IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 20, Issue 5, Ver. I (Sep - Oct 2018), PP 66-72
www.iosrjournals.org

DDOS Anomaly Detection And Mitigation In Software Defined Networking (SDN)

Abdul-Wadud Alhassan [1], Adebayo B. Salaudeen[2]
[1]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[2]School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
Corresponding Author: Abdul-Wadud Alhassan

Abstract: *The introduction of Software Defined Networking as a panacea to the global demand for a more secure and highly dependable internet infrastructure has also brought along security issues. The adoption of OpenFlow Protocol (OFP) by SDN as the way of communication between controllers and switches has not only brought about easy and direct manipulation of data for enhanced packet forwarding policies, but also renders the network vulnerable to security issues (DDOS attacks) since the OF switch has to ask the controller to install new rules for any new incoming packet. In this paper, we prove that SDN is capable of handling security threats that arise from the above vulnerability. We implement a DDOS detection module that detects and blocks SYN flood attacks aimed to destabilize the flow of normal network traffic among users in a software-defined networking environment.*

Keywords: *DDOS detection , Mininet, OpenFlow, Software Defined Networking, Security, SYN flooding*

Date of Submission: 14-09-2018 Date of acceptance: 29-09-2018

I. Introduction

The ever-increasing global demand for a secure and highly dependable information technology infrastructure has created the need for constant improvements to the current internet structure. Today, the rise of cloud computing, big data as well as the internet of things has created the need for a more secure and dependable network infrastructure [1]. However, there`s an issue of state-of-the-art security attacks and also unreliable internet services due to performance related issues in our current traditional network structure. This is due to the complexity of configuring current traditional network devices to respond to updates, as well as faults and loads. Therefore, the provision of a secure and reliable internet services has now become the focus of several organizations worldwide. Various internet approaches have been proposed and are been implemented by some of these organization. One of such implementations is Software Defined Networking (SDN) by The Open Networking Foundation (ONF) which is an emerging architecture that is dynamic, manageable, cost-effective and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications [1]. SDN breaks the vertical integration of traditional networks that bundles the control and data plane inside networking devices by decoupling the control plane from the data plane. This reduces network complexity and also enhances decision making as well as network configuration [2].

SDN employs the Open Flow protocol (OFP); the first and widely used standardized protocol which defines the way of communication between controllers and switches by adopting the concept of flows to determine traffic flow based on a per-flow basis through network devices [1]. This allows for direct manipulation of data in order to enhance policies for packet forwarding, unlike current traditional networks. In a typical SDN environment, the controller manages a set of flow tables in an OF switch. These flow tables contain flow entries made up of counters and match fields as well as a set of actions to apply to matching packets. All incoming packets to an OF switch are compared with its flow table entries to check for a matching rule, if a matching rule is found, the instructions associated with the specific flow entry are applied. If no matching rule is found for an incoming packet, the packet is forwarded to the controller inside a packet-in message to ask for further actions [3]. This makes the OFP significant for Software Defined Networking. However, this renders the network vulnerable to security as well as scalability issues since the OF switch has to ask the controller to install new rules for any new incoming packet. Distributed Denial of Service (DDOS) attackers can exploit this process by sending a large number of anomalous packets with different header fields, the OF switch upon receiving these packets forwards them to the controller for further actions since no matching rules are found. This leads to the consumption of resources that prevents the controller from answering requests from legitimate users [4]. Hence, the need for more work to be done on the security aspects of SDN. Also, the merits of SDN over the traditional networks render it prone to various forms of security threats [5].

Therefore, this paper seeks to address the issue of anomalous flow behavior in the form of DDOS SYN-flood attacks intended to destabilize the flow of normal network traffic among users in a software-defined networking environment. We implement a system that detects and mitigates anomalous flows sent from malicious hosts towards victim machines.

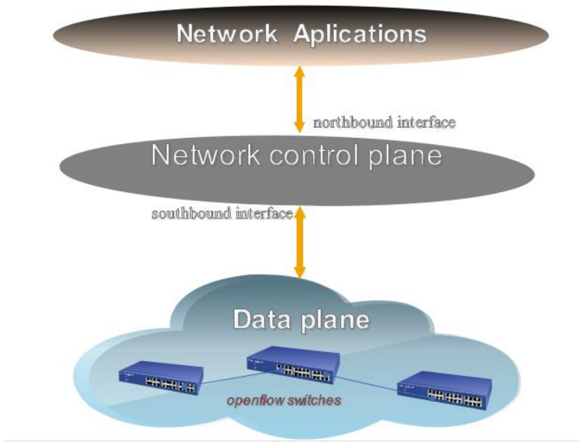


Figure 1: A simplified SDN architecture

The general layout of the paper is as follows: section II gives an insight of related works on some security aspects of SDN, section III describes the methodology of our proposed system whiles section IV details the experiments carried out in this paper. Also, our results and evaluation come in section V and we conclude in section VI.

