DDoS Flooding Attack Mitigation in Software Defined Networks

Safaa MAHRACH[1]
Computer, Networks, Mobility
and Modeling laboratory
Faculty of Sciences and Technology,
Hassan 1st University, Settat, Morocco

Abdelkrim HAQIQ[2]
Computer, Networks, Mobility
and Modeling laboratory
FST, Hassan 1st University, Settat, Morocco
e-NGN Research Group, Africa and Middle East

Abstract—**Distributed denial of service (DDoS) attacks which have been completely covered by the security community, today pose a potential new menace in the software defined networks (SDN) architecture. For example, the disruption of the SDN controller could interrupt data communication in the whole SDN network. DDoS attacks can produce a great number of new and short traffic flows (e.g., a series of TCP SYN requests), which may launch spiteful flooding requests to overcharge the controller and cause flow-table overloading attacks at SDN switches. In this research work, we propose a lightweight and practical mitigation mechanism to protect SDN architecture against DDoS flooding threats and ensure a secure and efficient SDN-based networking environment. Our proposal extends the Data Plane (DP) with a classification and mitigation module to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures. The simulation results indicate that the proposed defending mechanism may efficiently tackle the DDoS flood attacks in the SDN architecture and also in the downstream servers.**

Keywords*—*Software Defined Networks (SDN); Distributed Denial of Service (DDoS); network security; P4 language; DDoS mitigation

I. Introduction

Software-Defined Networking (SDN) is an emerging network architecture that enables dynamic and efficient programmability, management, and provision of the networks. This helps managers control the entire network systematically and globally, regardless of the underlying network infrastructure. SDN capabilities, such as network-visibility, centralized control, programmability, software-based traffic analysis enable the network administrator to implement easily in-network security functions, such as firewall, intrusion detection system (IDS), intrusion prevention system (IPS), etc. In this sense, many research works have developed SDN-based frameworks for enhancing network-security in smart grids, IoT, cloud, etc. [1], [2], [3], [4]. Although SDN features help in managing and protecting large networking environments against various threats, SDN itself suffers from both the present security attacks and new issues. Due to the centralized controller and the network programmability of SDN, new security challenges emerged including, malicious applications, controller-aimed distributed denial of service (DDoS) attacks, Fraudulent flow rules, flow-table overloading attacks, etc [7], [5], [6], [22].

The DDoS attacks which have been completely covered by the security community, today pose a potential new menace to the availability and scalability of the SDN network management. For example, an attacker can make the controller unavailable by producing a series of new TCP SYN requests, which involve the controller in a series of useless processes. In this case, the controller will be saturated and so unable to process legitimate requests. Moreover, an attacker may harness the limited storage capacity of the switch flow tables and launch the saturation attack (i.e. flow-table overloading attack). At the same time, SDN data plane presents a high throughput packet processing and filtering capacities which can be used to detect and mitigate DDoS attacks. In this research work, we exploit the SDN data plane capabilities to design a lightweight and practical DDoS mitigation mechanism for protecting the SDN architecture and ensure a secure and efficient SDN-based networking environment.

We extend the Data Plane (DP) with a classification and mitigation module to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures. Since the data plane has limited memory, we opted to use the stateless SYN cookie [30] technique which doesn't require storage of TCP connection states. We implement and perform the proposed mitigation prototype in P4 language [17] using the behavioral model (bmv2) software switch and Mininet emulator [16]. The simulation results indicate that the proposed defending mechanism may efficiently tackle the DDoS flood attacks in the SDN architecture and also in the downstream servers.

Benefits of the proposed approach:

- Defend the data plane from saturation attacks using SYN cookie, as a stateless technique, which doesn't require storage of TCP connection states.
- Discharged the communication path between control and data planes by performing detection, classification, and mitigation functions at the switch level.
- Protect the centralized controller and the downstream servers from the flooding attacks which affect their availability and scalability.
- Enhance the resistance of SDN technology against flooding DDoS attacks.

The present paper is organized as follows: In Section II we discuss how the SDN characteristics make it more vulnerable to DDoS attacks. In Section III, we evaluate some of the existing research works which propose security solution to detect and mitigate DDoS threats in SDN. We present the design and architecture of our approach “DDoS flooding attack mitigation in SDN” in Section IV. The simulation implementation and results are presented and evaluated in Section V. Finally, we give our conclusion and perspectives in Section VI.

II. Motivation

A. DDoS Impact in SDN

DDoS attacks which have been completely covered by the security community, today pose a potential new menace to the availability and scalability of the SDN network management [18], [7], [26], [27].

The SDN architecture is divided into three planes including, application plane, control plane, and data plane. All these planes and Application Programming Interfaces (APIs) can be targeted by the attackers to launch DDoS attacks.

- Control plane: The controller is the brain of the SDN architecture which provides central control over the network. Hence, it could be seen as a single point of failure if it is made unreachable by a Distributed Denial of Service (DDoS) attack. For example, an attacker can make the controller unavailable while producing a series of new TCP SYN requests, from distributed bot clients, which involves the controller in a series of useless processes. In this case, the controller will be saturated and so unable to process legitimate requests. In the control plane, DDoS attacks can target controller services, northbound interface, southbound interface, eastbound interface, and westbound interface.
- Data plane: The state-full implementations and functionalities in SDN like connection tracking (conntrack) [29] in Open vSwitch, OpenState [28], registers and counters in P4 [17], all require storing the state of flow in the data plane. Therefore, forwarding elements are again vulnerable to the saturation attack.
- Application plane: In the application plane, unauthenticated and unauthorized applications pose a great challenge for SDN. An attacker can launch an unauthenticated and unauthorized application with malicious programs running on the network devices and so the attacker can easily gain control of the network. Furthermore, the problem of isolation of applications and resources is not well solved in SDN; as a result, a DDoS attack on one application can affect other applications as well.

