Enviado: 20/11/2022 Recibido: 02/12/2022 Aceptado: 12/02/2023

UNIVERSIDAD DISTRITAL FRANCISCO JOSE DE CALDAS

VISIÓN ELECTRÓNICA

https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

Vulnerability analysis of an emulated SDN network by flooding HTTP and TCP packets

Análisis de vulnerabilidades de una red SDN emulada mediante la inundación de paquetes HTTP y TCP

Dairon Javier Ramos Suavita[1] , Edith Paola Estupiñán Cuesta[2] , Juan Carlos Martínez Quintero[3]

Abstract

This article implements a topology of an SDN network in the Mininet emulator where a web server is implemented in one of the devices in order to execute a denial-of-service attack by sending mass packets with the aim of analyze what vulnerabilities can be found in the data and control plane of the SDN network architecture. The results were captured with the Wireshark tool to analyze the packets that enter the controller and command line to obtain data such as RTT (Round-Trip Time) and the connection speed with the server, as a result, a decrease in performance was evidenced. of the network in terms of the connection speed with the server was less than 40Mbps and the RTT with values up to 352ms that takes a packet to go and return when the flood of packets is executing in the network.

Keywords: Attack, Controller, Mininet, RTT, Software Defined Networks, Vulnerabilities.

[1] Universidad Militar Nueva Granada, Colombia. E-mail: est.dairon.ramos@unimilitar.edu.co ORCID: https://orcid.org/0000-0003-0254-8742

[2] Telecommunications Engineer, Universidad Militar Nueva Granada, Colombia. Master in Electronic Engineering, Pontificia Universidad Javeriana, Colombia. Universidad Militar Nueva Granada/Telecommunications Engineering Program. E-mail: edith.estupinan@unimilitar.edu.co ORCID: https://orcid.org/0000- 0002-4100-4943

[3] Electronic Engineer, Universidad Manuela Beltrán, Colombia. Specialist in Physical and Computer Security, Escuela de Comunicaciones Militares, Colombia. Master in Automatic Production Systems, Universidad Tecnológica de Pereira, Colombia. Universidad Militar Nueva Granada/Telecommunications Engineering Program. E-mail: juan.quinteroq@unimilitar.du.co ORCID: https://orcid.org/0000-0001-9893-6592

Resumen

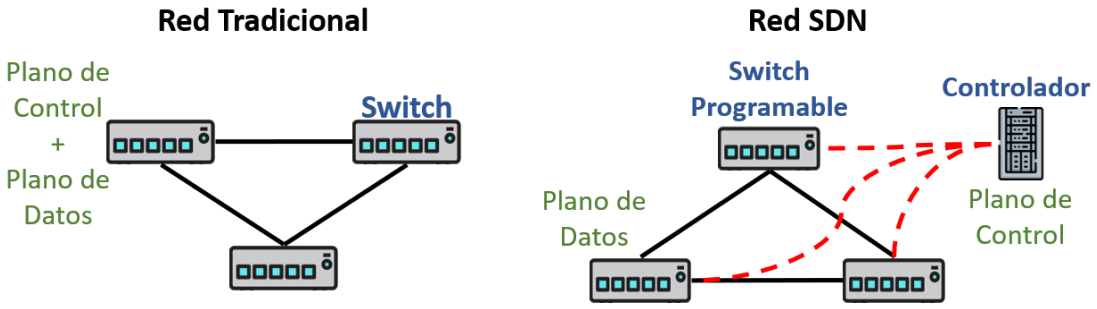
En este documento se realiza la implementación de una topología de una red SDN en el emulador Mininet donde se implementa un servidor web en uno de los dispositivos con el fin de ejecutar un ataque de denegación de servicio mediante el envío masivo de paquetes con el objetivo de analizar que vulnerabilidades se pueden encontrar en el plano de datos y de control de la arquitectura de la red SDN. Se capturaron los resultados con la herramienta Wireshark para analizar los paquetes que ingresan al controlador y línea de comando para obtener datos como el RTT (Round-Trip Time) y la velocidad de conexión con el servidor, como resultado se evidencio una disminución en el rendimiento de la red en cuanto a la velocidad de conexión con el servidor fue menos de 40Mbps y el RTT con valores hasta de 352ms que toma un paquete de ir y volver cuando se está ejecutando la inundación de paquetes en la red.

