Research Article

Ddos Attack Detection Using SDN Techniques

B V Baiju[a], S Mohamed Yahiya[b], P Akash Raj[c], S Sulman Farooq[d]

[a,b,c,d] Department of Information Technology, Hindustan Institute of Technology and Science, Chennai, India.
[a] bvbaiju@hindustanuniv.ac.in, [b] yahiyabinsiraj@gmail.com, [c] akashfrnds43@gmail.com, [d] Sulmanfarook531@gmail.com

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: Distributed denial-of-service (DDoS) attack is a favourite weapon for black hat hackers and cyber terrorists. In spite of the vast choices of traditional solutions which is available today that can mitigate the DDoS attacks, they are still growing to become more frequent, upscaled, and severe. These attacks can quickly malfunction websites ranging from a single webpage to bigger web applications that belong to famous organizations which can cause huge financial and reputation wise risks and losses. Hence there forms a requirement to follow a new network paradigm which can detect and mitigate such attacks. Software-defined networking is such an evolving paradigm which can reduce the network expenses and it is also potent to detect and mitigate DDoS attacks. We are inspired by the capabilities of SDN, and we are going to detect DDoS attack that can occur in SDN. We will virtualize a network that is built with Mininet, followed by linking the root switch that handles the whole network to an SDN controller (like RYU, NOX etc.) and an Intrusion Detection System (IDS e.g., SNORT). We use OpenFlow as the communication protocol for the SDN controllers to communicate with the virtual network. This system can detect the DDoS attack based upon SDN techniques and not based on traditional methods(like firewall rules).

Keywords: Distributed Denial of Service; Software-defined networking; Mininet; RYU;SNORT; OpenFlow; Intrusion Detection System.

1. Introduction

A distributed denial-of-service (DDoS) attack is conducted to block the usual traffic of the victim server, service or network by profuse the target or its surrounding infrastructure with a flood of malicious packets. DDoS attacks attain the viability by exploiting multiple computer systems that are already compromised/ acquired the access and deploying them as the source of overwhelming network traffic. Compromised devices include PCs, laptops and other devices that involve in network such as IoT devices. In layman’s terms, a DDoS attack can be compared with an unpredicted traffic jam blocking the highway, which prevents the regular traffic and hinders the vehicle movement.

Distributed denial-of-service (DDoS) attacks have been a real threat for cyber security, which in turn have been a threat for cloud security. These attacks are capable to inflict huge interference in the communication infrastructure. There are various reasons for carrying DDoS attacks over. These include hacktivism, industrial competition, disgruntled employee revenge etc. DDoS attacks can incapacitate networks and services by shattering the network devices (routers, switches, etc.) and servers with huge and malicious traffic. They can either cause the service to degrade or a total service denial resulting in a heavy loss, be it financial or reputation wise. Increasing dependency on data centre and internet has exasperated this problem. [1] Data centres that run critical services like smart grid etc., need to be safeguarded in order to provide hassle-free and highly dependable services.

Software-defined networking (SDN) is a network approach which uses application programming interfaces (APIs) or software-based controllers to interface with the primary hardware network and drive the traffic on a network. This model differs from the old school networks. Traditional networks use legacy hardware components (switches and routers) which control the network traffic. SDN can create and control a virtual network. It can control the hardware devices via software. Network Virtualization is the key factor for SDN that attracts the network personnel. While network virtualization allows organizations to split various virtual networks and cluster it into a single physical network, or to connect devices on different physical networks to create a single virtual network, software-defined networking enables a new way of controlling the routing of data packets through a centralized server.

OpenFlow (OF) is considered one of the first standards for software-defined networking (SDN). It defines the communication protocol for the SDN controller to communicate directly with the forwarding planes (like switches and routers) be it physical or software-based (virtual). Giotis K *et. al.* [2] uses this OpenFlow in SDN with a scalable and efficient mechanism to perform Anomaly Detection and Mitigation of DDoS attack.

Virtual software-defined networks can be easily developed and tested through Mininet. Mininet can be installed on any PC or laptop that enables to develop SDN. Mininet enables

- Building prototypes of Software-Defined networks rapidly
- Complex topologies can be developed and tested without wiring a physical network
- Multiple developers can work on the same topology without any dependency problems.

