS. Hence, in each experiment, we connect the IDS to one of the switches and compute the amount of traffic that has been mirrored to the IDS by the other switches. This is a relevant metric as this traffic consumes a tremendous amount of bandwidth in the network in its way to reach the IDS. We exclude the traffic that naturally crosses the location of the IDS as the traffic reaches that location with no cost.

Figure 6 shows the amount of mirrored traffic for different IDS locations (ie, switches). This amount does not take into consideration the traffic that is naturally forwarded through that location. For instance, if the IDS is connected to switch S4, the traffic naturally crossing S4 will be directly forwarded to the IDS with no cost, and hence, we do not include it in

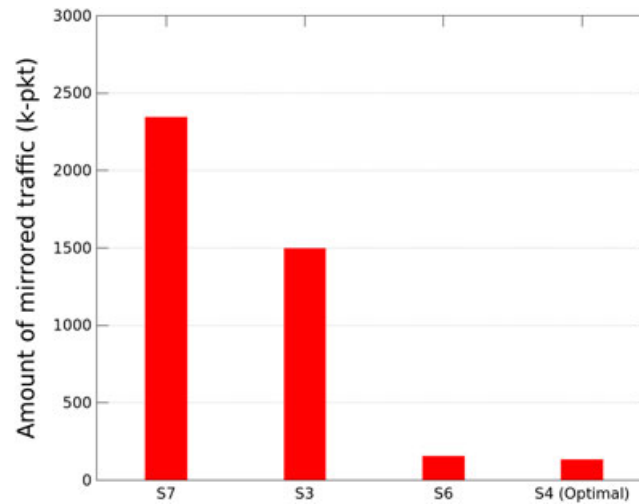


FIGURE 6 Amount of traffic mirrored to an intrusion detection system location

the computed traffic. We only consider the traffic forwarded by all the other switches to switch S4 because it consumes the network bandwidth to reach the IDS. From the figure, we can see that, when the optimal location of the IDS is used, ie, the switch S4, the amount of traffic is the lowest compared to other locations such as switches S7, S3, and S6. In particular, the amount of traffic mirrored to this optimal location is about 90% less than some other locations (eg, S7). It is also worth noting that the accuracy of the IDS remains 100% regardless of its location. This is expected as the IDS receives the same traffic wherever it is located.

4.4 | SDN-Guard efficiency

In the second part of our study, we focus on studying the efficiency of SDN-Guard. We mainly analyze the following metrics: the controller incoming throughput, the average TCAM table size of the switches, the end-to-end network throughput, and the average RTT. To show the benefits of the proposed scheme, we run the same experiment 2 times: one time with the baseline routing algorithm that uses the shortest path to deliver packets (ie, without SDN-Guard) and another time with SDN-Guard activated.

- **Controller incoming throughput:**

To study the behavior of the controller during an attack, we analyze the throughput of packet-in messages received by the controller from all the switches in the network. Figure 7 shows the throughput of packet-in messages received by the floodlight controller requesting new flow rules. It is clear that during the attack, there is a surge in the packet-in number received by the controller. However, compared with the baseline, we can see that SDN-Guard succeeds in

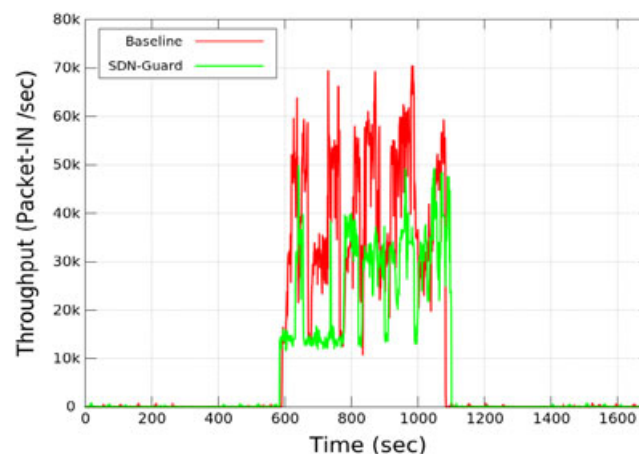


FIGURE 7 Controller incoming throughput. SDN, software-defined networking

reducing this throughput by up to 32%. This is mainly because SDN-Guard sets high hard time-outs to the forwarding rules associated with the malicious flow, and thereby significantly reduces the need to re-inquire the controller for new flow rules.