Palabras clave: Ataque, Controlador, Mininet, RTT, Redes definidas por Software, Vulnerabilidades.

1. Introduction

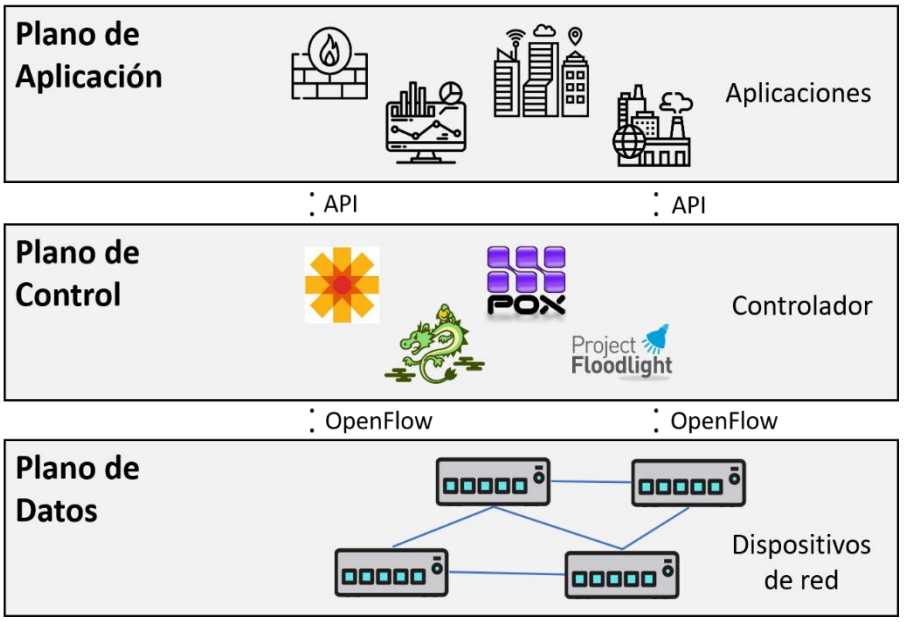
Software Defined Networking is an emerging technology in which the software is decoupled from the hardware, offering better advantages in network management by allowing programming and controlling the entire network from a single point of control instead of doing it device by device and traditional networks have a decentralized design in which each of its components is configured, having in the same computer the control plane and data plane. [1] (Figure 1). Its architecture is divided into three (Figure 2): Application plane, control plane and data plane, this structure provides fast access and better use of network resources, as the architecture is designed to streamline IT management and help organizations keep pace with today's growing traffic [2].

Figure 1. Traditional network vs SDN network.



Source: own.

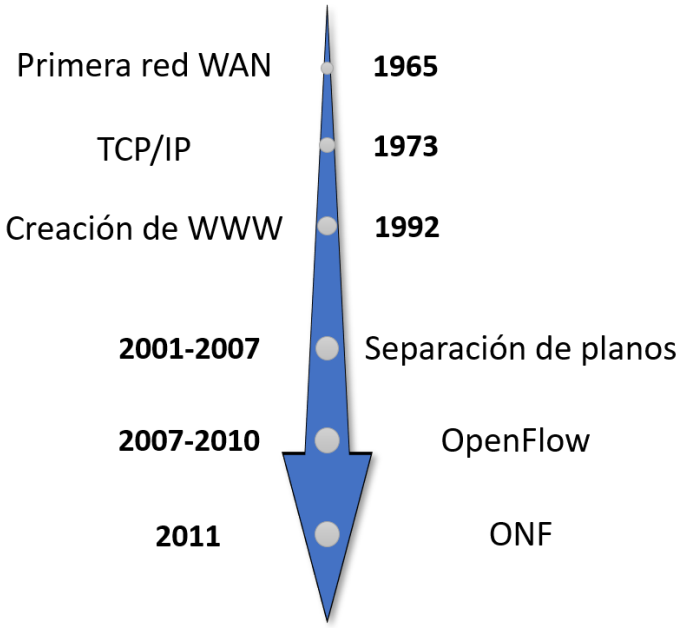
Figure 2. SDN architecture.



Source: own.