2. Related Work

Sergei Dotcenko*et. al.* [3] used the Mininet for implementing a rapid prototype of OpenFlow network. They built an OpenFlow network as an SDN network and connected their own IDS, which consists of statistic collection and processing module and decision-making module. They made it to check their IDS for any active security threat detection. They have combined their proposed algorithm with the decision making which are based on fuzzy rules.

Sumantra I *et. al.* [4] proposed and formulated an effective scheme that can detect and mitigate a DDoS attack in a software-defined network. They have used entropy as a measure to detect DDoS attacks. Their approach used the source IP and several attributes of TCP flags and calculates entropy from them. When the entropy becomes less than the defined threshold, their system finds the attacker source and implies the mitigation strategy. All these processes will happen in their controller present in the network.

One of the drawbacks presents is that they have not used an IDS (Intrusion Detection System) to detect a DDoS attack. If an IDS is used in the network, both the control plane and the data plane can share the work of detecting the attack. IDS can detect the attack while the SDN controller can mitigate the attack.

Ahuja *et. al.* [5] proposed an approach to detect and mitigate a DDoS attack from the OpenFlow statistics. They collect the traffic statistics from various switches. They calculate the packet rate and bandwidth from the statistics. When attack happens, these values shoot up high. The forwarding logic is modified in order to normalize the packet rate and bandwidth, which actually drops the packets from the malicious host instead of forwarding those packets. They collect the statistics and mitigate the DDoS attack, which might cause a delay in network, since the values may not be calculated real-time. Hence, an IDS can be used, where the detection rules can be defined and through that, the attack can be detected and then be mitigated by the SDN controller.

Deepa *et. al.* [6] proposed a hybrid machine learning model to safeguard the network from the DDoS attack. Their hybrid model provides more accuracy, less false alarm rate, and more detection rate when compared with simple machine learning models. They have used Support Vector Machine (SVM) and Self Organizing Maps (SOM) as their models. In their model, the attacks are identified, by passing the traffic through SOM module. To detect new kind of attacks, the traffic from SVM module is again passed through the SOM module. The particular connection is closed and the rules will be updated in the table, after the detection of attack.

Thomas *et. al.* [7] proposed a method that can detect the DDoS attack from the traffic monitoring method iftop in the server. They use iftop as a third-party application and it will check the traffic for a specific amount of time. iftop is a traffic monitoring application which can be used to find the bandwidth of the packets that are incoming, also with address. They have used SOM algorithm as the algorithm to detect the DDoS attack. By using the iftop, firewall actively detects and blocks the address, which is forwarded by the server from the analysis of network traffic. But using firewall and detecting cum mitigating the DDoS attack is an old school and slower way.

Chin *et. al.* [8] proposed a DDoS attack detection approach that can coordinate the monitors present in the SDN and the controllers are concentrated on an SDN Open Virtual Switch (OVS), which can inspect the packets selectively on demand. They have derived to use a synergistic monitoring, detection and mitigation strategy to detect TCP SYN flood attack.

This system can only detect the TCP SYN attack, which cannot be considered as an effective mechanism to detect various DDoS attack. But approach can be implied along with combination with other approaches to attack other types of DDoS attacks extensively.

Lim S *et. al.* [9] investigated on DDoS attack detection and mitigation on SDN. It was discussed to detect the malicious and benign traffic using POX controller by a packet filtering function. OpenFlow protocol was used in the network environment to effectively detect and mitigate DDoS attacks.

Karan B V *et. al.* [10] proposed a DDoS attack detection system with two levels of security. One layer was protected with Snort. The other layer was protected with SVM and DNN algorithms. They trained the models using KDD-99 dataset. A similar proposal was done by MisharaniMeti*et. al.*[11] where they use two algorithms namely, SVM and NN.

Hassan Z et. al. [12] proposed a DDoS attack detection system using SNORT alone. They have also discussed some existing techniques to overcome/mitigate a DDoS attack. A similar work was proposed by NenovaM *et. al.*[13] where they use the Snort IDS to detect DDoS attacks.

