Security of Software Defined Networks (SDN) Against Flow Table Overloading Attack

Article *in* Journal of Advanced Research in Dynamical and Control Systems · October 2019

DOI: 10.5373/JARDCS/V11SP10/20192866

2 authors:

Rafal Nader Neamah
Al-Mustaqbal University College
1 PUBLICATION **2** CITATIONS
SEE PROFILE

Wesam S. Bhaya
University of Babylon, IT College
36 PUBLICATIONS **403** CITATIONS
SEE PROFILE

Security of Software Defined Networks (SDN) Against Flow Table Overloading Attack

Rafal Nader, Information Networks Dept., College of Information Technology, University of Babylon, Iraq.

Wesam Bhaya, Information Security Dept., College of Information Technology, University of Babylon, Iraq.

Abstract--- The fundamental innovation for Software Defined Network (SDN) is to dissociate the control plane from the data plane of the network devices and give central management over the network. SDN security is one of the serious challenges. Usually, the flow table of the SDN switches has a size limited due to the power-hungry and expensive features of Ternary Content Addressable Memory (TCAM), therefore makes it easier to be exploited by a flow table overloading attacks. The attacker has transmitted multiple flows to the controller, so the switch's memory (TCAM) has flooded with controller replies. To protect SDN from flow table overloading attack, we proposed an algorithm based on entropy variation of new incoming flows. As a result, the flow table overloading attack is detected through 250 packets with detection rate 95% and Precision 96% successfully.

Keywords--- SDN Security, Flow Table, Entropy, DDOS Attacks.

I. Introduction

Software Defined Networking (SDN) a novelty model for computer networking. Classical networks have focused hardware, while SDN has changed its oriented from the hardware to the software[1].SDN paradigm contains three layers, as to be seen in Figure1. Network administrators can deploy services by northbound APIs. Likewise, south bound APIs to communicate with the switches in the data plane. Generally,in southbound SDN protocols, the most utilization standard protocol is Open Flow[2].

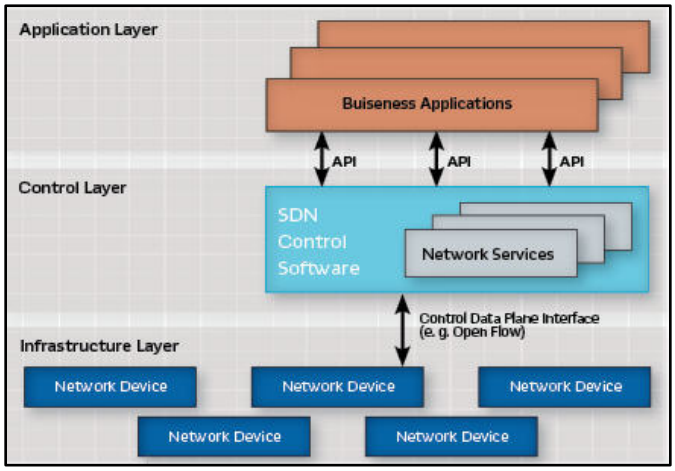

Figure 1: The SDN Frame Work [3]

In the Open Flow model, all the switches connected to the controller. Every switch holds a flow table. Assigning, modifying and omitting rules in flow table switches have controlled remotely by the controller. The switches carry out packet processing pursuant to rules of flow table switches [3], [4]Flow table of the switch has TCAM hungry; due it has contained relatively more fields in the header than conventional routers and switches. The flow table of Open Flow switch v1.0 has consisted of 12 fields for the header, consequently the size of the header more than 237 bits per flow entry. Hence, the flow table size often be attacked and can be destroyed by a flow table overloading attack will be converted to DDoS attack [5].

In the genuine world, attackers can use vulnerability computers to send Packet In messages to the controller. In the response of those messages, the switch has been flooded, and then new rules are not installed of Legal new packets due to TCAM's limited memory [5].

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

It is very tricky to distinguish attacks at data plane due Open Flow made switches have no intelligence, so we will try to detect flow table overloading attack in control plane. To fulfill this aim, a rapid and operative method has needed. Meanwhile, it must be lightweight to avoid exaggerated processing power usage, essentially at the peak of an attack.The proposal that submitted, the randomness of incoming flows has measured. The best way to measure randomness is entropy. It has measured the probability of an event occurring relative to the total number of events.In this paper, through the property of entropy, we design algorithms to detect flow table overloading attack and defend against this attack. Based on the experiments that will be performed on a real map, we will set threshold experimental for entropy, when entropy value less than the threshold, the attack will be detected.