As we have seen above, DDoS attacks have a great impact on SDN application, control, and data planes. These challenges motivate us to make a study of the state-of-the-art of the detection and mitigation methods and techniques [9], [10], [11], [12], [14], [15] proposed to address DDoS impact in SDN network.

III. Related Work

Most of the research projects [23], [24], [25], [2], [13] develop defending mechanism against DDoS attacks using SDN whereas, SDN architecture itself suffer from different kind of DDoS attacks. A limited number of works [9], [10], [11], [12], [14], [15] address the potential challenges to mitigate DDoS attacks in SDN. Among which we present and analyze the following research works:

Mousavi et al. [9] exploited the SDN controller’s broad network feature to protect SDN architecture from DDoS attacks. They proposed an entropy-based solution, that works within the controller, to detect DDoS attacks against SDN controllers. The proposed algorithm adds two functions to the controller; One is collecting the new incoming packets to the destination IP addresses into window size 50 and the other computes the entropy of each window and compares it to an experimental threshold. In [10] Tran et al. proposed the ODL-ANTIFLOOD solution which detects and mitigates the Controller-aimed DDoS flooding attacks. The proposed detection technique is developed based on the combination of entropy and packet in rate methods. The mitigation technique includes three steps: identify attack sources, mitigating the impact of attacks, and recovering SDN system after the attacks. Tank et al. [11] suggested an SDN-based Network Intrusion Detection System architecture . The proposed NIDS takes advantage of software-based traffic analysis and logical centralized control of SDN for detecting intrusion. They constructed a simple Deep Neural Network (DNN) for the intrusion detection system using the NSL-KDD Dataset. In [12] Li et al. applied the deep learning model to detect DDoS attacks in Software Defined Networking (SDN) environment using the ISCX data set. After the collection and analysis of network traffic feature information, the deep learning model is used for feature reduction and DDoS attack detection.

Their suggested solutions involve the controller in every operation to detect and mitigate DDoS attack, which may overload the control and data planes path and creates a potential bottleneck that impacts the network performance and restricts the scalability and reactivity of SDN controller. Moreover, the [11], [12] use complex method (i.e., deep learning) which requires a high memory and processing requirements.

Shin et al. [14] presented Avant-guard as a detection and prevention solution against the TCP SYN flooding attack. Avant-guard extends the data plane with the connection migration module which proxying the incoming TCP SYN packets and prevent the control plane from saturation attack. LineSwitch [15] is an improved Avant-guard solution. LineSwitch addressed the vulnerabilities and limitations of Avant-guard by proxying a minimum number of TCP SYN requests. The implementation of [14], [15] solutions requires complex design and several lines to add and modify to extend and customize the data-plane.

IV. System Design

Distributed Denial of Service (DDOS) is one of the former and the most common attacks which is increasing in size and frequency. In the past year, the cybersecurity vendor Akamai recorded hundreds of DDoS attacks per week. Recently, Kaspersky Labs observe an 84 percent rise in the number of DDoS attacks during the first three months of 2019 [8].

As presented in the literature, and as we discussed above in Section II, DDoS attacks are considered among the major security threats that exploit SDN vulnerabilities. The SDN

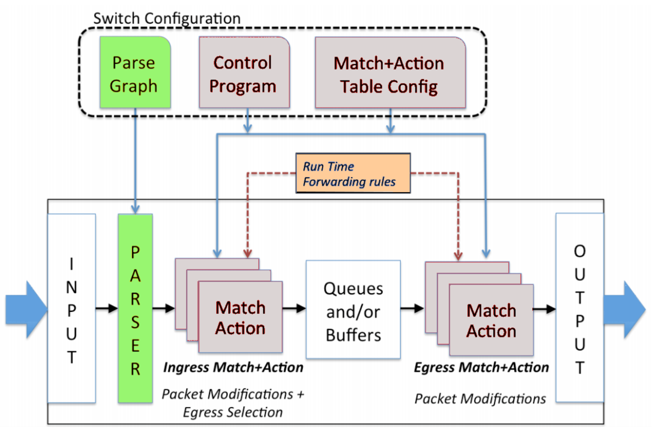


Fig. 1. Abstract forwarding model

centralized controller is an appealing target for DDoS threats. For example, an attacker can make the controller unavailable while producing a series of new TCP SYN requests, from multiple bot clients, which involves the controller in a series of useless processes. In this case, the controller will be saturated and unable to process legitimate requests. Moreover, an attacker may harness the limited storage capacity of the switch flow tables and launch the saturation attacks (i.e.,flow-table overloading attacks).

In this research work, we interest in implementing a lightweight and practical mechanism to protect the centralized SDN controller and the data plane from SYN flood attack, and achieve an efficient networking system that can resists against DDoS flooding attack.

The SDN data plane is a fast path; it has a high processing power which enables it to process a large number of packets in small time. This feature motivates us to perform a simple defensive mechanism (i.e., SYN cookie) in switch level to handle the incoming packets and classify the successful TCP connections from the failed one, which can be flooding attacks. Moreover, the data plane has limited memory, so we opted to use SYN cookie technique which doesn't require storage of TCP connections states.