Rana A *et. al.* [14] proposed a DDoS attack detection system that can detect the presence of attack as well as identify the route of the attack. They are using an optimized SVM algorithm which was integrated with SNORT IPS. This helps the entire network to detect the DDoS attack and provide a mitigation strategy.

Akyazi U *et. al.* [15] proposed yet another DDoS attack detection mechanism using mobile agents. They tested four intrusion detection methods. MIT DARPA LLDOS 1.0 was the dataset used in the system.

3. Proposed Methodology

A. System Setup

Our proposed system is built with the help of Mininet. Mininet is mainly used to create virtual networks, where we can establish our desired network with the help of GUI and CLI provided by Mininet. Mininet GUI is used to design the topology of the network. Here, we can design the network with required number of hosts and required number of forwarding devices etc.

Since this software can help to design a software-defined network, adding a controller to the designed network is must. This controller is responsible to monitor and regulate the network traffic. This software opens up a way to virtualize and plan a real physical network. Here, we can emulate our own physical network and see the efficiency of our virtual network instead of building our network physically and get confused on how to build it at the first time.

OpenFlow is the protocol used in this system. This is actually a protocol used in SDN, where this protocol helps the controller to communicate with the data (forwarding) plane. This is a standard protocol used in any SDN network. Through OpenFlow, the SDN controller writes the changes happened to the forwarding plane. By this feature, partitioning traffic, flow controlling for ideal performance, and testing novel applications and configurations etc. can be done by the network administrator.

SNORT is the IDS used here. SNORT is the first and the foremost Intrusion Prevention System (IPS) in the world. Snort is open-source, which means anyone can contribute to the development of SNORT. Snort IPS detects malicious network traffic through a series of rules. SNORT uses such rules by which it finds packets that match against them. Through this approach, it generates alerts for the user. SNORT can be used in three ways: like tcpdump, it can be used as a packet sniffer, to debug network traffic, it can be used as a packet logger, or it can also be used as a complete Intrusion Detection and Prevention System(IDS/IPS). SNORT can be used personally and commercially as well.

SNORT rules are used to detect the DDoS attack in the network. When the SNORT matches the packets to detect the attack with the rules, it instructs the SDN controller to block the IP of the host that cause(s) the DDoS attack. SNORT rules can be defined, configured and modified at /opt/snort/snort.conf location.

In the system, a separate network for hosts is built and a separate network is also built for servers. Both the hosts and servers are bridged with a root switch. Only this root switch is responsible to carry the whole network traffic. SDN controller and the IDS will be communicating with the root switch.

In short, once the network connection is established in the system, the SDN controller establishes a socket communication between the root switch, followed by the packet mirroring from the root switch to the SNORT IDS. SNORT detects the DDoS attack by matching the packets against the rules, followed by sending the event to the SDN controller. SDN controller blocks the IP(s) after receiving the events from SNORT.

Unlike the old school way of using firewall, where the rules are added manually after an occurrence of DDoS attack into it, our system can detect the DDoS attack based on the events which is an agile and feasible way to detect the DDoS attack. Through this, network delay can also be minimized, it can reduce the overhead in the control plane in the SDN, by sharing the work with the data plane and this method is also cost-effective.

B. Algorithm

Firstly, we have to create a virtual network where we implement the DDoS attack. The network might need 4 hosts and 2 servers. A separate switch is connected between 4 hosts in order to communicate between the hosts and a separate switch is connected between 2 servers for effective communication between the servers. A root switch will be connected between the hosts and servers for the purpose of communication between the servers and hosts.

The RYU controller will be connected to the root switch. Also, an IDS will also be connected with the root switch. The controller monitors and controls the traffic in the root switch. IDS is used to detect any attack related packets and traffic in the network.

While the network is established, the root switch and the SDN controller will establish a socket communication between them in order to control the network traffic in the root switch.

After the socket communication, the SDN controller will instruct the root switch to mirror the packets to the IDS. SNORT is the IDS used in our project. Now the root switch will mirror the packet to the SNORT. Based on the detection rules, SNORT will analyse the packets and detect a possible DDoS attack. Once the SNORT detects the DDoS attack, the SNORT sends the event with the details of the attack. Now the controller receives the event. Based on the details provided by SNORT, the controller blocks the IP from where the DDoS attack initiated

The algorithm for the DDoS attack detection is mentioned aside.