II. Related Works

One of the important vulnerabilities is the restriction space of flow table in SDN has given a lot of interest in the researcher community. Miao et al.[6]presented Devo flow to Flow Management in the networks. They put simplest design changes to conserve flows in switches and keep going flow management effectively, by pushing flows back to the switches and inserting statistical information collection mechanisms.

Kanizo et al. [7] proposed the Palette to address the issue of rule placement, their aim to decrease the number of rules in switches. The concept of the palette, division policies and distribute them on all switches. But Kang et al.[8]solved problem distribute policies by making each path separately depending on metrics of the network.

Yuan et al.[5]proposed peer support strategy, which merges the obtainable idle flow table resource of the SDN to relieve flow table overloading attack (transformed DDoS attack) on a single switch of the system.

Suh et al. [9] presented a Content-oriented Networking Architecture (CONA), it executed in Open Flow platform. In CONA, the agents deliver requested contents to hosts and they keep track to requests, hence CONA will be effective against resource attacks.

Mehdi et al. [10] utilize the highest entropy appreciation mechanism to define the ordinary traffic distribution to repair network safety problems in office and home networks by using SDN. As well, Mousavi et al. [11] presented an entropy-based mechanism for fast disclosure of abnormal traffic.

III. Proposed Method

To get the best security mechanism, the method of detection and prevention flow table overloading attack must be executed in our network. Figure 2explains detection and prevention attack Process in general.

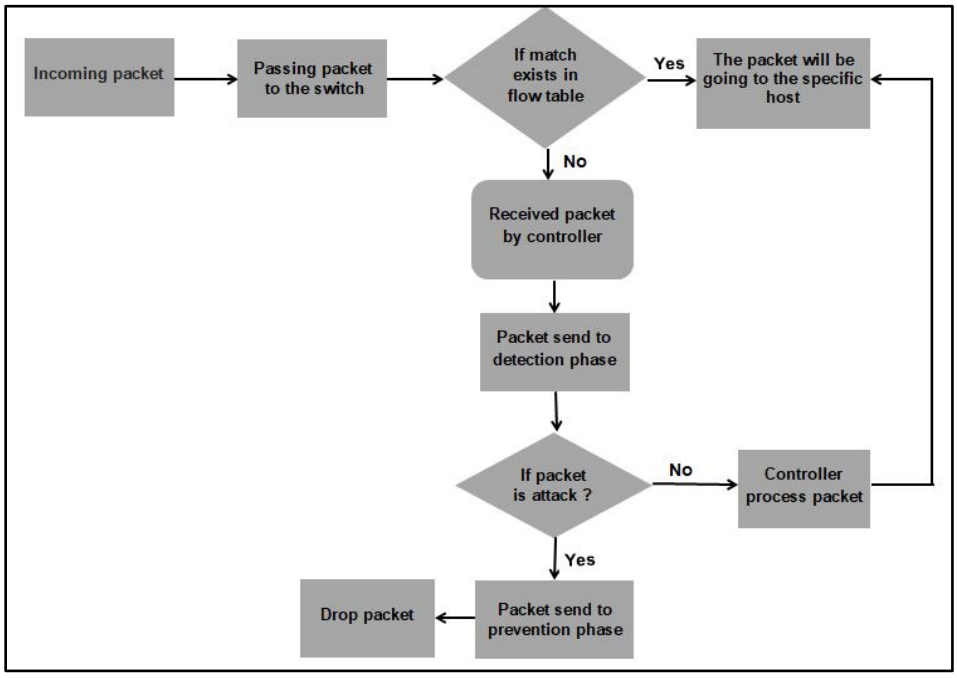

Figure 2: Detection and Prevention Attack Process

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

3.1Entropy Based Flow Detection Model

The main reason to choice entropy to detect flow table overloading attack, in order to the capability to measure randomness flows that are arriving at a network. If randomness of flows high, entropy value is high also and vice versa. To use entropy to detect the attack, we need two fundamental components. The first component window size has based on the number of packets or time duration; calculate entropy in each window to measure the uncertainty of incoming flows. The second component threshold, if entropy value less than the experimental threshold, overloading attack will be detected.Formula of entropyas seen below [11,eq.(1)].

$$H(X) = -\sum_{i=1}^{n} p(xi)\log_2 p(xi) \qquad (1)$$

where n is the number of flow packet in the window and p(xi) probability of each flow packet in the window. Hence, H(X) the result of entropy for all flow packet.