SDN networks arise from the separation of the planes (between 2001 and 2007) allowing to manage growing networks, between 2007 and 2010 the OpenFlow protocol was developed in order to control the forwarding behavior and communication between the two planes. [3] (Figure 3). Previous studies have analyzed that SDN networks, due to their architecture design, have different threats that can affect network performance and the possible attacks with the greatest threat are related to packet flooding in the network that aim to saturate or wear out the network resources [4]. [4][5][6][7].

Figure 3. Evolution of Data Networks.



Source: own.

In this paper a topology is implemented to perform the analysis of its vulnerabilities by means of packet flooding. This paper is divided into three main chapters which covers the study of the SDN network: 1) Proposed scenario with tests, 2) Implementation of the topology and 3) Results and Analysis.

1.1. Methodology

The following methodology was established, which consists of three phases for the development of the activities of this study, as shown in Figure 4:

- **Proposed scenario:** This phase proposes an SDN topology to be implemented and establishes tests to evaluate the network performance.
- **Implementation:** This phase implements the proposed topology in an emulator with the integration of a controller and runs the tests designed to test the SDN network.
- **Results and analysis:** This phase captures the data from each of the tests and performs the respective analysis of possible vulnerabilities in the emulated SDN network.

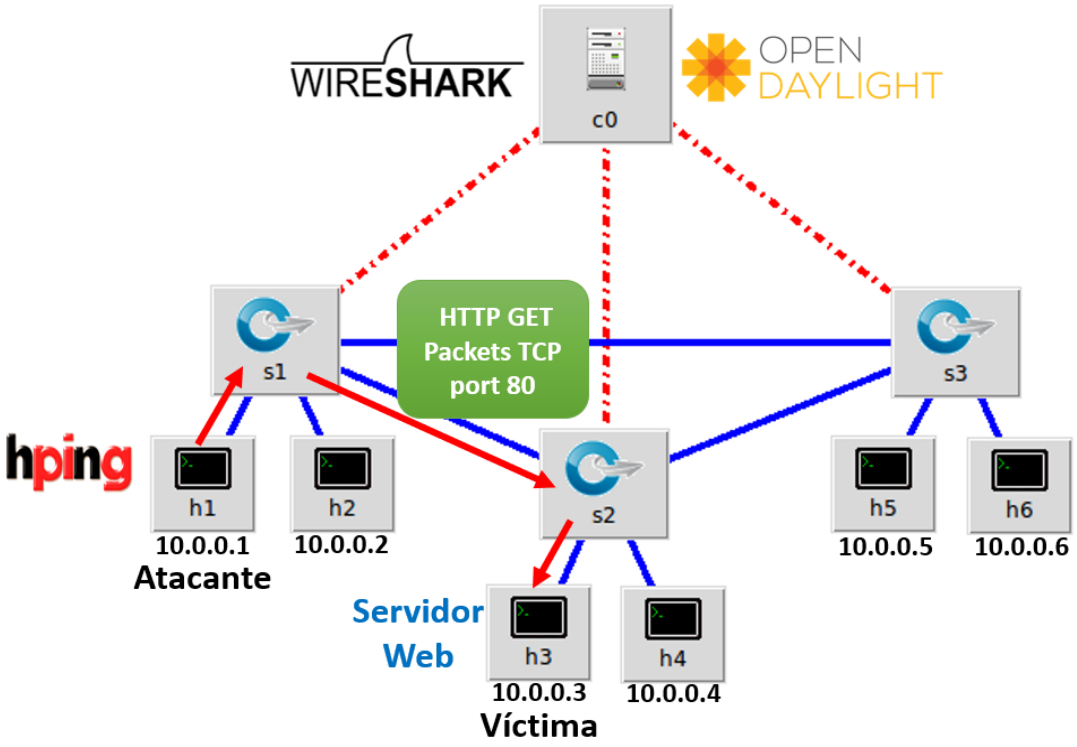
Figure 4. Phases of the methodology.

Source: own.

2. Scenario

For this study the following scenario is established consisting of six (6) hosts distributed in the network, three Open vswitch switches, the integration of the OpenDayLight driver and a Simple Web server as shown in Figure 5, implemented in the Mininet emulator with the objective of applying two evaluative tests to analyze which vulnerabilities can be discovered according to the results obtained from the Wireshark tool and command line. Details of the tests are given in section 2.1 and 2.2.