Step 1: Initialize the network devices and the network.

Step 2: Establish a socket communication from the controller to IDS.

Step 3: Mirror the packet from the root switch to IDS.

Step 4: Detect the attack based on the rules defined in the IDS.

Step 5: If the rule is satisfied, send a relative event to the SDN controller.

Step 6: Let the controller receive the event and drop the packets coming from the malicious/attacking host(s).

Pseudocode: **SAY Algorithm**

```
1   BEGIN
2   start hosts(h1,h2,h3,h4);
3   start servers(s1,s2);
4   start controller(con1),snort(ids1);
5   switch_assign(sw2 -> h1,h2,h3,h4);
6   switch_assign(sw3 -> s1,s2);
7   switch_assign(sw1 -> sw2, sw3);
8   socket_initiate(con1,ids1,"unsock");
9   mirror_packet(sw1-eth2,ids1);
10  while(rule(anySourceIP,anyDestIP,timestamp,alert_id,threshold_type,packet_count,duration,payload_size)==true)
11          event=send_event((sourceIP,destIP,alert_id,message,buffer)->(ids1,con1));
12          receive_event(con1,sourceIP,destIP,alert_id,message,buffer);
13          if(packet_count>threshold && seconds<=1 && lower_thres<=payload_size<=upper_thres)
14          block_machine(sourceIP,ACTION="drop_packet");
15          goto 9;
16  end_snort(sw1-eth2,network,interface="INTR_UNREACHABLE")
17  END
```

4. SIMULATION AND ANALYSIS

For simulation of DDoS attack mininet is the framework used to build a virtual network. SNORT IDS detects the attack from the defined rules and RYU controller controls the virtual network and also takes care of blocking the hosts that cause the DDoS attack. The step by step analysis of the project can be seen below

A. Experimental Topology

A sample network is built with 4 hosts and 2 servers. OVS switches with OpenFlow protocol support are connected as per the topology. Finally a remote controller with address 0.0.0.0 and port 6633 is attached to control the sample network.

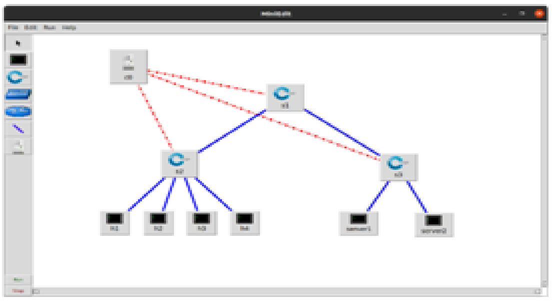


Fig 4.1 The topology of our network

In order to use the RYU controller to control the network and SNORT IDS to detect an attack, SNORT and RYU are installed from the installer menu. SNORT can be installed from the apt-get tool and ryu-manager tool must be installed from the Python PIP module since the RYU works on Python. After both the modules were installed, the ryu-manager is located using the locate command. Then the ryu-manager must be started from their own directory and hence, the directory path is changed to that directory. Ryu-manager tool is available in the python dist-packages directory.

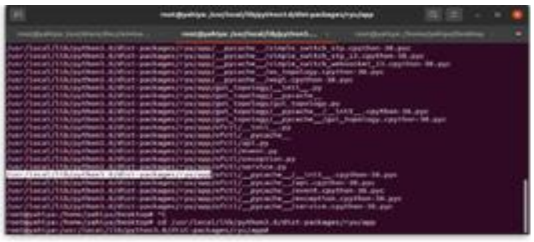
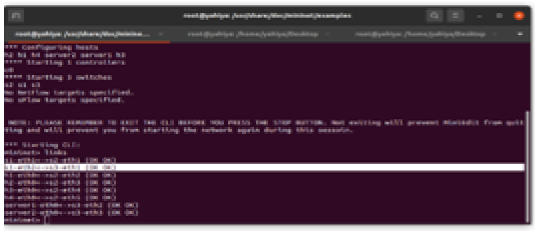


Fig 4.2 Location of RYU Manager

After the directory is changed, we have opened a new terminal window to open the Snort IDS. We stopped the Snort service, in order to start the Snort in the required interface. To know the exact interface, links command is used in the mininet CLI in order to know the links between the network devices. In order to mirror the packets from root switch, it was found out that “s1-eth2” is the right interface to start the Snort service on.