II. Related Works

[6] Investigated how SDN can be used to overcome and block legitimately looking DDOS attacks mounted by a large number of bots. They constructed a system that effectively blocks a botnet-based DDOS attack by using the standard OpenFlow interface. In [7], the authors presented an autonomous detection system which characterized traffic attributes of six flows to provide real-time detection and mitigation. Their system is able to perform constant traffic monitoring by analyzing packet flows through traffic monitoring. SLICOTS, an effective and efficient countermeasure to mitigate TCP SYN flooding attack in SDN was proposed by [4] implemented on a GENI [8] virtual testbed. SLICOTS blocks malicious hosts by surveying ongoing TCP connection. The author in [9] also proposed LineSwitch, an efficient and effective solution against control plane saturation attack by combining SYN proxy techniques and probabilistic blacklisting of network traffic. Several authors also designed firewalls that block unwanted hosts from communicating with others in the network. [10] and [11] are among a few who implemented firewalls in SDN. The limitation of these firewalls is that the system does not keep any state of information to track down ongoing connections.

III. Methodology

Security-related anomalies occur due to malicious activities that attempt to disrupt the normal functioning of the network. One of such anomalies is in the form of DDOS attacks where the attackers attempt to victimize a system or a service by sending mass requests, beyond the capacity of the system or the service [12]. The security methods of our current traditional network system like firewalls, load-balancers etc. tend not to be reliable in dealing with today`s DDOS attacks, hence the need to for SDN security [13].

In this paper, we design a system in which the SDN controller observes traffic from multiple hosts and blocks incoming flows from hosts that do not conform to the normal TCP three-way handshake. We test our system by introducing SYN-flood attacks sent from a malicious host with the intent of causing data to control plane saturation in an SDN.

1.1. TCP- SYN Flood

Introduced by Zeigler [14], TCP-SYN flood is a type of DDOS attack in which attackers send a lot of SYN packets to their target exhausting the targeted servers' resources thereby rendering it unresponsive to legitimate traffic. The attacker creates many half-open TCP connections without sending ACK messages required to complete the normal TCP three-way handshake connection between source and destination [4]. In the case of SDN, SYN flood attacks aim to saturate the controller by generating a huge number of new network flows using spoofed IDs by taking advantage of communication between the data and control plane [9]. OF switches request necessary actions to be taken on each new SYN packet sent from hosts by contacting the controller which then analyzes the flow information, prepares and sends a response (forwarding rule) to the OF switch. The OF switch, in turn forwards the SYN packet to the target host. The host responds with a SYN-ACK back to the OF switch which again forwards to the controller requesting a forwarding rule. The controller responds by installing forwarding rules on the OF switch which finally forwards the packet to its destination. Therefore, in a SYN flood attack, since the controller will be busy installing forwarding rules to serve forged IPs, control plane saturation can occur rendering the controller impaired to respond to legitimate network flows. Figure 2 shows the necessary requirements for completing a normal TCP three-way handshake connection whiles Fig. 3 depicts how a SYN flood attacks occur in SDN as explained above.