Figure 5. Proposed scenario implemented in Mininet.



Source: own.

2.1. Evaluation: Slowhttptest

In this first test, the Slowhttptest tool is used to send HTTP GET requests from the H1 device (10.0.0.1) to the web server (10.0.0.3), this test allows to identify the capacity of the number of connections that the server can maintain before the service collapses.

2.2. Evaluation: Package Variation

The second test consists of sending TCP packets through port 80 used by the Web Server, these packets have different sizes as shown in Table 1, which varies the packet size in three cases to identify the behavior of the network by capturing the RTT metrics for the time taken by the packet, the number of packets/sec in the controller (127.0.0.1) to request new flow rules and the connection speed between the client and the server.

Table 1. Variation in package size.

Attack	Packet size (bytes)	Metrics
Test 1	54	▪ Packets/sec ▪ RTT ▪ Connection speed
Test 2	554	
Test 3	1454	

Source: own.

3. Implementation

The topology emulation is performed on an Ubuntu operating system on which the Mininet software for network implementation is installed and the OpenDayLight driver for traffic management is linked. The device characteristics are listed in Table 2 below.

Table 2. Equipment and software requirements.

Requirements	Data
Operating System	Ubuntu 16.04 LTS
RAM memory	6 GB
CPU	Intel Core i3, 2Ghz
Mininet	2.3.0
OpenDayLight	0.3.0 Lithium

Source: own.

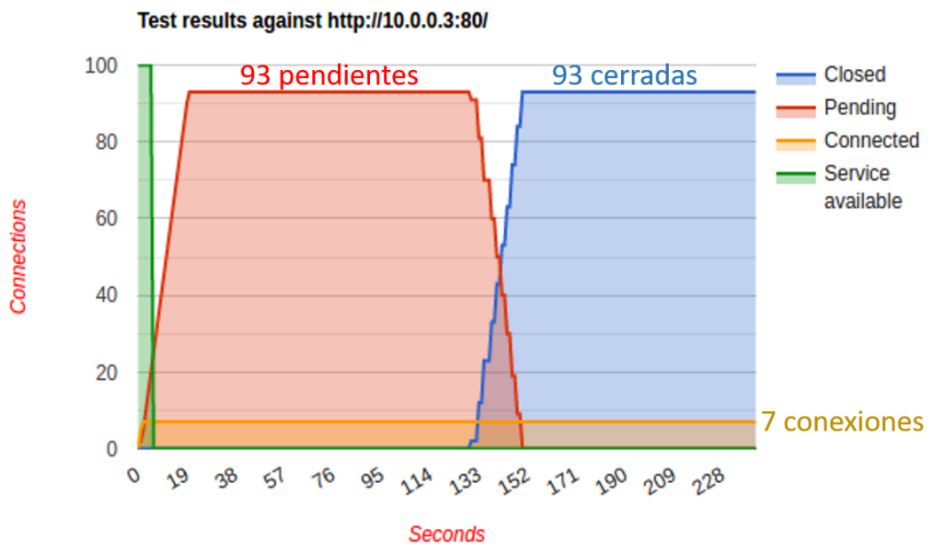
4. Results and Analysis

The following are the results and analysis of the data obtained in the different tests applied to evaluate the network.

4.1. Evaluation: Slowhttptest

In this test the HTTP server is implemented in the H3 computer to execute the test with the Slowhttptest tool from the H1 computer (10.0.0.1) sending 100 GET requests to the Web server (10.0.0.0.3). Figure 6 shows the test performed, where it is identified that out of the 100 connections only 7 managed to connect to the server before the Web server collapsed, the remaining 93 requests were pending and after a while they were closed because the server could not process the requests.