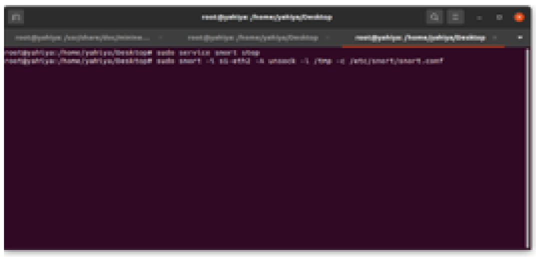
```
*** Configuring hosts
h2 h1 h4 server2 server1 h3
*** Starting 1 controllers
c0
*** Starting 3 switches
s2 s1 s3
No NetFlow targets specified.
No sFlow targets specified.

 NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE [illegible]

*** Starting CLI:
mininet> links
[illegible]
mininet>
```

Fig. 4.3 The links between the network devices

To start the Snort, the service “snort” is first stopped and then started again from the interface “s1-eth2” to mirror the packets from the root switch. Then the Snort will be started on that interface.



```
root@yahiya:/home/yahiya/Desktop# sudo service snort stop
root@yahiya:/home/yahiya/Desktop# sudo snort -i s1-eth2 -A unsock -l /tmp -c /etc/snort/snort.conf
```

Fig. 4.4 Command to start the Snort

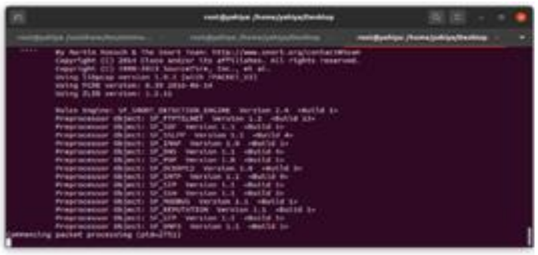
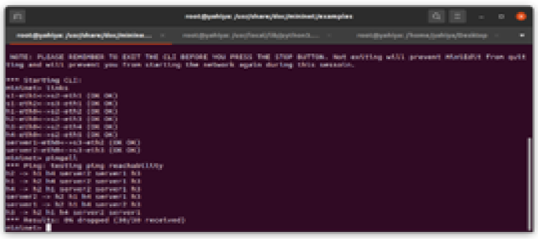


Fig. 4.5 The Snort IDS is working on the interface “s1-eth2”

The Snort will send the alerts through the UNIX socket logging. A UNIX socket communication must be established between the RYU controller and the Snort IDS. So in order to start a socket communication, the ryu package has a file called “simple_switch_snort.py” which will be used to start the communication.

Now the “simple_switch_snort.py” will be started using ryu-manager. After starting the RYU controller, each device in the virtual network must ping with the other devices. To check this, “pingall” command is typed to see the network links. If the packets are 100% transmitted and 0% dropped, then it indicates that the RYU controller properly controls the virtual network and is ready.



```
NOTE: PLEASE REMEMBER TO EXIT THE CLI BEFORE YOU PRESS THE STOP BUTTON. Not exiting will prevent MiniEdit from quitting and will prevent you from starting the network again during this session.

*** Starting CLI:
mininet> links
[illegible]
mininet> pingall
*** Ping: testing ping reachability
[illegible]
*** Results: 0% dropped (30/30 received)
mininet>
```

Fig. 4.6 “pingall” command

Now after starting the SNORT and RYU, now the ICMP flood attack is conducted to simulate the DDoS attack in the network. Appropriate command is typed and the DDoS attack is detected. If the Snort and RYU have been correctly configured, the alert message can be seen in the RYU window.

```
[illegible]
```

Fig. 4.7 Detection of ICMP flood from Snort

The packets and alerts details are manually extracted from the Snort statistics and further calculation. The analysis of attack detection can be seen below.

B) Analysis

True Positive alerts and false negative alerts are analysed from the packet data generated by Snort. If detecting ICMP flood attack through the packet transmission behaviour happens, then detecting a normal ping test could be considered as false positive. For example, if 1000 packets are detected malicious from the ICMP flood attack, detecting 1000 packets from a genuine ping transit between the devices is considered as a false positive alert.