Recently, Afek et al. [19] have implemented different SYN cookie methods, as anti-spoofing mechanisms, in OpenFlow 1.5 using Open vSwitch (OVS) and P4 to protect downstream server. Their programming experience indicates that the implementation of novel applications which operate with new header fields in OpenFlow environment required complex design and several lines to add and modify compared to the programming in P4 switch. That experience confirm and validate for us the use of P4 programming language [17] and bmv2 software switch to implement our proposed SYN flood mitigation mechanism.

A. Selected Methods

1) SYN cookie technique: SYN cookie [30] prevents the memory consumption caused by the half-open SYN attacks (i.e., TCP SYN flood attacks). The system (i.e., server or switch in our case) enabling the SYN cookie technique intercepts the SYN request received from a client and sends back a SYN-ACK packet with a pre-generated cookie (i.e., the Initial Sequence Number (ISN)). The cookie or ISN is generated using some details of the initial SYN packet and cryptographic hashing function to make the decoding of the cookie more complicated (as shown in Section IV-A2).

If the mitigating system receives an ACK packet from the client (with pre-generated cookie +1), it checks the validation of the TCP sequence number (i.e., is the ACK-1) as shown in Section IV-A2. There are different SYN cookie methods that exist to interact between clients and servers, among which we define; TCP proxy, TCP reset, safe reset, HTTP redirect [19]. In this work, we select the TCP-Reset method (Fig. 2) to use in our active defensive mechanism against SYN flooding attacks.

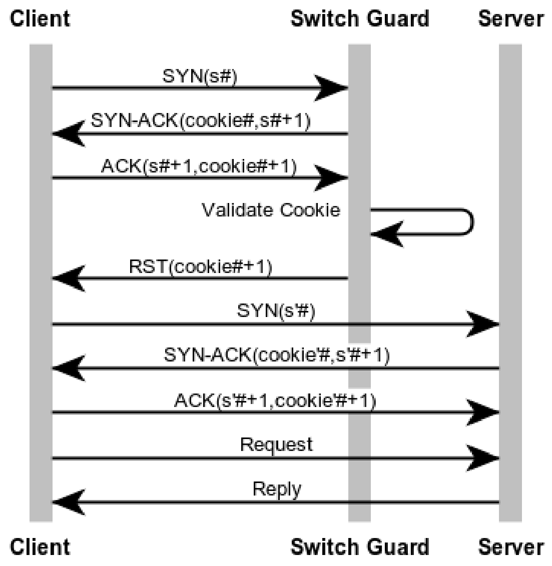


Fig. 2. TCP Reset method

TCP-reset method [31]: When the client is authenticated (i.e using SYN cookie) the mitigating system classifies it as legitimate (i.e., the system installs a rule by recording the source IP of the connection as legitimate). Then the switch sends back a TCP-reset packet (i.e., with source IP of the original server) to the client in order to enable him to re-establish the connection directly with the server. The advantage of this method is that is suitable for all TCP connections that attempt to connect when a RST packet is received.

2) Pre-generated Cookie: According to [20], the implementation of the SYN cookie technique must fulfill the following basic requirements:
- Cookies should contain some details of the initial SYN packet and its TCP options.
- Cookies should be unpredictable by attackers. It is recommended to use a cryptographic hashing function in order to make the decoding of the cookie more complicated. To this end, we select the recommended Linux SYN cookie method for generating and validating cookies [21].

TABLE I. PARAMETERS OF THE LINUX IMPLEMENTATION [21]

Parameter	Description
K_1, K_2	Secret keys
IP_s, IP_d	Source and destination IP addresses
$Port_s$, $Port_d$	Source and destination ports
ISN_s, ISN_d	Source and destination initial sequence numbers
ACK	Acknowledgement number
SEQ	Sequence number
MSS	2 bit index of the client's Maximum Segment Size
Count	32 bit minute counter
Hash()	32 bit cryptographic hash

Cookie generation:

$$H_1 = hash(K_1, IP_s, IP_d, Port_s, Port_d) \tag{1}$$

$$H_2 = hash(K_2, count, IP_s, IP_d, Port_s, Port_d) \tag{2}$$

$$ISN_d(cookie) = H_1 + ISN_s + (count \times 2^{24}) \tag{3}$$

$$+(H_2 + MSS) \mod 2^{24}$$

Cookie validation:

$$ISN_d = ACK - 1 \tag{4}$$

$$ISN_s = SEQ - 1 \tag{5}$$

$$count(cookie) = (ISN_d - H_1 - ISN_s)/2^{24} \tag{6}$$

$$MSS(cookie) = (ISN_d - H_1 - ISN_s) \tag{7}$$

$$\mod 2^{24} - H2 \mod 2^{24}$$

As we can see above and in Table I, we calculate the two hash values H1 and H2 (based on TCP options, secret keys k1, k2 and count) then we use them with ISNs and MSS to generate the cookie (ISNd), as it is shown in (3). For the cookie validation, there are 2 integrity controls (count(cookie) and MSS(cookie)). The first one checks the age of the cookie. The second evaluates whether the value of the MSS is within the 2 bit range (0-3). If the cookie meets both integrity controls, it is considered valid, and the connection can be accepted.

B. System Architecture

The goal of our proposed DDoS defensive mechanism is to activate SDN data plane with smart and advanced functions to prevent and mitigate SYN flood attacks in SDN architecture. To reach this aim, we extend the Data Plane (DP) with a classification and mitigation module to analyze and classify the new incoming packets and perform the adaptive countermeasures. In addition, we develop and implement a simple program (i.e., in python language) in the control plane (i.e., P4Runtime) to interact with P4 switch messages (packet-in).