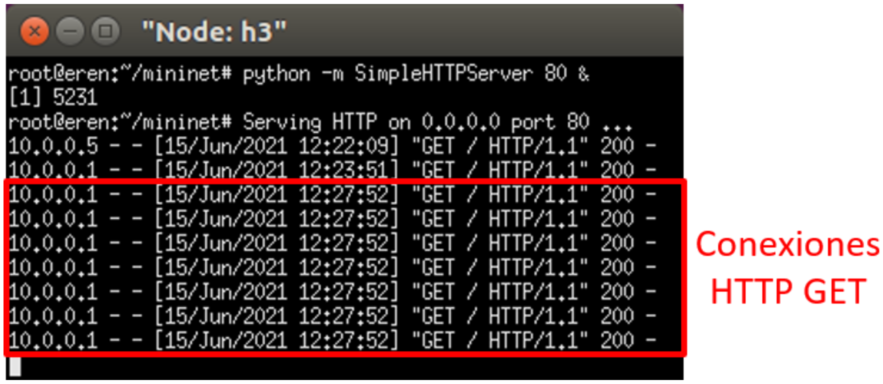
Figure 6. Connection test with Slowhttptest.



Source: own.

Figure 7 shows the connections successfully established with the web server (10.0.0.3) where the type of HTTP GET requests, the IP address of the attacker H1(10.0.0.1) that establishes the connection and the HTTP protocol are identified. The last connections made in the test are evidenced where it was identified that the server (10.0.0.3) maintained only 7 requests before the web server crashed.

Figure 7. Connection history on the Web server.



Source: own.

4.2. Evaluation: Variation of packages

As a second test, three packets were sent, which consisted of increasing the size to observe its behavior in terms of the controller and the connectivity in the Web server. The size of the three packets sent were: 54Bytes, 554Bytes and 1454Bytes. With the help of the Wireshark tool we observed that the packets that arrived at the controller present an increase in the length of the message of 108bytes that corresponds to the OpenFlow protocol that is identified in Figure 8, also the messages are of type "Packet_In" that has the function of requesting a new flow rule for each packet that is arriving at the switches to be forwarded.

Figure 8. Packet_In messages in the controller.

Prueba 1: 54 Bytes

No.	Time	Source	Destination	Protocol	Length	Info
515…	6.081221083	127.0.0.1	127.0.0.1	OpenFlow	162	Type: OFPT_PACKET_IN
515…	6.081231906	127.0.0.1	127.0.0.1	OpenFlow	162	Type: OFPT_PACKET_IN
515…	6.081240991	127.0.0.1	127.0.0.1	OpenFlow	162	Type: OFPT_PACKET_IN
515…	6.081261750	127.0.0.1	127.0.0.1	OpenFlow	162	Type: OFPT_PACKET_IN

Prueba 2: 554 Bytes

No.	Time	Source	Destination	Protocol	Length	Info
2082	8.365535040	127.0.0.1	127.0.0.1	OpenFlow	662	Type: OFPT_PACKET_IN
2083	8.365601553	127.0.0.1	127.0.0.1	OpenFlow	662	Type: OFPT_PACKET_IN
2084	8.366313741	127.0.0.1	127.0.0.1	OpenFlow	662	Type: OFPT_PACKET_IN
2085	8.366357659	127.0.0.1	127.0.0.1	OpenFlow	662	Type: OFPT_PACKET_IN