3.2 Utilizing SDN Specification

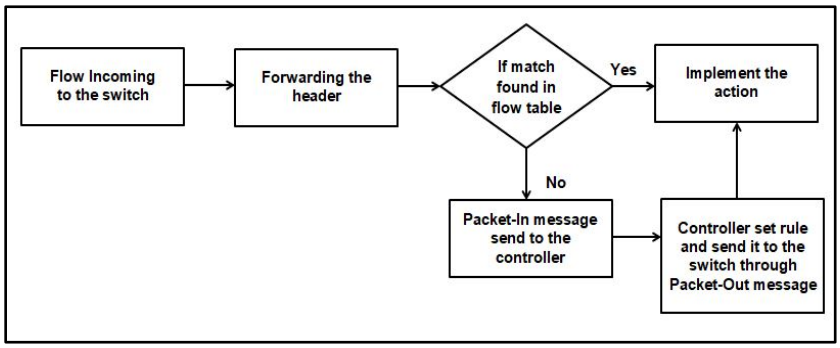

Figure 3: Packet Processing in SDN

For each new incoming flow packet to switch, it will search for a Match Fields in the flow table. If the packet is matched, the packet will be processed by the Action Set. Otherwise, the message will be transmitted to the controller called Packet-in. Then the controller determines how to handle the packet by putting the rule. Information rule includes (ip-address, mac-address, and port) and puts actions for processed this packet by drop or forwarding with priority by sending a packet-out message by the controller, as to be seen in Figure 3.The known fact about new flows coming to the controller destination host is within the network of the controller. Hence, the attacker can generate multiple flows to one host or more to instruct the controller to install rule in victim switch, thus the switch is flooded. As a result, the level of randomness can be found by computing the entropy based on the size of the window. If we have 50 flows and each flow transmit to a specific destination, so the probability of per destination address must be Pi $= \frac{1}{\text{pk}} = \frac{1}{50}$. The value of entropy H according to Eq.(1) must be $-\sum_{1}^{50} 0.02 \times \log_2 0.02 = 5.66$. But if 20 flows heading to one destination as a result entropy value decreased to 3.91.By using the entropy for collecting randomness of incoming flows in the SDN controller. On another hand, the controller has an overview of the network, through this we can evaluate the rate of flows incoming to controller and determine whether an attack is found or not.

3.3 Flow Table Overloading Attack Detection

For detection of overloading attack, we monitor the destination IP address of incoming flows by creating a hash table as a dictionary in python language. If dest_IP new in a hash table, this IP will be added to table with counter one, if IP has found in this table only the counter will be increased. After a hash table has contained on 50 packets, entropy value will be calculated. This mean window size equal to 50, the reason that, to get the best results according to the number of hosts found in the network. Another reason to calculate values faster without effect on CPU.

$$W = \{(a1, b1), (a2, b2), (a3, b3)\} \qquad (2)$$

Equation 2 represents a hash table, $a1$ means des_IP and $b1$ means frequency this des_IP and W is window size of the hash table.

$$Pi = \frac{ai}{\sum_{i=1}^{N} ai} \qquad (3)$$

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

Equation 3 represents the probability frequency of each des_ IP. Therefore we can calculate the value of entropy by equations 2, 3.Values of entropy relative stable in normal traffic, when there is victim switch in the network, the number of packets belong to hosts to the same switch will be increased. Hence, the values of entropy will decrease sharply. The detection scheme as described in Figure 4.This scheme involves two models. The first model for information gathering for not matched incoming flows and great hash table. The second model for detect flow table overloading attack. The steps of the scheme as shown in Algorithm.