Classification and Mitigation

The classification and mitigation module allows the Data Plane (DP) to analyze and classify the new incoming packets (i.e., which not exist in the flow table). DP will be able to classify the benign packets from the SYN flood attacks and to execute the adaptive countermeasures.

In first, DP checks if the packet exists in the flow table if so, the packet will be immediately forwarded to the destination server. Otherwise, the DP initiates a classification phase (Fig. 3) which classifies the validated TCP connections from failed ones (i.e., half-open SYN attacks or invalidated TCP connections).

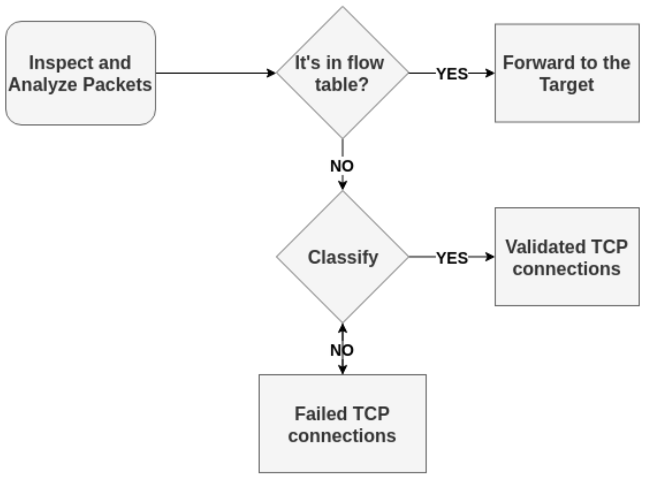


Fig. 3. TCP SYN flood defensive flowchart

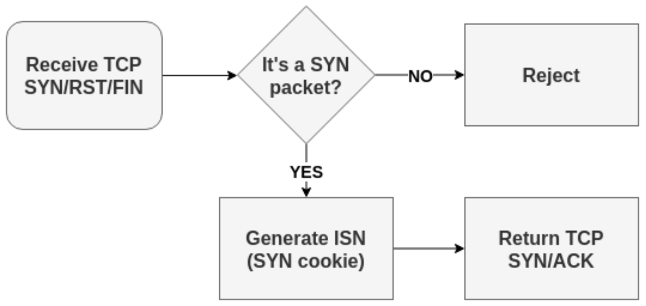


Fig. 4. Handling TCP SYN packets chart

When the DP receives a TCP SYN/RST/FIN packet, as shown in Fig. 4, it checks whether it is a SYN packet. If so, the DP responds to the SYN request client by a SYN-ACK packet with a pre-generated cookie (see IV-A2). The cookie or the Initial Sequence Number (ISN) is created by hashing details about the initial SYN packet and its TCP options. If the packet is not a TCP SYN packet (i.e., TCP FIN or TCP RST), it is rejected.

If the DP receives an ACK packet, as shown in Fig. 5, it checks the validation of the TCP sequence number (i.e., is the ACK-1) as shown in Section IV-A2. For the validation of the cookie, we control two integrities; count(cookie) for checking the age of the cookie (i.e., must be lower to 2min), and MSS(cookie) for evaluating whether the value of the MSS is within the 2-bit range (0-3).

If the TCP connection is validated the switch transmits the ACK packet to the controller to write the required flow-entry. At the same time, the switch sends back a TCP-reset packet (i.e., with source IP of the original server) to the client in order to enable him to re-establish the connection directly with the server. In our system, we use the P4Runtime API as a control plane specification for controlling the data plane

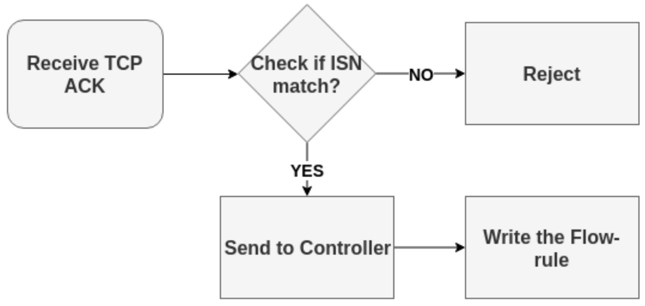


Fig. 5. Handling TCP ACK packets chart

elements executing the P4 programs.

P4Runtime API program

We develop a simple program in python language to interact with P4 switch messages (packet-in). The program connects the P4Runtime controller to the switch(s), installs the P4 program on the switch, populates the defined match action tables, and describes how the controller treats the packet-in messages.

When the controller receives a packet-in message it learns from the details of the initial SYN packet to create the requested flow-rule. It extracts the addresses of IP source, IP destination, Mac source, and Mac destination, and the ports of source and destination. It checks if the new flow has already a flow-rule in the memory of the controller, if it exists it writes it on the switch, if not it creates a new flow rule for the new flow and writes it on switch flow-table. Each flow rule is installed with a timeout value, both hard and idle timeouts, to avert the growth of rules in the switch flow-tables.

V. Simulation

A. Environment

We use the Behavioral model-bmv2 framework which supports the P4 software switch, with v1model architecture. This reference implementation covers P4.16 specification version 1.2.0 [17], and it functions as the data plane. We exploit bmv2 for using two targets `simple_switch` (i.e., this target is a software switch, running on a general-purpose CPU, such as Intel/AMD/etc., that can execute `P4_14` and `P4_16` programs) and `simple_switch_grpc`. The difference between both of the targets is that `simple_switch_grpc` can also accept TCP connections from a controller, where the format of control messages on this connection is defined by the P4Runtime API specification. The P4Runtime API is a control plane specification for controlling the data plane elements of a device or program defined by a P4 program. P4c is used as a compiler; it compiles the P4 program (.p4) into the JSON file to be implemented on the P4 software switch, and defines the tables and actions, in Protobuf format(), existing in P4 program to be populated by the controller.