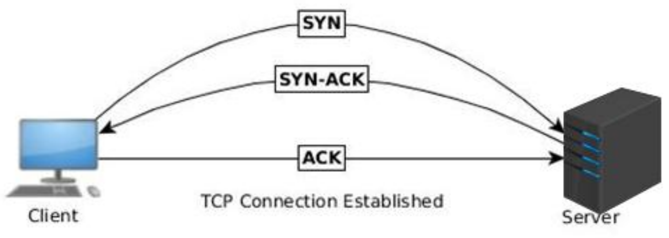


Figure 2: Normal TCP three-way handshake

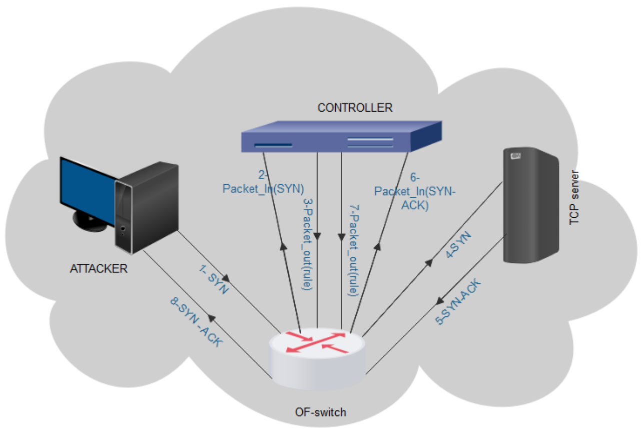


Figure 3: SYN flood in SDN

In order to implement our anomalous flow detection module, we setup up an SDN virtual network using the following:
VMWARE workstation 12 which provides a suitable environment for installing virtual machines.
Mininet VM: a network emulator used in creating realistic SDN networks.
POX controller: a python based SDN controller.
We set up the VM (Mininet emulator version 2.2.1) on an ubuntu 14.04 server as guest operating system installed on a VMWARE workstation 12 virtual machine. The virtual machine is running on a 64-bit windows 10 operating system. Using python, we created two scripts, one to launch a SYN-flood attack (Attack.py) and one for detection (Detection.py). Then we also create a network topology that consists of an OpenFlow switch, remote controller (POX) and 9 hosts each assigned IP addresses. After successfully setting up our network topology and running a successful ping command to test connectivity between hosts, we start our detection module on the controller alongside a layer 3 learning switch (l3_learning) using: ***./pox.py forwarding.l3_learning detection.*** After that, we open a host terminal and run our attack with the command *sudo python attack.py*. Our DDOS detection system acts based on a threshold of 50 packets per second. Therefore, it blocks IP addresses that send more than 50 SYN requests per second believed to be SYN-FLOOD attacks. Figure 4 illustrates the nature of our network topology whiles figure 5 shows how our detection system handles an attack from a host.