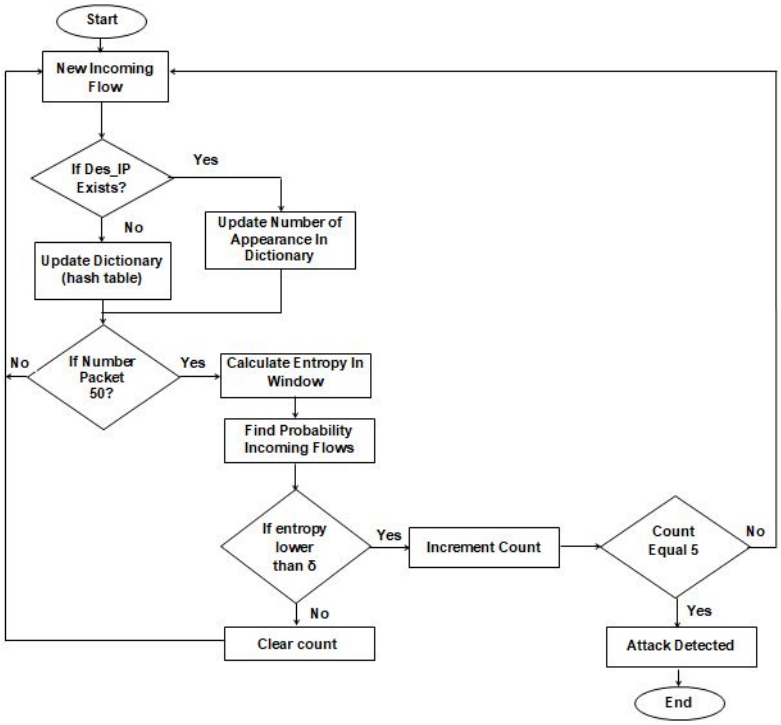

Figure 4: Flowchart of Proposed Flow Table Overloading Attack Detection

Algorithm Detection_Flow Table_Over loading_ Attack

01: **Step 1: Initialization**
02: Flow_In_Packet_Count = 0
03: Window Number (Win_Num) = 0
04: **Step 2: Waiting Flow_In_Packet to process**
05: Flow_ In_Packet_Count = Flow_ In_Packet_Count +1
06: If Des_IP found in Dictionary
07: Insert number from appearance in Dictionary
08: Else
09: Append new input in Dictionary
11: **Step 3: Test Window Size (Win_S) has been arrived**
12: Flow_In_Packet_Count % Win_S != 0
13: Go to step 2
14: **Step 4: Calculate Entropy (ENT) of Window**
15: Determination the threshold (**δ)**
16: If ENT <**δ**
17: Win_Num = Win_Num + 1
18: If Win_Num == 5
19: Alarm for attack progress
20: Else
21: Go to step 2
22: Else
23: Win_Num = 0
24: Go to step 2

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

A threshold experimental has chosen. If entropy value less than the threshold five times sequential, it means there is an attack on the switch. We have chosen five windows consecutive due its contain on lowest false positive, means that the authorized flow classified as an attack.

To defend against this attack, the proposed model has applied in the controller. This model creates the flow rule to block flows which belong to certain Des_IP.

IV. Experiment and Results

In this section, we will explain the tools used in the experiment setup, how to choose the threshold and results based on the value of the threshold.

4.1 Experiment Setup

The first department of our experiment is to choose the mininet that is a network emulation tool for SDN. Mininet built into Linux that used features lightweight virtualization in the Linux kernel, such as CPU bandwidth isolation, process groups, and merge them with virtual Ethernet links. Therefore the system becomes faster and contains more hosts than another emulator [12].The controller used in our experiment is POX controller. It is fast, lightweight, and it is programmed in Python language. POX enhanced version of its predecessor NOX controller [13].The proposed SDN topology is Aconet Austria topology as shown in Figure 5. This consists of 11 Open Virtual Switch (OVS) and 66 hosts, due each switch has 6 hosts connected to it.