Although if we create a rule and implement the rule for false positive alerts, the false positive alert itself becomes a false negative alert when both the rules for a normal ping and a ping flood attack gets satisfied when generating a ping flood attack. Hence it is evident that false negative alerts should be eliminated.

The Snort rules are satisfied and met based on the packet count within the provided no. of seconds. If the packet transit between the network devices exceeds the packet count within the given time interval, then the behaviour will be detected as an attack.

The detection might happen from an ICMP flood attack and a genuine ping request. Hence, there might be chance of more false negative alerts based on the no. of ping requests. Eliminating the false negative alerts is hypothetical but those can be reduced to a certain extent. In order to reduce the false negative alerts, the packet count and the duration can be raised to a certain amount to reduce false negative alerts. Here, the false negative alerts have been analysed using varying no. of packet count.

Firstly, the false negative alerts have been plotted by considering 10 packets per second as an attack(considering the ‘threshold’ type).

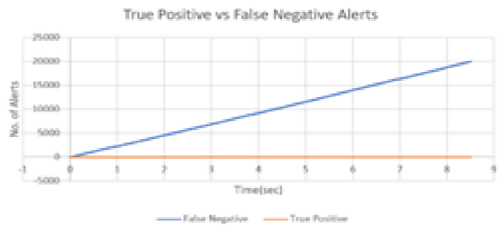


Fig. 4.8 True Positive vs False negative Alerts (For 20 packets/second)

Secondly, the false negative alerts have been plotted by considering 100 packets per second (considering the 'threshold' type). The false negative alerts have been reduced to 1/10th of the previous false negative alerts.

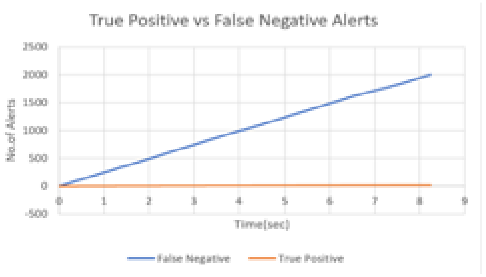


Fig. 4.9 True Positive vs False negative Alerts (For 100 packets/second)

Thirdly, the false negative alerts have been plotted by considering 1000 packets per second (considering the 'threshold' type). The false negative alerts have been reduced to 1/100th of the firstly generated false negative alerts.

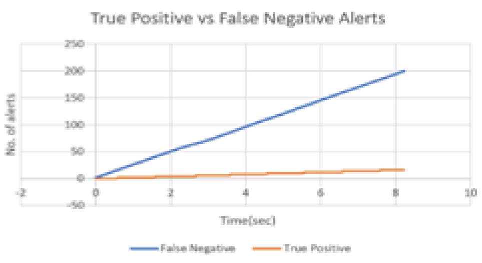


Fig. 4.10 True Positive vs False negative Alerts (For 1000 packets/second)

Finally, the false negative alerts have been plotted by considering 1000 packets for 100 seconds(considering the 'both' type). The false negative alerts have been reduced to 1/1000th of the firstly generated false negative alerts. It is because the rule that is satisfied when a normal ping occurs will wait for longer duration.

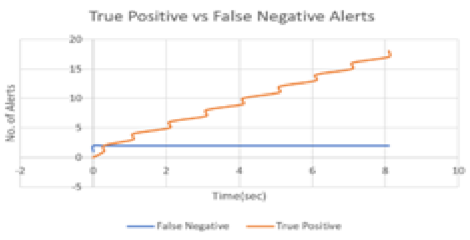
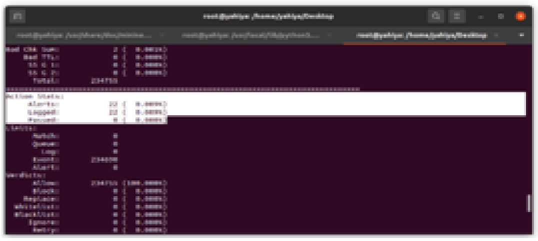