B. Implementation

Our focus here is the software-based implementation, used to direct the packet handling process inside the switch, unlike the traditional SDN data plane which requires hard and heavy changes to extend and customize the data-plane. Our approach is based upon the Abstract forwarding model of P4 (see Fig. 1) which consists of headers, describe the fields and their sizes of each header within a packet; parser, defines the permitted header sequences within packets; control flow, organizes the layout of match action tables within ingress and egress pipeline, and the packet flow through the pipeline; match action tables, associate user-defined keys with actions; stateful memories, counters, meters, and registers are used to store information across packets.

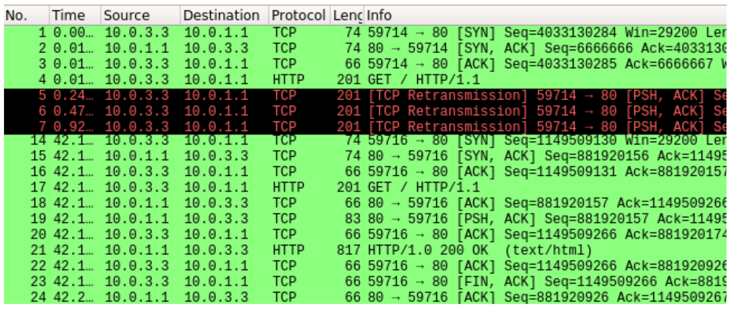
We develop a P4 program which describes how packets are analyzed and classified benign packets from flooding attacks, and which defines the adaptive actions and countermeasures to perform. The proposed processing pipeline in the data-plane is presented below. The parser is changed to extract TCP fields and TCP options. We add three match action tables: the first one, named `ipv4_lpm()`; which handles the incoming packets and forward to the target those matching the existing flow rules, the second one, named `return_SYN_ACK()`; responds to the SYN packet (which doesn't correspond to any flow entry) by sending a `SYN_ACK` packet with a pre-generated sequence number (see Section IV-A2), the third one, named `ACK_verify()`; it checks and verify the validity of the sequence number existing in the received ACK packet. If the sequence number of the ACK packet is validated the action `send_to_cpu()` is performed to forward the validated ACK packet to the control plane for adding the requested flow entry. The control plane (P4Runtime) learns from the received ACK packet to build the requested flow entry, and then write it on the data plane for handling the incoming packets from the same flow. Optionally, the switch sends back a TCP-reset packet (i.e., with source IP of the original server) to the client to enable him to re-establish the connection directly with the server.

```
 apply {
ipv4_lpm.apply();

if (hdr.tcp.flags == 02) {
   return_synack();
}
else if (hdr.tcp.flags == 10) {
    ACK_verify();
    if ((meta.meta.count_cookie <= 2) &&
    (0 <= meta.meta.mss_cookie) &&
    (meta.meta.mss_cookie <= 3)){
send_to_cpu();}
else{
drop(); }
}}
```

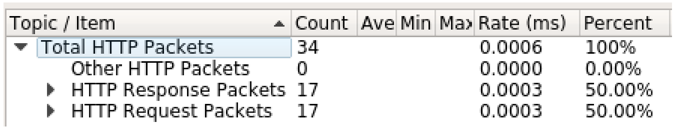
C. Use Case

The test environment of this experiment includes a p4 software switch in which our `p4_program` is implemented, P4Runtime API used to populate the existing match tables, a server that hosts HTTP service, benign clients which send HTTP requests, and an attacker who performs a TCP SYN flood attack using hping3. Fig. 6 shows a new client which sent an HTTP request targeted to the web server (10.0.1.1). Since the client is new (i.e., the proper rule doesn't exist in



```
No.  Time    Source    Destination Protocol Lenç Info
   1 0.00…   10.0.3.3  10.0.1.1    TCP        74 59714 → 80 [SYN] Seq=4033130284 Win=29200 Ler
   2 0.01…   10.0.1.1  10.0.3.3    TCP        74 80 → 59714 [SYN, ACK] Seq=6666666 Ack=403313C
   3 0.01…   10.0.3.3  10.0.1.1    TCP        66 59714 → 80 [ACK] Seq=4033130285 Ack=6666667 W
   4 0.01…   10.0.3.3  10.0.1.1    HTTP      201 GET / HTTP/1.1
   5 0.24…   10.0.3.3  10.0.1.1    TCP       201 [TCP Retransmission] 59714 → 80 [PSH, ACK] Se
   6 0.47…   10.0.3.3  10.0.1.1    TCP       201 [TCP Retransmission] 59714 → 80 [PSH, ACK] Se
   7 0.92…   10.0.3.3  10.0.1.1    TCP       201 [TCP Retransmission] 59714 → 80 [PSH, ACK] Se
  14 42.1…   10.0.3.3  10.0.1.1    TCP        74 59716 → 80 [SYN] Seq=1149509130 Win=29200 Ler
  15 42.1…   10.0.1.1  10.0.3.3    TCP        74 80 → 59716 [SYN, ACK] Seq=881920156 Ack=11495
  16 42.1…   10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [ACK] Seq=1149509131 Ack=881920157
  17 42.1…   10.0.3.3  10.0.1.1    HTTP      201 GET / HTTP/1.1
  18 42.1…   10.0.1.1  10.0.3.3    TCP        66 80 → 59716 [ACK] Seq=881920157 Ack=1149509266
  19 42.1…   10.0.1.1  10.0.3.3    TCP        83 80 → 59716 [PSH, ACK] Seq=881920157 Ack=11495
  20 42.1…   10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [ACK] Seq=1149509266 Ack=881920174
  21 42.1…   10.0.1.1  10.0.3.3    HTTP      817 HTTP/1.0 200 OK  (text/html)
  22 42.1…   10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [ACK] Seq=1149509266 Ack=881920926
  23 42.1…   10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [FIN, ACK] Seq=1149509266 Ack=8819
  24 42.2…   10.0.1.1  10.0.3.3    TCP        66 80 → 59716 [ACK] Seq=881920926 Ack=1149509267
```