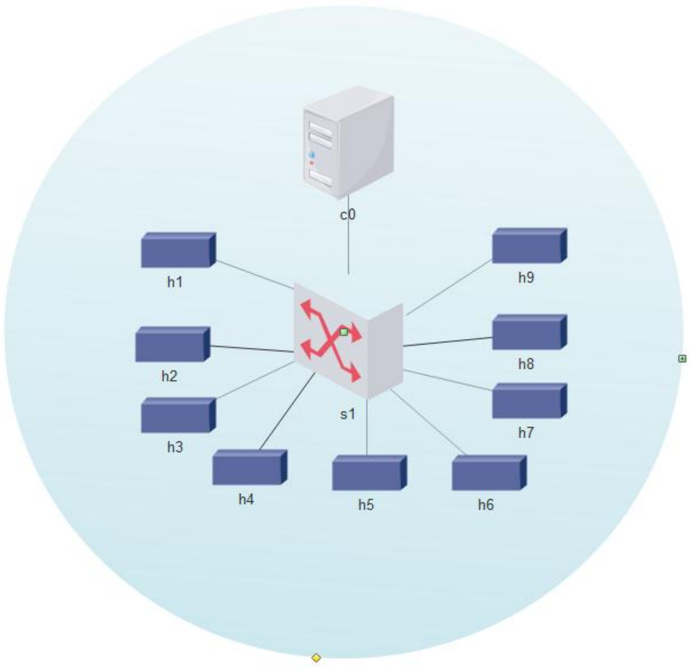


Figure 4: Network topology

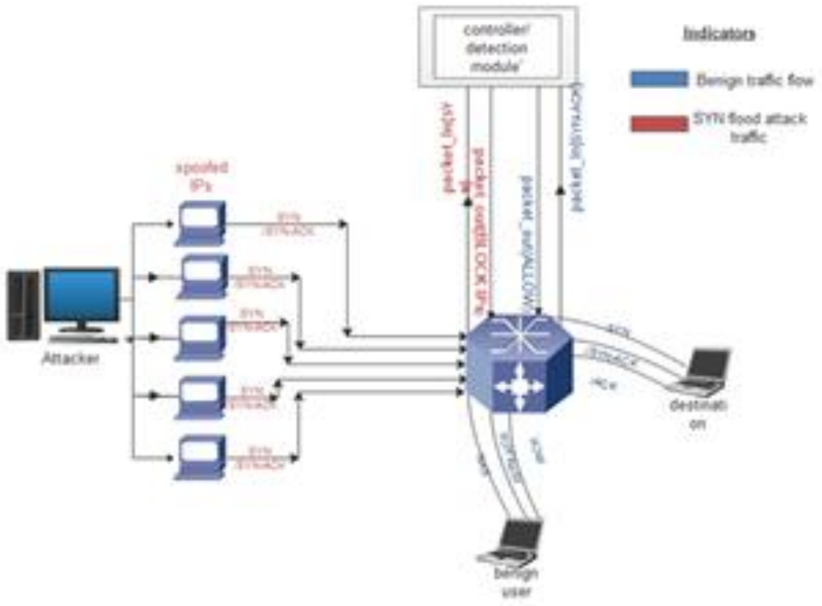
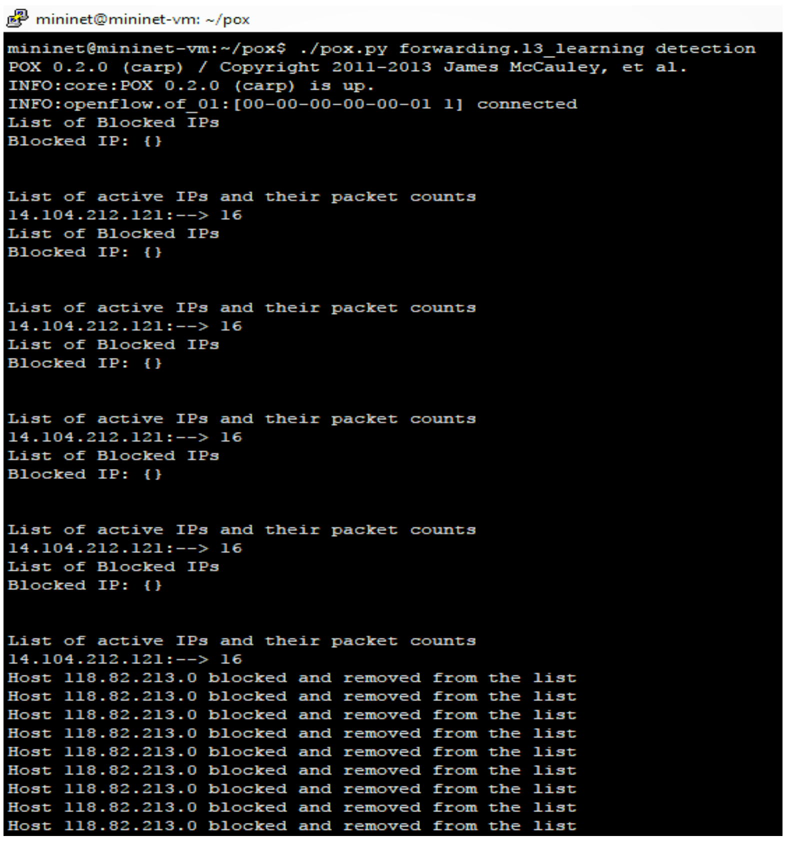