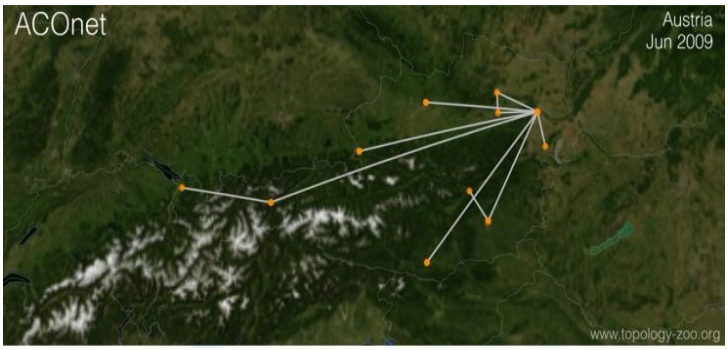

Figure 5: SDN Topology

To generate real attacks we used the Scapy tool [14] as a completely powerful tool. In our experiment, to overflow switch we need to generate massive of Packet_In messages. To get these messages we use Scapy.

Two various python scripts have written the first for legitimate traffic has des_IP started from 10.0.0.1 to 10.0.0.66, the second for attack traffic.

In this Scenario to make single switch victim in the network host 10.0.0.1 generate normal traffic, host 10.0.0.2 and host 10.0.0.3 generate attack traffic to two hosts belong to victim switch.

4.2 Test and Results

In our solution to detect flow table overloading attack, we monitor values of entropy. If values less than the existing threshold, and they continuous five successive windows, means that attack is found.

In order to find an appropriate threshold, we implemented the code 11 times to show the influence of an attack on the entropy. The highest value entropy during attack intervals considered an appropriate threshold. Then we stop execution and put the threshold value. We notice False Positive has sharply reduced, and False Negative has kept within a fixed range, but False Negative not effect in the flow table of the switch, because rules of packets will be removed after a period. Table1 represents the threshold value chosen when the attack at a rate of 80%.

To get an appropriate threshold value, we have executed the following:

- At first, we compute the minimum confidence interval in a normal case. This can be accomplished by the mean entropy of normal traffic minus confidence interval.
- Then, we compute the maximum confidence interval in the attack case. This can be accomplished by the mean entropy of attack traffic plus confidence interval.
- After that we notice the difference in values, hence we determine the threshold.

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

Table 1: Find Optimal Threshold

Measures	Normal-Traffic	Attack-Traffic 80%
Mean Entropy Values	0.8654	0.5395
Standard Deviation	0.0500	0.0777
Confidence Interval	0.0014	0.0023
Confidence-Min	0.8640	0.5372
Confidence-Max	0.8668	0.5418
Threshold value	0.5539	

In spite of previous calculations shows that 0.5418 could be the threshold, but after running the code for eleven times. We noticed 0.5539 is the optimal threshold; due it has less false negatives.

To test this solution, we sent 5,000 packet_In to controller contain normal and attack packets, to get for an attack in rate 80%, we sent 4000 packets attack from out 5000 packets. We notice through 250 packets there is an attack on victim switch. In Figure 6we capture for some packets toward to victim switch, we notice overload on two interfaces belong victim switch.

Figure 6: Effect on Interfaces of Victim Switch

The horizontal line and the vertical line of Figure 7 are denoted to the window and entropy value of each window in respectively. The blue line curve denotes legitimate traffic and red line curve denotes attack traffic.

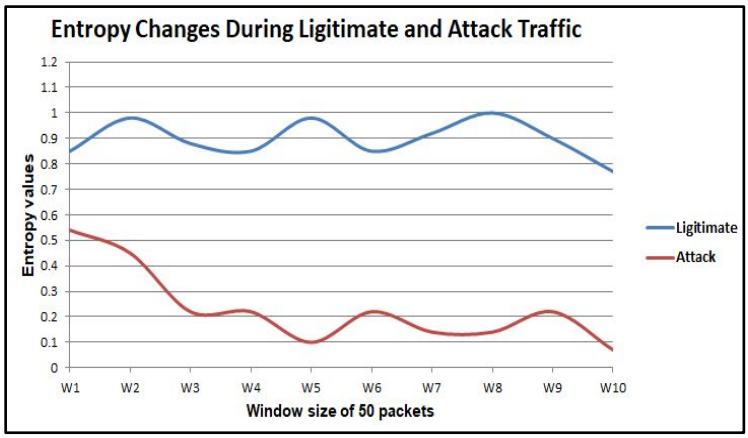

Figure 7: Entropy Changes During Legitimate and Attack

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

Detection results will be explained in Table 2.