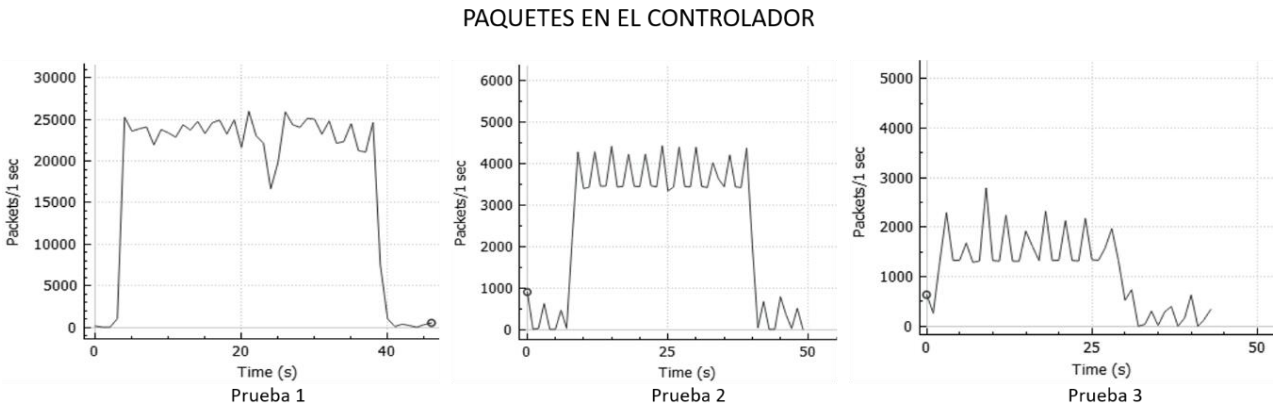
Prueba 3: 1454 Bytes

No.	Time	Source	Destination	Protocol	Length	Info
5159	4.508825012	127.0.0.1	127.0.0.1	OpenFlow	1562	Type: OFPT_PACKET_IN
5160	4.509997914	127.0.0.1	127.0.0.1	OpenFlow	1562	Type: OFPT_PACKET_IN
5162	4.511158534	127.0.0.1	127.0.0.1	OpenFlow	1562	Type: OFPT_PACKET_IN
5163	4.512306830	127.0.0.1	127.0.0.1	OpenFlow	1562	Type: OFPT_PACKET_IN

Source: own.

Wireshark allows to visualize the traffic of the controller as shown in Figure 9, it is identified that the number of packets/sec decreases as the packet size increases which shows that the larger the packet size the less packets are generated to attack while with the size of 54bytes up to 25000 packets/sec of "Packet_In" messages are obtained in the controller.

Figure 9. Packages in the Controller according to the test.

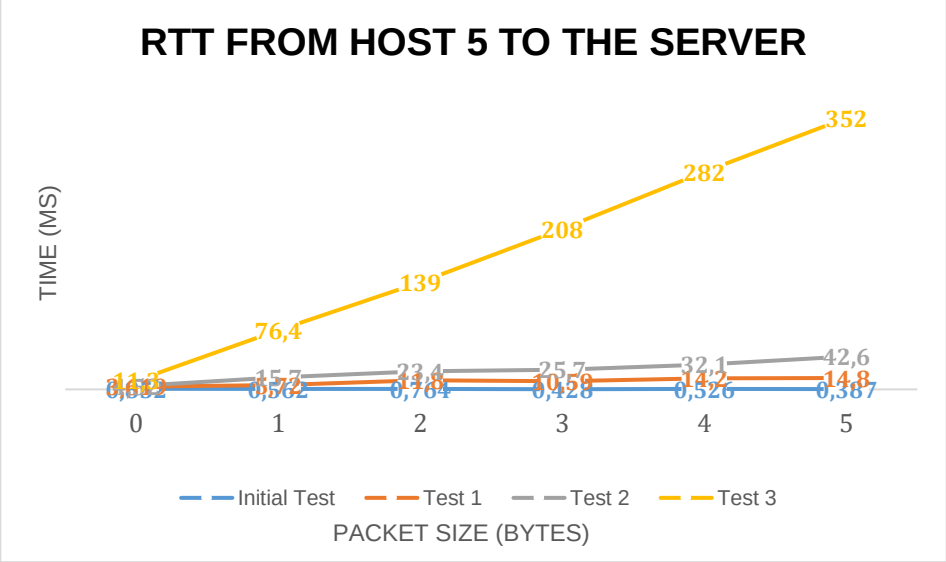


Source: own.

Now we proceed to analyze the RTT in communication with the server from the H5 equipment (10.0.0.5) which is a legitimate user located at one end of the network. For this test, a series of data is taken at different times with the three types of packets and also the time of the traffic of an initial test when there is no packet flooding in the SDN network as a reference point for the analysis, as a result the following analysis is obtained (see Figure 10 and Table 3).

For the initial test traffic the RTT did not exceed 0.852ms, once started with the packet flooding, the time increase for the three cases increased as the attack progresses, in time 5 the first packet obtained 14.8ms as maximum, the second packet was 42.6ms and the third packet of 1454 bytes presented a greater impact on the packet times reaching 352ms round trip considering that it is the case with less packets generated per second.

Figure 10. RTT of the packets to the server.



Source: own.