- **Average switch table size:** The switch table size can be easily flooded with DoS attacks because of the huge amount of new flows sent to switches, requiring a large number of flow rules to be stored.

It is clearly shown in Figure 8 that, with SDN-Guard, the number of flow rules in the table of the switch S_1 decreases by up to 26% compared to the baseline. This is achieved because (1) the malicious traffic is forwarded through the least-used links in terms of bandwidth consumption and switch TCAMs, which means that flow entries will be inserted through switches of different paths (ie, not only the switches of the shortest path), and (2) the aggregation module makes sure to minimize the number of flow entries of the malicious flow by aggregating them using common properties (eg, same source and destination and same next hop). We also note that similar results were found for the other switches in the considered network.

- **Throughput from source to destination:** We also measure the source-to-destination throughput, which is the number of packets sent from the source h_1 and received at the destination h_6 . Figure 9 shows that, before the attack, there is almost no packet loss as the throughput of the sent packets is equal to the received one. However, during the attack, the received throughput is much less than the received showing a high loss rate. With SDN-Guard, the received throughput is relatively less affected by the attack and still higher than the baseline case (ie, without SDN-Guard). By analyzing the results obtained during the attack, we find also that, with the baseline, there is 40% packet loss compared to 35% with our solution. This decrease in the packet loss is achieved by SDN-Guard because malicious traffic is balanced across the least-used links, which reduces congestion risks. Although this is not one of our main objectives, but it can be considered as another benefit of SDN-Guard.

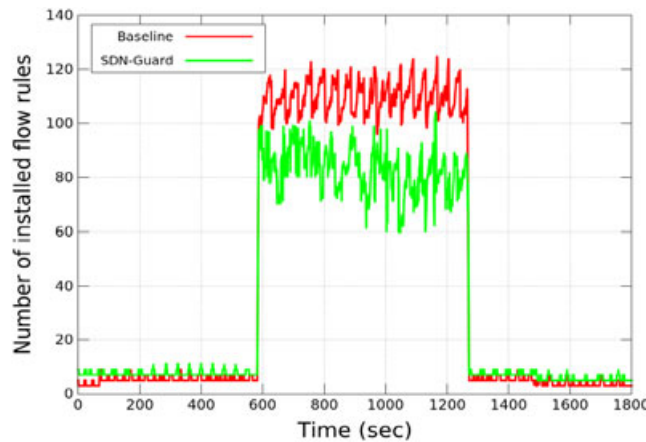


FIGURE 8 Average table size of switch s_1 . SDN, software-defined networking

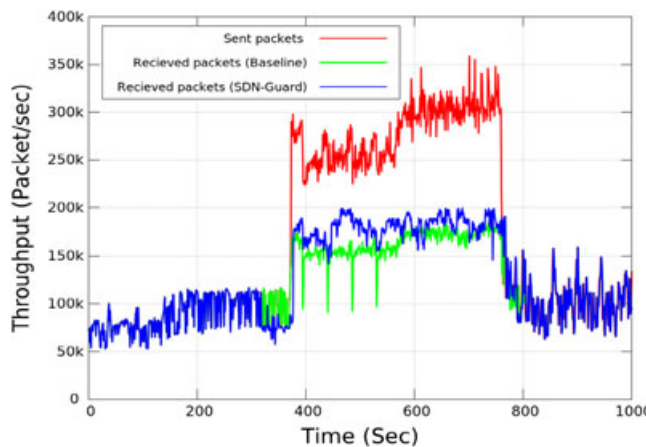


FIGURE 9 Sent and received packet throughput. SDN, software-defined networking

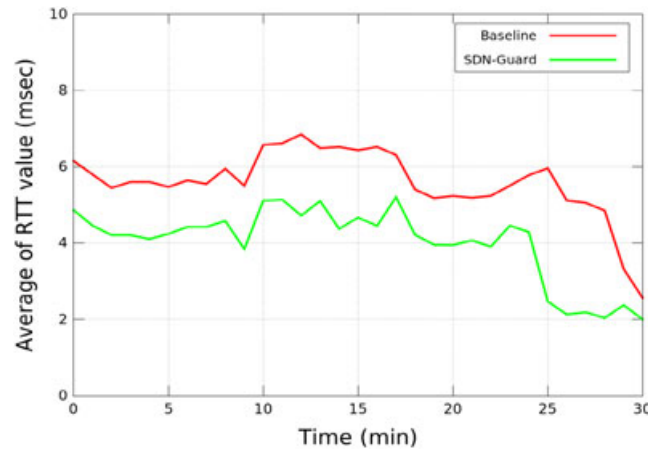


FIGURE 10 Average round-trip time (RTT) between h_1 and h_6 . SDN, software-defined networking