Table 2: The Detection Results

Measure	Value
Sensitivity	0.8156
Specificity	0.9711
Precision	0.9667
Negative Predictive Value	0.8365
False Positive Rate	0.0289
False Discovery Rate	0.0333
False Negative Rate	0.1844
Detection Rate	0.9500
F1 Score	0.7949

V. Conclusion

Flow table overloading attack is a serious threat in SDN; It is very significant to recognize the attack before it occurs. In this paper, we presented a method a rapid and operative to detect and prevent the attack in SDN. Due the controller has an overview of the network; we were able to use the destination IP of flows incoming to the controller and determine whether an attack is found or not, within the first 250 packets. One of these advantages of the presented method is flexibility. Any parameter such as size of the window and threshold can be modified to correspond to the given values by the admin of the network. The results of propose method denote the value of False Positive has very fewer, and False Negative has kept within a fixed range, but False Negative not effect in the flow table of the switch, because rules of these attack packets will be removed after a period.

References

[1] T. Jamal, P. Amaral, and K. Abbas, "Flow Table Congestion in Software Defined Networks,"vol.57, pp. 48–53, 2018.

[2] D. Kreutz, F.M.Romos, P. Verissimo, C.E. Rothenberg, S. Azodolmolky, and S.Uhlig, "Software-Defined Networking : A Comprehensive Survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan, 2015.

[3] A. Apostolov, "SDN-Software Defined Networking," pacw.org, June. 2016. [Online]. Available: https://www.pacw.org/nocache/issue/june_2016_issue/news/technology_news/sdn_software_defined_networking.htm. [Accessed: July.14, 2016].

[4] N. Ruchansky and D. Proserpio, "A (not) NICE way to verify the openflow switch specification," *Journal of ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 527–528, Oct.2013.

[5] H. Jin, J. Shen, S. Yu, D. Zou, W. Qiang, and B. Yuan, "Defending against Flow Table Overloading Attack in Software-Defined Networks," *Journal of IEEE Transactions on Services Computing*, pp. 1–14, Aug.2016.

[6] A.R. Curtis, J.C. Mogul, J. Tourilhes, P. Yalagandaula, P. Sharma, and S. Banerjee, "Devoflow: scaling flow management for high-performance networks," *Journal of ACMSIGCOMM Computer Communication Review*, vol.41, no.4, pp. 254-265, Aug. 2011.

[7] Y. Kanizo, D. Hay, and I. Keslassy, "Palette: Distributing tables in software-defined networks," *Journal of IEEE INFOCOM*, pp. 545–549, 2013.

[8] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the ' One Big Switch ' Abstraction in Software-Defined Networks," *In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, ACM,2013.

[9] J. Suh, H. Choi, W. Yoon, T. You, T. T. Kwon, and Y. Choi, "Implementation of Content-Oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure," *Journal of European Net FPGA Developers Workshop*, pp. 1–5, 2010.

[10] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting Traffic Anomaly Detection Using Software Defined Networking," *Journal of Springer*, pp. 161–162,Sep. 2011.

[11] S. M. Mousavi and M. St-hilaire, "Early Detection of DDoS Attacks against SDN Controllers," *In International Conference on Computing, Networking and Communications (ICNC)*, pp. 77–81, 2015.

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019

[12] R. Oliveira , C. Schweitzer , A. Shinoda and L.Prete, "Using Mininet for emulation and prototyping Software-Defined Networks," *in IEEE Colombian Conference on Communications and Computing (COLCOM)*, pp.1-6, 2014.

[13] N. Gude, T. Koponen, J .Pettit, B. Pfaff, M. Casado, N. Mckeown, and S. Shenker "NOX: towards an operating system for networks," *Journal of ACM SIGCOMM Computer Communication Review*, vol.38, no.3, pp.105-110, July.2008.

[14] S. Bansal, and N. Bansal, "Scapy-a python tool for security testing," *Journal of Computer Science& Systems Biology*,vol.8,no.3, p.140,May.2015.

DOI: 10.5373/JARDCS/V11SP10/20192866
ISSN 1943-023X

Received: 20 Aug 2019/Accepted: 17 Sep 2019