Fig. 4.11 True Positive vs False negative Alerts (For 1000 packets/second)

It can be noticed that the true positive alerts stays constant throughout the 4 different graph plots. This is because, an ICMP Flood Attack is detected at a default rate of 5000 packets per second, from the second rule. This is why the true positive rate stays constant throughout the analysis.



```
Bad Chk Sum:            2 (  0.001%)
    Bad TTL:            0 (  0.000%)
     S5 G 1:            0 (  0.000%)
     S5 G 2:            0 (  0.000%)
      Total:       234755
===============================================================================
Action Stats:
     Alerts:           22 (  0.009%)
     Logged:           22 (  0.009%)
     Passed:            0 (  0.000%)
Limits:
      Match:            0
      Queue:            0
        Log:            0
      Event:       234698
      Alert:            0
Verdicts:
      Allow:       234755 (100.000%)
      Block:            0 (  0.000%)
    Replace:            0 (  0.000%)
  Whitelist:            0 (  0.000%)
  Blacklist:            0 (  0.000%)
     Ignore:            0 (  0.000%)
      Retry:            0 (  0.000%)
```

Fig. 4.12 No. of alerts detected by Snort out of an ICMP Flood attack (at an interval of 10 seconds.)

Although the rule was set for detecting a normal ping, It was concluded that normal ping does not be needed to detect since raising alerts for normal ping is futile and might become a bottleneck because the normal ping rule gets satisfied once (atleast) even when the ping flood attack happens. Hence we removed the rule. After that, we find nearly 0% of false positive and false negative alerts alone.

Moreover, we have also covered the Fraggle attack. We implemented the SNORT rules to detect a UDP packet too. Detailed analysis for the Ping flood (Smurf) attack and Fraggle (UDP Flood) attack can be found below.

If our system detects the attack only based on count, the attacker can still bypass such rule by inflating the packet size larger and reducing the packets per second. By this way, the attacker can conduct the DDoS attack. To mitigate such attacks yet again, the detailed analysis of the mitigation strategy can be read below.

C. ICMP Ping Flood Attack / Smurf Attack Detection and Mitigation

This is an attack where ICMP packets are used to flood the victim server through consuming the network bandwidth of the victim server. The objective of our system is to detect and mitigate the malicious packets going through the network.

There is a chance of bypassing the IDS, if the IDS is detecting the attack only on packets per second filter. An attacker might attack with huge data packets with lower transmission rate. This leads to crashing the server.

Therefore, we have wrote a series of rules that can detect with the packet count as well as the packet payload size. Once the packet payload size was measured by the IDS, the IDS can detect the occurrence of bandwidth overflow and hence dropping the packets coming from the malicious hosts.

Fig. 4.13 Series of ICMP rules

There is a chance for attacker to slowdown the server rather than crashing the server. Because, if the attack was happening within the bandwidth, none of the rules might get satisfied and the IDS tends to think it as a normal traffic.

In order to overcome that issue, we declared a default rule that gets satisfied everytime when the packet payload size crosses the server's MTU size.

Fig. 4. 14 The default ICMP rule to filter the packets that are bigger than the server's MTU size.

An MTU stands for Maximum Transmission Unit. This is the size determined by every server in order to restrict the huge packets from entering into the server.

D. Fraggle / UDP Flood Attack Detection and Mitigation

This is an attack where the UDP packets are used to flood the victim server through consuming the network bandwidth of the victim server. Such situation must also been detected and mitigated.

The attacker can still use the above mentioned techniques to bypass the IDS rules. Hence, we have also wrote a series of rules that can detect with the packet count as well as the packet payload size.

Fig. 4.15 Series of UDP rules

The default rule to detect the MTU size overflow is also defined for UDP packets too.

Fig. 4.16 The default UDP rule to filter the packets that are bigger than the server's MTU size.

The MTU size for the Ping Flood Attack and the UDP Flood Attack is determined to 1460 bytes. For the experiment purpose, the server's bandwidth is set to 10 Mbps.