Fig. 6. Classification of successful TCP connection from flood attack: How the mitigation switch traits the legitimate packets



Topic / Item	Count	Ave	Min	Max	Rate (ms)	Percent
▾ Total HTTP Packets	34				0.0006	100%
Other HTTP Packets	0				0.0000	0.00%
▸ HTTP Response Packets	17				0.0003	50.00%
▸ HTTP Request Packets	17				0.0003	50.00%

Fig. 7. Wireshark statistics: HTTP packet counter in the Web server

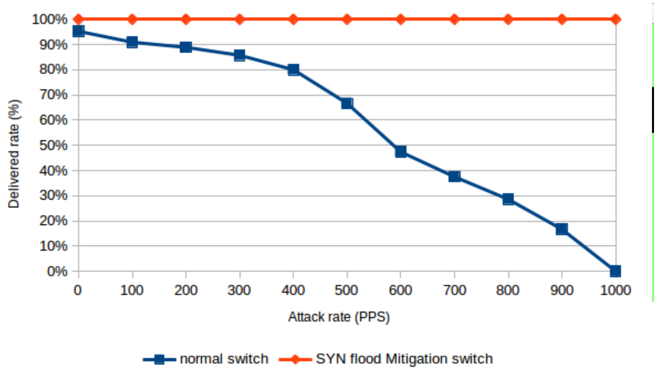
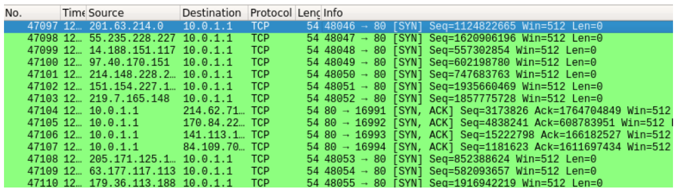


Fig. 8. Percentage of successfully delivered packets to the HTTP server from benign clients



```
No.    Time Source          Destination Protocol Lenç Info
 47097 12…  201.63.214.0    10.0.1.1    TCP        54 48046 → 80 [SYN] Seq=1124822665 Win=512 Len=0
 47098 12…  55.235.228.227  10.0.1.1    TCP        54 48047 → 80 [SYN] Seq=1620906196 Win=512 Len=0
 47099 12…  14.188.151.117  10.0.1.1    TCP        54 48048 → 80 [SYN] Seq=557302854 Win=512 Len=0
 47100 12…  97.40.170.151   10.0.1.1    TCP        54 48049 → 80 [SYN] Seq=602198780 Win=512 Len=0
 47101 12…  214.148.228.2…  10.0.1.1    TCP        54 48050 → 80 [SYN] Seq=747683763 Win=512 Len=0
 47102 12…  151.154.227.1…  10.0.1.1    TCP        54 48051 → 80 [SYN] Seq=1935660469 Win=512 Len=0
 47103 12…  219.7.165.148   10.0.1.1    TCP        54 48052 → 80 [SYN] Seq=1857775728 Win=512 Len=0
 47104 12…  10.0.1.1        214.62.71…  TCP        54 80 → 16991 [SYN, ACK] Seq=3173826 Ack=1764704849 Win=512
 47105 12…  10.0.1.1        170.84.22…  TCP        54 80 → 16992 [SYN, ACK] Seq=4838241 Ack=608783951 Win=512
 47106 12…  10.0.1.1        141.113.1…  TCP        54 80 → 16993 [SYN, ACK] Seq=15222798 Ack=166182527 Win=512
 47107 12…  10.0.1.1        84.109.70…  TCP        54 80 → 16994 [SYN, ACK] Seq=1181623 Ack=1611697434 Win=512
 47108 12…  205.171.125.1…  10.0.1.1    TCP        54 48053 → 80 [SYN] Seq=852388624 Win=512 Len=0
 47109 12…  63.177.117.113  10.0.1.1    TCP        54 48054 → 80 [SYN] Seq=582093657 Win=512 Len=0
 47110 12…  179.36.113.188  10.0.1.1    TCP        54 48055 → 80 [SYN] Seq=1916942219 Win=512 Len=0
```

Fig. 9. How our mitigation system reacts to spoofed SYN flood attack

flow-table), the mitigation switch will respond by a SYN-ACK packet and then check the validation of the received ACK packet (see the three first lines). This latter is sent to the controller to add the appropriate flow-entry. Then, the legitimate client (10.0.3.3) connects directly to the web server and get the requested data as we can see in the lines (14-24).

In this scenario, we measure the delivered packet rate of benign clients under network saturation attack. Fig. 9 indicates a spoofed SYN flood attack targeted the web server. In this case, the mitigation switch intercepts and responds to the received SYN requests to validate the successful TCP connections and ignore the failed one. Even though the network is under several flood attack, the web server is available (i.e., because it receives only the authenticated TCP connections) and can process and respond to the HTTP requests from legitimate clients as shown in Fig. 7. To further show the impact of saturation attacks on normal traffic in detail, we vary the packet sending rate of the TCP SYN flood attack from 0 to 1000 per second, and at the same time, we send the HTTP requests from 15 benign clients. The test results are shown in Fig. 8.