Figure 5: Topology setup on mininet VM

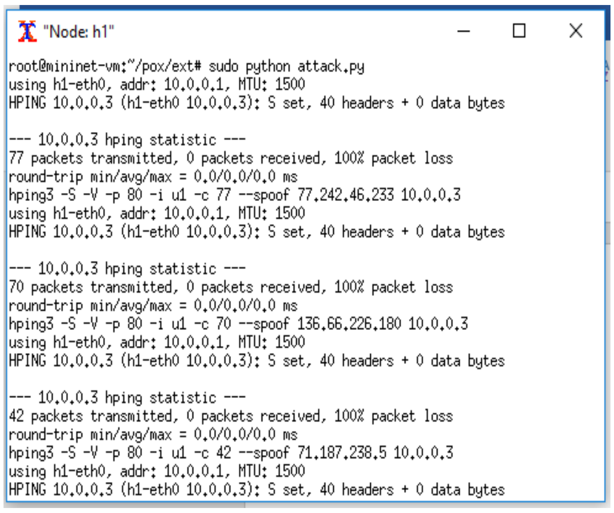
IV. Results

Fig. 5 shows that our detection module successfully blocked anomalous packets sent from a malicious host believed to be attempting a SYN flood attack while figure 6 shows the attack been launched by the malicious host (host 1). We also tested the network performance (bandwidth, latency, RTT) of our detection module running with the controller before and during the attack and also the performance of the controller running on only normal l3_learning.



```
mininet@mininet-vm: ~/pox
mininet@mininet-vm:~/pox$ ./pox.py forwarding.l3_learning detection
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
INFO:core:POX 0.2.0 (carp) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 1] connected
List of Blocked IPs
Blocked IP: {}

List of active IPs and their packet counts
14.104.212.121:--> 16
List of Blocked IPs
Blocked IP: {}

List of active IPs and their packet counts
14.104.212.121:--> 16
List of Blocked IPs
Blocked IP: {}

List of active IPs and their packet counts
14.104.212.121:--> 16
List of Blocked IPs
Blocked IP: {}

List of active IPs and their packet counts
14.104.212.121:--> 16
List of Blocked IPs
Blocked IP: {}

List of active IPs and their packet counts
14.104.212.121:--> 16
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
Host 118.82.213.0 blocked and removed from the list
```

Figure 6: detection module running with a layer 3 learning switch on pox controller



```
"Node: h1"
root@mininet-vm:~/pox/ext# sudo python attack.py
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.3 (h1-eth0 10.0.0.3): S set, 40 headers + 0 data bytes

--- 10.0.0.3 hping statistic ---
77 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
hping3 -S -V -p 80 -i u1 -c 77 --spoof 77.242.46.233 10.0.0.3
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.3 (h1-eth0 10.0.0.3): S set, 40 headers + 0 data bytes

--- 10.0.0.3 hping statistic ---
70 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
hping3 -S -V -p 80 -i u1 -c 70 --spoof 136.66.226.180 10.0.0.3
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.3 (h1-eth0 10.0.0.3): S set, 40 headers + 0 data bytes

--- 10.0.0.3 hping statistic ---
42 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
hping3 -S -V -p 80 -i u1 -c 42 --spoof 71.187.238.5 10.0.0.3
using h1-eth0, addr: 10.0.0.1, MTU: 1500
HPING 10.0.0.3 (h1-eth0 10.0.0.3): S set, 40 headers + 0 data bytes
```

Figure 7: SYN flood attack from host 1

1.2. Bandwidth

Iperf was used to determine the bandwidth (network throughput) of our DDOS detection system compared with that of the normal host-switch-controller connection without a detection system. We run 4 Iperf commands from different hosts in both scenarios. The results in Fig 6 show that the difference in network throughput between our detection module and that of the host-switch-controller connection is not that much. But our detection consumes less bandwidth compared with the later.

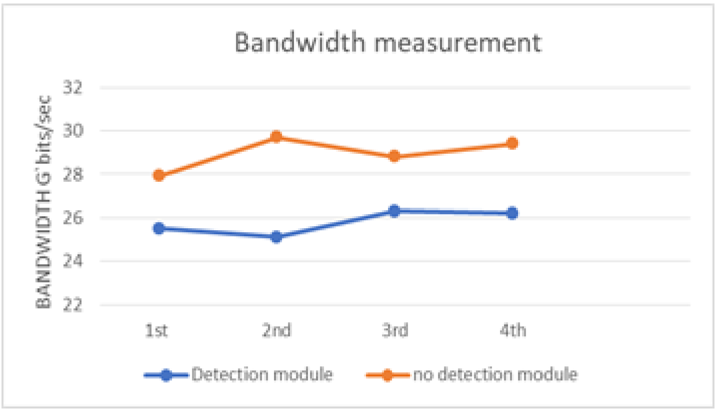


Figure 8: Network throughput

1.3. Round Trip Time(RTT)

We determined the average RTT of our simulation by sending ping commands from four hosts in different scenarios. The first was sending ping commands from the hosts with our detection module running alongside an l3. learning switch on the controller whiles at the same time we launched our SYN flood attack. Secondly, we launched our attack from one host and also sent ping commands from four other hosts. Figure 7 shows that, the average RTT was very high in our detection module. This is due to the fact that, our detection module inspects packets sent from hosts through the OF switch to the controller before the controller installs forwarding rules on the switch. This causes a delay since anomalous packets were detected and had to be blocked in order to allow for benign network traffic flow. But with the pox controller running alongside l3. learning switch without a detection module, we observed that the RTT was very low.

This is due to the fact that, the normal forwarding rule installed on the switches by the controller allowed free packet flow from the host, hence making the network susceptible to SYN flood attacks.