Fig. 4.17 Detection of UDP flood from Snort

5. Conclusion And Future Work

Thus, this paper presents about the detection and mitigation of DDoS attack using Software Defined Network and an IDS. We have analysed and discussed about detecting the attack and mitigating the DDoS attack by measuring the traffic anomalies and packet payload size. In this study, we have considered two server nodes out of which one server was targeted and attacked. It is also concluded that the system can work for detecting DDoS attack on multiple hosts and on multiple servers. However, the attacker can still exploit the system and bypass the security measure by the fragmentation attack. We need to know the fragmentation offset in order to write a series of rules based on the offset(size). In future, we will implement even advanced techniques in order to know about the fragmentation offset, thereby enhancing the features of our system to detect the DDoS attacks even better and quicker

References

6. Narmeen Zakaria Bawany; Jawwad A. Shamsi; Khaled Salah: "DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions," Arab J Sci Eng 42:425–441, 2017.
7. Giotis, K.; Argyropoulos, C.; Androulidakis, G.; Kalogeras, D.; Maglaris,V.: "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," Comput. Netw. 62, 122–136, 2014.
8. Dotcenko, S.; Vladyko, A.; Letenko, I.: "A fuzzy logic-based information security management for software-defined networks," in 16th International Conference on Advanced Communication Technology (ICACT), Pyeongchang, Korea (South), 16-19 Feb. 2014

I. Sumantra, Dr. S. Indira Gandhi: "DDoS attack Detection and Mitigation in Software Defined Networks," in 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, 3-4 July 2020.

9. Nisha Ahuja, Gaurav Singal: "DDOS Attack Detection &amp; Prevention in SDN using OpenFlow Statistics," in2019 IEEE 9th International Conference on Advanced Computing (IACC), Tiruchirappalli, India, 13-14 Dec. 2019.
10. V.Deepa, K.MuthamilSudar, P.Deepalakshmi; "Detection of DDoS Attack on SDN Control plane using Hybrid Machine Learning Techniques," in 2018 International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 13-14 Dec. 2018.
11. Roshni Mary Thomas, Divya James: "DDOS Detection and Denial using Third Party Application in SDN," in 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), Chennai, India, 1-2 Aug. 2017
12. Tommy Chin Jr., Xenia Mountrouidou, Xiangyang Li, KaiqiXiong: "Selective Packet Inspection to Detect DoS Flooding Using Software Defined Networking (SDN)," in 2015 IEEE 35th International Conference on Distributed Computing Systems Workshops, Columbus, OH, USA, 29 June-2 July 2015.
13. S. Lim; J. Ha; H. Kim; Y. Kim; S. Yang: "A SDN-oriented DDoS blocking scheme for botnet-based attacks," 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), Shanghai, China, 8-11 July 2014.

14. Karan B. V.; Narayan D. G.2; P. S. Hiremath; “Detection of DDoS Attacks in Software Defined Networks,” in 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), Bengaluru, India, 20-22 Dec. 2018.
15. NisharaniMeti; Narayan D G; V. P. Baligar: “Detection of Distributed Denial of Service Attacks using Machine Learning Algorithms in Software Defined Networks,” in 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13-16 Sept. 2017.
16. Zohaib Hassan; Shahzeb; Roman Odarchenko; SergiyGnatyuk; Abnash Zaman; Masroor Shah: “Detection of Distributed Denial of Service Attacks Using Snort Rules in Cloud Computing &Remote Control Systems,” in 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), Kiev, Ukraine, 16-18 Oct. 2018.
17. Maria Nenova; Denis Atanasov; KirilKassev; Andon Nenov: “Intrusion Detection System Model Implementation against DDOS attacks,” in 2019 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS), Tel-Aviv, Israel, 16 January 2020.
18. Rana Abubakar; AbdulazizAldegheishem; Muhammad Faran Majeed; Amjad Mehmood; Hafsa Maryam; Nabil Alrajeh; Maple Carsten; Muhammad Jawad: “An Effective Mechanism to Mitigate Real-time DDoS Attack Using Dataset,” IEEE Access, Vol. no.: 8, pg.:126215 – 126227, 20 May 2020.
19. UgurAkyazi; A. SimaEtanerUyar: “Distributed Intrusion Detection using Mobile Agents against DDoS Attacks” in 2008 23rd International Symposium on Computer and Information Sciences, Istanbul, Turkey, 27-29 Oct. 2008.