With SYN flood mitigation switch: The SYN flood packets are blocked by the mitigation switch and so, only the validated TCP connections are transmitted to the controller for writing the new flow-rules. In this case, web server will receive only the SYN requests from benign clients. As shown in 8, the packet rate of benign clients are `100%` delivered to the web server, even while the network is under a severe network saturation attack.

With normal switch; All the SYN requests, from the attacker, are forwarded to the controller as packet-in messages and so, the controller is saturated and the flow table is also overloaded, which explain, the packet rate of benign clients are nearly `0%` delivered.

D. Evaluation and Comparison

Under SYN flooding attack the data plane receives many TCP SYN packets, from random IP sources, that it will then transmit to the controller. These TCP requests may saturate the channel between control and data planes, the couple of SDN controller and switches, and also the downstream servers.

The proposed SYN flood mitigation mechanism enables the data plane to handle the new incoming packets and forward to the controller only the requests of the successful TCP connections. It discharges the controller-switch path and reduces the involvement of SDN controller. Thus, the SDN network will be more protected from Controller-aimed distributed denial of service (DDoS) attack and flow-table overloading attack. Moreover, it defends the downstream servers from DDoS flooding attacks by replying unconditionally to the received TCP SYN requests. Consequently, the proposed mechanism makes the SDN network more scalable and efficient to resist such attacks.

Our prototype uses the SYN cookie technique which doesn't require the storage of network states in the data plane. This is in contrast to previous works, such as Avant-guard [14], which use SYN proxy method to protect SDN controller against SYN flood attacks. The SYN proxy requires the information storage (timestamp, sequence number, source IP and port) throughout the TCP connections, which gives rise to a new type of SYN flooding attack named Buffer Saturation Attack.

The implementation of the proposed method uses software-based environment to direct the packet handling process inside the switch, unlike the traditional hardware-based implementation which requires hard and heavy changes to extend and customize the data-plane.

In comparison to some existing solutions, our approach implements simple method functionalities over SDN data plane rather than complex methods, such as machine learning or deep learning which require high memory and processing requirements [11], [12].

Moreover, it enhances the resilience against TCP SYN Flood attack and it may also be used to defence attacks based on HTTP, SMTP, and other protocols. This can be done while using specific SYN cookie methods, such as HTTP Redirect and TCP Safe Reset [19].

VI. Conclusion

The growing adoption of SDN technology in controlling and securing the legacy networks, such as cloud, IOT, cyber-physical systems have highlighted the requirement to analyze and evaluate the benefits and vulnerabilities of SDN architecture. While DDoS attacks remain a top threat that is growing in size and frequency of reported incidents, SDN opens the door for yet new vulnerabilities to this type of attacks. In this paper, we have examined how the SDN characteristics make it more vulnerable to DDoS attacks and we have evaluated some of the existing research works which propose solutions to detect and mitigate DDoS threats in SDN. In this sense, we have presented and implemented a lightweight and practical mitigation mechanism to protect SDN network against DDoS flooding attack. The proposed approach enables the data plane to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures.

In comparison to the existing solutions, our approach activates the mitigation of DDoS flooding attack at the SDN data plane without any external and dedicated appliance. Consequently, it prevents and reduces the risk of saturation attack in the SDN controller and switches. Moreover, the simulation results indicate that the proposed mechanism may efficiently tackle the DDoS flood attacks in both SDN architecture and downstream servers. For future work, we plan to conduct a study of the different network systems, such as cloud and IoT to decide which one to choose as an application domain in our experiments. Moreover, further experiments and simulations will be performed to support more sophisticated attacks.

References

[1] Maziku, H., Shetty, S. and Nicol, D.M., 2019. Security risk assessment for SDN-enabled smart grids. Computer Communications, 133, pp.1-11.

[2] Mahrach, S., El Mir, I., Haqiq, A. and Huang, D., 2018. SDN-based SYN flooding defense in cloud. Journal of Information Assurance and Security, 13(1).

[3] Gonzalez, C., Charfadine, S.M., Flauzac, O. and Nolot, F., 2016, July. SDN-based security framework for the IoT in distributed grid. In 2016 International Multidisciplinary Conference on Computer and Energy Science (SpliTech) (pp. 1-5). IEEE.

[4] Zaalouk, A., Khondoker, R., Marx, R. and Bayarou, K.M., 2014, May. OrchSec: An orchestrator-based architecture for enhancing network-security using Network Monitoring and SDN Control functions. In NOMS (pp. 1-9).

[5] Zhang, H., Cai, Z., Liu, Q., Xiao, Q., Li, Y. and Cheang, C.F., 2018. A survey on security-aware measurement in SDN. Security and Communication Networks, 2018.

[6] Rawat, D.B. and Reddy, S.R., 2016. Software defined networking architecture, security and energy efficiency: A survey. IEEE Communications Surveys & Tutorials, 19(1), pp.325-346.

[7] Mahrach, S. and Haqiq, A., 2019. DDoS Defense in SDn-Based Cyber-Physical Cloud. Cybersecurity and Privacy in Cyber Physical Systems, p.133.

[8] DDoS Attacks in Q1 2019 report. https://securelist.com/ddos-report-q1-2019/90792/.