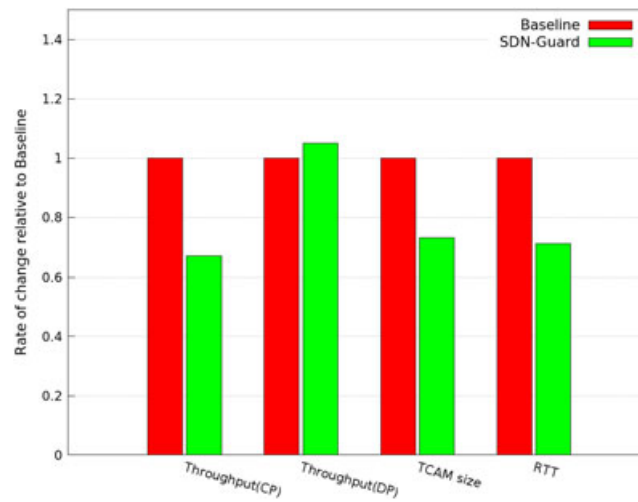


FIGURE 11 Software-defined networking (SDN)–Guard efficiency on performance parameters. CP, control plane; DP, data plane; RTT, round-trip time

- **Impact on average RTT:** Another important performance metric is the source-to-destination RTT. We hence draw the average RTT value over time to study the impact of DoS attack on this parameter with and without using SDN-Guard. We can see from Figure 10 that the average RTT value decreases by up to 23% when SDN-Guard is activated. This is because hard time-outs are set high for malicious traffic. Hence, the switches do not have to request new flow rules much often for the same flow. This eliminates the time needed to send the request to the controller and waiting the flow entry.

Figure 11 compares the baseline approach and SDN-Guard with regards to the studied metrics, ie, throughput at the control plane (throughput CP), throughput at the data plane (throughput DP), TCAM size, and the RTT. We have normalized the values according to the baseline value (we divide the metric average by the metric average found with the baseline, and hence, the normalized value of any metric for the baseline is always equal to 1). We can see from the figure that the data plane throughput at the edges is slightly improved using SDN-Guard (ie, 5%). The improvement is not high because, with SDN-Guard, all the traffic, including the malicious flows, is sent to the destination). However, with SDN-Guard, we reduce to up to 30% the control plane throughput, the TCAM size at the switches, and the RTT as shown in Figure 11.

5 | CONCLUSION

In this paper, we proposed SDN-Guard, a holistic approach to mitigating DoS attacks in SDN networks. Indeed, SDN-Guard relies on the alerts generated by an intrusion detection system that analyzes the whole network traffic to

efficiently protect SDN networks against DoS attacks by dynamically rerouting potential malicious traffic, adjusting flow time-outs and aggregating flow rules associated with the malicious traffic.

Furthermore, we investigated the impact of the IDS location on the amount of the traffic mirrored by the switches towards it. We proposed an ILP to find the optimal location of the IDS and the switch that should forward each flow to minimize this traffic. We have also studied the use of sampling techniques to reduce the amount of mirrored traffic sent to the IDS from the switches. We found that carefully placing the IDS and selecting the switches to mirror the traffic to the IDS could decrease the amount of the switch-to-IDS traffic by up to 90%. We also found that 11% of the traffic is sufficient to detect all DoS attacks. Finally, we evaluated the performance of SDN-Guard. The conducted experiments using Mininet showed that SDN-Guard succeeds in minimizing the impact of DoS attacks and reduces by up to 32% the controller incoming throughput and the control plane bandwidth and cuts down by up to 26% switch memory usage. Furthermore, we showed also that SDN-Guard significantly reduces packet loss and average packet RTT in the network during DoS attacks.