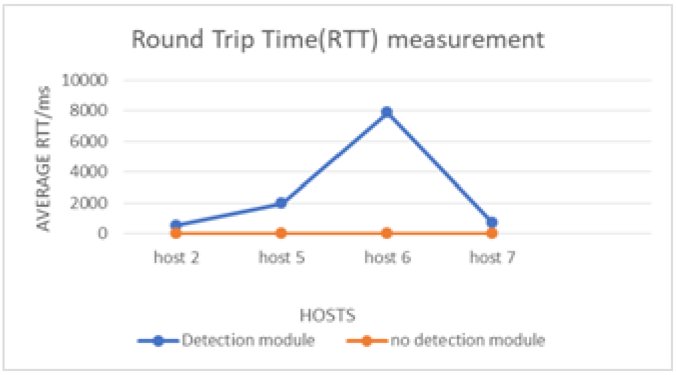


Figure 9: RTT measurement

V. Conclusion

In this article, we proved SDN is capable of handling security related issues that have been a cause for worry in our current traditional networks by implementing a software-defined network-based system that detects and blocks anomalous SYN-flood packets sent from hosts by using a threshold based on the number of packets sent per second. Our system proved to handle the issue of bandwidth consumption a little more than the normal connection between host-switch and controller module in the event of a SYN attack. We also observed that our detection module causes a delay in RTT which is a setback especially in the case where attackers, as well as benign users, send a greater number of packets per second. In our future work, we intend to apply machine learning methods to our detection module in order to handle SYN flood attacks efficiently. And also, we intend to implement our system on a large scale using a virtual testbed that supports large scale SDN experiments.

Acknowledgements

I would like to thank my supervisor Prof. Wei Sonjie for his assistance in this work.

References

[1]. Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. *White paper, Open Networking Foundation,* Palo Alto, CA, USA, 2012.
[2]. D. Kreutz, F. M., V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE,* 2014, vol. 103, no. 1, pp. 14-76.
[3]. Openflow Switch Specification. Version 1.1.0 Implemented (Wire Protocol 0x02), 2011.
[4]. R. Mohammadi, R. Javidan, and M. Conti. SLICOTS: An SDN-Based Lightweight Countermeasure for TCP SYN Flooding Attacks. *IEEE transactions on Network and Service Management*, 2017, vol. 14, no. 2, pp 487-497.
[5]. L. Schehlmann, S. Abt and H. Baier: Blessing or Curse? Revisiting Security Aspects of Software-Defined Networking. *IEEE 10th international conference on Network and Service Management.* 2014.
[6]. S. Lim, J. Ha, H. Kim. Y. Kim, S. Yang. A SDN-Oriented DDoS Blocking Scheme for Botnet-Based Attacks. *IEEE*, 2014.
[7]. L. F. Carvalho, G. Fernandes Jr., Joel J. P. C. Rodrigues, Leonardo S. Mendes, Mario Lemes Proenc¸a Jr. A Novel Anomaly Detection System to Assist Network Management in SDN Environment. *IEEE,* 2017.
[8]. M. Berman, J. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskarf, "GENI: A federated testbed for innovative network experiments*," Computer Networks,* Vol. 61, 2014.
[9]. M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks*," in Proc. 10th ACM Symp. Inf. Comput. Commun. Security,* Singapore, 2015 pp. 639–644.
[10]. C. N. Shivayogimath1, N.V. Uma Reddy, "Modification of L3 Learning Switch Code for Firewall Functionality in Pox Controller (Working on SDN With Mininet*), IJRET: International Journal of Research in Engineering and Technology,* 2015*, Vol 04 Issue: 06,* pISSN: 2321-7308.
[11]. W. M. Othman, H. Chen, A. Al-Moalmi and A. N. Hadi, "Implementation and performance analysis of SDN firewall on POX controller, " *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, 2017, pp. 1461-1466.
[12]. D. Kumar, B. Jugal, K. Kalita*: Network Anomaly Detection: A Machine Learning Perspective* (Taylor & Francis Group, LLC. 2014).
[13]. T. T. Huong, N. H. Thanh. Software Defined Networking-based One-Packet DDoS Mitigation Architecture. *IMCOM '17 Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication,* 2017, article no 110.
[14]. B. Ziegler, "Hacker tangles Panix Web site," *Wall Street J.,* 1996, vol 12.
[15]. Mininet, *http://mininet.org/overview.*

* Abdul-Wadud Alhassan . " DDOS Anomaly Detection And Mitigation In Software Defined Networking (SDN)." IOSR Journal of Computer Engineering (IOSR-JCE) 20.9 (2018): 66-72.