[9] Mousavi, S.M. and St-Hilaire, M., 2018. Early detection of ddos attacks against software defined network controllers. Journal of Network and Systems Management, 26(3), pp.573-591.

[10] Tran, N.T., Le, T.L. and Tran, M.A.T., 2018, November. ODL-ANTIFLOOD: A Comprehensive Solution For Securing OpenDayLight Controller. In 2018 International Conference on Advanced Computing and Applications (ACOMP) (pp. 14-21). IEEE.

[11] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R. and Ghogho, M., 2016, October. Deep learning approach for network intrusion detection in software defined networking. In 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM) (pp. 258-263). IEEE.

[12] Li, C., Wu, Y., Yuan, X., Sun, Z., Wang, W., Li, X. and Gong, L., 2018. Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN. International Journal of Communication Systems, 31(5), p.e3497.

[13] Mahrach, S., Mjihil, O. and Haqiq, A., 2017, December. Scalable and Dynamic Network Intrusion Detection and Prevention System. In International Conference on Innovations in Bio-Inspired Computing and Applications (pp. 318-328). Springer, Cham.

[14] Shin, S., Yegneswaran, V., Porras, P. and Gu, G., 2013, November. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 413-424). ACM.

[15] Ambrosin, M., Conti, M., De Gaspari, F. and Poovendran, R., 2015, April. 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 (pp. 639-644). ACM.

[16] Mininet emulator. http://mininet.org/.

[17] The P4 Language Specification. https://p4.org/p4-spec/docs/P4-16-v1.2.0.pdf.

[18] Yan, Q., Yu, F.R., Gong, Q. and Li, J., 2015. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges. IEEE Communications Surveys & Tutorials, 18(1), pp.602-622.

[19] Afek, Y., Bremler-Barr, A. and Shafir, L., 2017, May. Network anti-spoofing with SDN data plane. In IEEE INFOCOM 2017-IEEE Conference on Computer Communications (pp. 1-9). IEEE.

[20] Simpson, W.A., 2011. TCP cookie transactions (TCPCT).

[21] Echevarria, J.J., Garaizar, P. and Legarda, J., 2018. An experimental study on the applicability of SYN cookies to networked constrained devices. Software: Practice and Experience, 48(3), pp.740-749.

[22] Zhang, H., Cai, Z., Liu, Q., Xiao, Q., Li, Y. and Cheang, C.F., 2018. A survey on security-aware measurement in SDN. Security and Communication Networks, 2018.

[23] Swami, R., Dave, M. and Ranga, V., 2019. Software-defined Networking-based DDoS Defense Mechanisms. ACM Computing Surveys (CSUR), 52(2), p.28.

[24] Birkinshaw, C., Rouka, E. and Vassilakis, V.G., 2019. Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks. Journal of Network and Computer Applications, 136, pp.71-85.

[25] Rebecchi, F., Boite, J., Nardin, P.A., Bouet, M. and Conan, V., 2019. DDoS protection with stateful software-defined networking. International Journal of Network Management, 29(1), p.e2042.

[26] Dong, S., Abbas, K. and Jain, R., 2019. A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. IEEE Access, 7, pp.80813-80828.

[27] Krishnan, S. and Oliver, J.J.E., 2019, April. Mitigating DDoS Attacks in Software Defined Networks. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI) (pp. 960-963). IEEE.

[28] Bianchi, G., Bonola, M., Capone, A. and Cascone, C., 2014. OpenState: programming platform-independent stateful openflow applications inside the switch. ACM SIGCOMM Computer Communication Review, 44(2), pp.44-51.

[29] The connection tracking system. http://docs.openvswitch.org/en///latest/tutorials/ovs-conntrack/

[30] Fontes, S.M., Hind, J.R., Narten, T. and Stockton, M.L., International Business Machines Corp, 2006. Blended SYN cookies. U.S. Patent 7,058,718.

[31] Touitou, D., Pazi, G., Shtein, Y. and Tzadikario, R., Cisco Technology Inc, 2011. Using TCP to authenticate IP source addresses. U.S. Patent 7,979,694.

AUTHORS' PROFILE

Safaa Mahrach has received Master degree in Networks and Computer Systems in 2013 from Hassan 1^{st} University of Settat, Morocco, and she is currently working toward the PhD degree from Computer, Networks, Mobility and Modeling laboratory, Mathematic and Computing Department at Hassan 1^{st} University, Morocco. His research interests are Security of computer networks and Cloud systems using the Software Defined Networking (SDN) paradigm and the P4 programming language. She is member of the P4 Language Consortium.

Abdelkrim HAQIQ has a High Study Degree and a PhD , both in the field of modeling and performance evaluation of computer communication networks, from the University of Mohammed V, Agdal, Faculty of Sciences, Rabat, Morocco. Since September 1995 he has been working as a Professor at the department of Mathematics and Computer at the Faculty of Sciences and Techniques, Settat, Morocco. He is the Director of Computer, Networks, Mobility and Modeling laboratory. He is also the General Secretary of the electronic Next Generation Networks (e-NGN) Research Group, Moroccan section. He is an IEEE Senior member and an IEEE Communications Society member. He was a co-director of a NATO Multi-Year project entitled "Cyber Security Analysis and Assurance using Cloud-Based Security Measurement system", having the code: SPS-984425. Dr. Abdelkrim HAQIQ's interests lie in the areas of modeling and performance evaluation of communication networks, mobile communications networks, cloud computing and security, queueing theory and game theory. He is the author and co-author of more than 150 papers (international journals and conferences/workshops). He is also a member of the board of the International Journal of Intelligent Engineering Informatics.