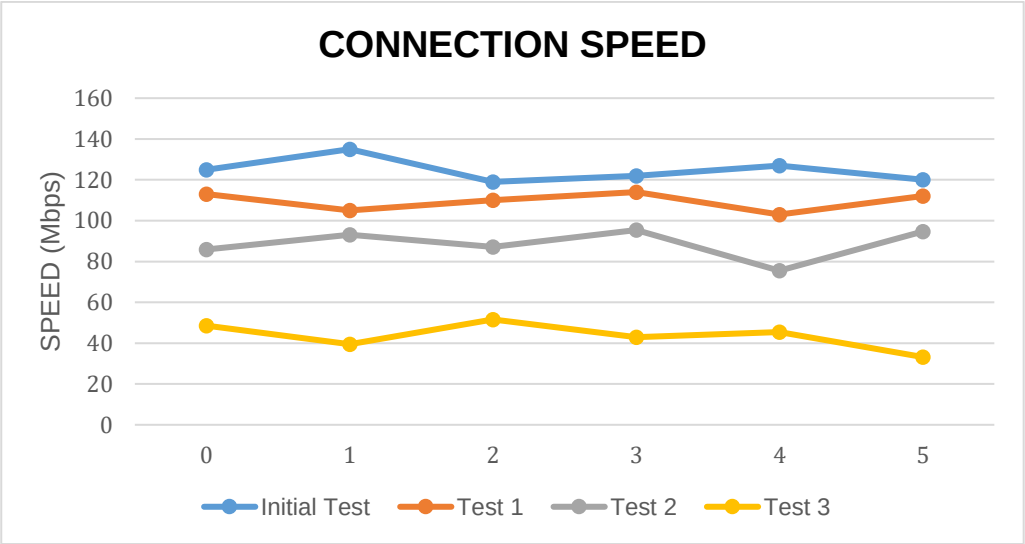
Table 3. Times of each test.

	Initial Test	Test 1 (54bytes)	Test 2 (554bytes)	Test 3 (1454bytes)
0	0.852	3.619	4.52	11.3
1	0.562	5.72	15.7	76.4
2	0.764	11.8	23.4	139
3	0.428	10.59	25.7	208
4	0.526	14.2	32.1	282
5	0.387	14.8	42.6	352

Source: own.

In addition, the analysis of the bandwidth taken from the H5 client (10.0.0.5) establishing connection with the web server (10.0.0.3) is performed, the data obtained are shown in Figure 11. It is evident that the best bandwidth is obtained when the server is not being flooded with forged packets being higher than 120 Mbps, but as the size of the packets increases it decreases becoming less than 50 Mbps for the case of the 1454 bytes packet.

Figure 11. Connection speed with the server.

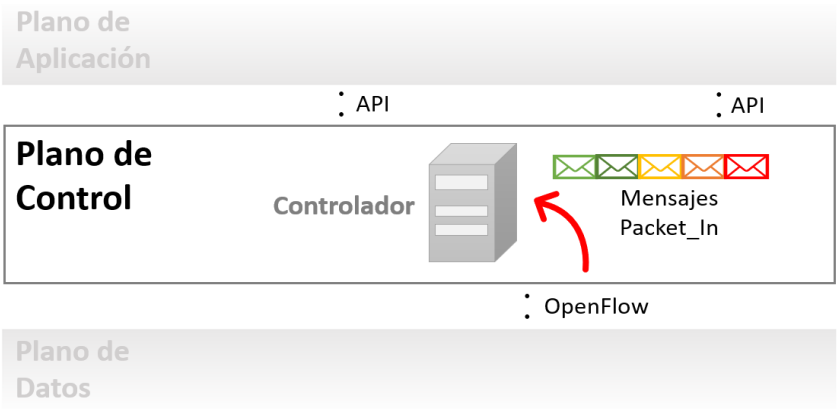


Source: own.

4.3. Vulnerabilities

According to the results and analysis of the data obtained, we proceed to identify the possible vulnerabilities of the network emulated in Mininet, which shows that the control plane is a critical point since it is located in the controller that makes the decisions of the entire network (see Figure 12), it is vulnerable because the attacker sends a large number of packets observed in Wireshark that will request the controller a new flow rule for each one in order to exhaust the performance of the network, as evidenced in the results of the tests applied.