ORCID

Mohamed Faten Zhani  <http://orcid.org/0000-0001-9511-346X>

REFERENCES

1. Open Networking Foundation. <https://www.opennetworking.org/>. Accessed May 10, 2016.
2. Scott-Hayward S, O'Callaghan G, Sezer S. IEEE SDN for Future Networks and Services (SDN4FNS). Trento, Italy: 2013:1-7.
3. Dridi L, Zhani MF. SDN-Guard: DoS attacks mitigation in SDN networks. In: IEEE International Conference on Cloud Networking (Cloudnet); 2016.
4. Shin S, Gu G. Attacking software-defined networks: a first feasibility study. In: ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking; Hong Kong, China: 2013:165-166.
5. Kandoi R, Antikainen M. Denial-of-service attacks in OpenFlow SDN networks. In: IFIP/IEEE International Symposium on Integrated Network Management (IM); Ottawa, Canada: 2015:1322-1326.
6. Wei L, Fung C. FlowRanger: a request prioritizing algorithm for controller DoS attacks in software defined networks. In: IEEE International Conference On Communications (ICC); London, UK: 2015:5254-5259.
7. Sahay R, Blanc G, Zhang Z, Debar H. Towards autonomic DDoS mitigation using software defined networking. In: Network and Distributed System Security (NDSS) Symposium; San Diego, CA: 2015:1-7.
8. Dao N-N, Park J, Park M, Cho S. A feasible method to combat against DDoS attack in SDN network. In: International Conference on Information Networking (ICOIN); Cambodia: 2015:309-311.
9. Wang H, Xu L, Gu G. Floodguard: a dos attack prevention extension in software-defined networks. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN); Washington, DC: 2015:239-250.
10. Shin S, Yegneswaran V, Porras P, Gu G. Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: ACM SIGSAC conference on Computer & Communications Security; Berlin, Germany: 2013:413-424.
11. Mohammadi R, Javidan R, Conti M. SLICOTS: an SDN-based lightweight countermeasure for TCP SYN flooding attacks. *IEEE Transactions on Network and Service Management*. 2017;14(2):487-497.
12. Lopez MA, Mattos DMF, Duarte OCMB. An elastic intrusion detection system for software networks. *Ann Telecommun*. 71(11):595-605.
13. Chin T, Mountrouidou X, Li X, Xiong K. Selective packet inspection to detect DoS flooding using software defined networking. In: IEEE International Conference on Distributed Computing Systems Workshops; Columbus, OH: 2015:95-99.
14. Xing T, Huang D, Xu L, Chung C-J, Khatkar P. Snortflow: a openflow-based intrusion prevention system in cloud environment. In: GENI Research and Educational Experiment Workshop (GREE); Salt Lake City, UT: 2013:89-92.
15. Androulidakis G, Papavassiliou S. Improving network anomaly detection via selective flow-based sampling. *IET Commun*. 2008;2(3):399-409.
16. Kawahara R, Ishibashi K, Mori T, et al. Detection accuracy of network anomalies using sampled flow statistics. In: IEEE Global Telecommunications Conference (GLOBECOM); Washington, DC: 2007:1959-1964.
17. Ha T, Kim S, An N, et al. Suspicious traffic sampling for intrusion detection in software-defined networks. *Computer Networks*. 2016;109(P2):172-182.
18. Shmoys DB, Tardos É, Aardal K. Approximation algorithms for facility location problems (extended abstract). In: Annual ACM Symposium on Theory of Computing (STOC). ACM; New York, USA: 1997; 265-274.
19. Drezner Z, Hamacher HW. *Facility Location: Applications and Theory*. New York: Springer; 2004.
20. Ipsolve. <http://lpsolve.sourceforge.net/5.5/>. Accessed April 10, 2017.
21. Gupta BB, Joshi RC, Manoj M. An efficient analytical solution to Thwart DDoS attacks in public domain. In: International Conference on Advances in Computing, Communication and Control (ICAC3); Mumbai, India: 2009:503-509.
22. Tjhai GC, Papadaki M, Furnell SM, Clarke NL. The problem of false alarms: evaluation with snort and DARPA 1999 dataset. In: International Conference of Trust, Privacy and Security in Digital Business (TrustBus); Turin, Italy: 2008:139-150.

23. Hu L, Li T, Xie N, Hu J. False positive elimination in intrusion detection based on clustering. In: International Conference on Fuzzy Systems and Knowledge Discovery (FSKD); Zhangjiajie, China: 2015:519-523.
24. Mininet Overview. <http://mininet.org/overview/>. Accessed May 26, 2016.
25. Floodlight controller. <http://www.projectfloodlight.org/>. Accessed March 10, 2016.
26. Snort. <https://www.snort.org/>. Accessed April 10, 2017.
27. Hping3: Network tools. <http://linux.die.net/man/8/hping3/>. Accessed May 10, 2016.

How to cite this article: Dridi L, Zhani MF. A holistic approach to mitigating DoS attacks in SDN networks. *Int J Network Mgmt.* 2017;e1996. <https://doi.org/10.1002/nem.1996>