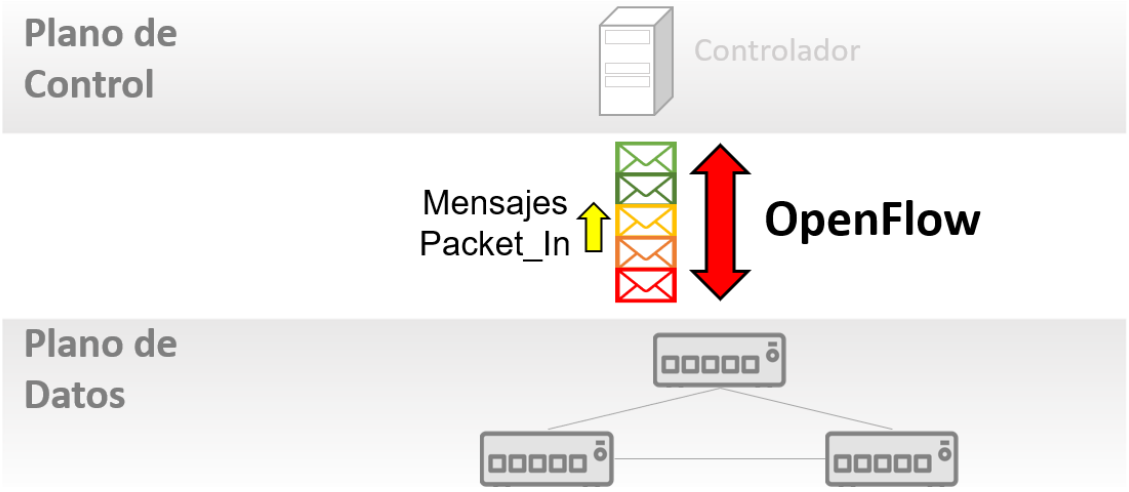
Figure 12. Saturation of the controller with Packet_In messages.



Source: own.

The link that communicates the control and data plane is another point, since this is used to send the "Packet_In" messages that are generated from the switches to the controller, by sending packets with IP spoofing, a message is generated for each new IP received by the switch.

Figure 13. Link congestion.



Source: own.

5. Conclusions

The implementation of the topology and execution of the tests has allowed observing the behavior of the SDN network against these types of packet flooding attacks that seek to take advantage of the decoupling of the planes to overload the network with packets in order to reduce its performance, affecting legitimate users on the network.

It is evident that in the data plane the packet flooding attack causes the switches to generate flow rule requests to the controller by means of "Packet_In" messages, as a consequence, the control plane is overloaded by the requests generated to create a new flow rule. As a result, there is a decrease in the connection speed with the Web server with values below 40Mbps for packet flooding of 1454 bytes and in the RTT there was an increase in times of up to 352ms compared to the initial test which was 0.852ms when there was no packet flooding.

Acknowledgments

This work was developed within the GISSIC research group. Product derived from the Maxwell research group funded by the Vice-Rectory of Research of the Universidad Militar Nueva Granada.

References

[1] VMware, "¿Qué son las redes definidas por software (SDN)? | Glosario de VMware | ES." https://www.vmware.com/es/topics/glossary/content/software-defined-networking.html

[2] Citrix, "¿Qué son las redes definidas por software (SDN)? - Citrix Mexico." https://www.citrix.com/es-mx/solutions/app-delivery-and-security/what-is-software-defined-networking.html

[3] M. Marchetti, "The road to riches," Sales Mark. Manag., vol. 150, no. 10, p. 128, 2013. https://doi.org/10.2307/j.ctvc77cz1.22

[4] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-Defined Networking Security: Pros and Cons," IEEE Commun. Mag., vol. 53, no. September, pp. 48-54, 2015. https://doi.org/10.1109/MCOM.2015.7120048

[5] A. Feghali, R. Kilany, and M. Chamoun, "SDN security problems and solutions analysis," Int. Conf. Protoc. Eng. ICPE 2015 Int. Conf. New Technol. Distrib. Syst. NTDS 2015 - Proc., 2015. https://doi.org/10.1109/NOTERE.2015.7293514

[6] J. Singh and S. Behal, "Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions," Comput. Sci. Rev., vol. 37, 2020. https://doi.org/10.1016/j.cosrev.2020.100279

[7] A. Pradhan and R. Mathew, "Solutions to Vulnerabilities and Threats in Software Defined Networking (SDN)," Procedia Comput. Sci., vol. 171, no. 2019, pp. 2581-2589, 2020. https://doi.org/10.1016/j.procs.2020.04.280

.