2022 International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER) | 978-1-6654-8716-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/DISCOVER55800.2022.9974724

Detection and Mitigation of Low and Slow DDoS attack in an SDN environment

A N H Dhatreesh Sai
Dept. of CSE
Ramaiah Institute of Technology
Bengaluru, India
1ms18cs002@gmail.com

B H Tilak
Dept. of CSE
Ramaiah Institute of Technology
Bengaluru, India
1ms18cs034@gmail.com

N Sai Sanjith
Dept. of CSE
Ramaiah Institute of Technology
Bengaluru, India
1ms18cs079@gmail.com

Padi Suhas
Dept. of CSE
Ramaiah Institute of Technology
Bengaluru, India
1ms18cs087@gmail.com

Dr. Sanjeetha R
Dept. of CSE
Ramaiah Institute of Technology
Bengaluru, India
sanjeetha.r@msrit.edu

***Abstract*—Distributed Denial of Service (DDoS) attacks aim to make a server unresponsive by flooding the target server with a large volume of packets (Volume based DDoS attacks), by keeping connections open for a long time and exhausting the resources (Low and Slow DDoS attacks) or by targeting protocols (Protocol based attacks). Volume based DDoS attacks that flood the target server with a large number of packets are easier to detect because of the abnormality in packet flow. Low and Slow DDoS attacks, however, make the server unavailable by keeping connections open for a long time, but send traffic similar to genuine traffic, making detection of such attacks difficult. This paper proposes a solution to detect and mitigate one such Low and slow DDoS attack, Slowloris in an SDN (Software Defined Networking) environment.The proposed solution involves communication between the detection and mitigation module and the controller of the Software Defined Network to get data to detect and mitigate low and slow DDoS attack.**

***Index Terms*—Software Defined Networking (SDN), Low and Slow Distributed Denial of Service (DDoS) attacks, Slowloris.**

I. Acknowledgement

The authors would like to thank the Ramaiah Institute of Technology alumni association for funding this project.

II. Introduction

A. Software Defined Networking (SDN)

Software defined networking is a technology in networking that aims to make a network more programmable and easier to manage. SDN divides the network into the control plane and the data plane. The control plane consists of the controller, the main element of the network. The data plane is responsible for the forwarding of data.

All elements of the network are connected to the controller. The controller basically controls the entire network, telling switches and routers in the network what to do. Communication between the controller and the connected devices is done through OpenFlow protocol. Modules and flow rules are programmed into the controller. The switches act as the data plane, forwarding the data.

In traditional networks, the control plane and the data plane are closely related. However, in SDN, the control plane is abstracted from the data plane, making the network centralised.

SDN bought in many features that were absent in traditional networks, but these networks are still vulnerable to Distributed Denial of Service (DDoS) attacks. By making the control plane abstract from the data plane, other features in traditional networks, such as firewalls are also absent.

B. Distributed Denial of Service (DDoS) attacks

Volume based DDoS attacks (such as ping floods, UDP floods, etc.) send large amounts of packets to the victim service, rendering it unresponsive. Low and Slow DDoS attacks (such as Slowloris, Slow post, etc.) will send traffic that is similar to genuine traffic, but keep the connections open for a long time, thereby exhausting the resources.

C. Slowloris

Slowloris is a type of Low and Slow DDoS attack that aims to make the server unresponsive by keeping the connections open as long as possible. It is an application layer DDoS attack. Slowloris opens connections to the server. It then starts sending partial HTTP request packets at regular intervals. The server expects the request to be complete and hence keeps the connection open for a longer time. Eventually, the server can't keep any more concurrent connections open, thereby rejecting genuine connection requests.

This paper proposes a solution to Detect Slowloris attack hosts and mitigate further attacks from them in an SDN environment.

III. Literature Survey

In [1], the authors talk about method for botnets low rate a DDoS attack based on the behavior of the botnet. Because of the highly intentional cyberattacks it has led to many great developments in this particular area.

In [2], the authors talk about how slow DDoS attacks can be detected and different preventative methods and strategies have also been discussed in this paper and an innovative attach detection model is suggested and many different procedures and many processes models have also been discussed. The impact of trace back preparation is also discussed. Many researchers have been working and developed various techniques to prevent and protect various attacks on network level.

It is very big challenge and concern to identify which is a DDoS attack traffic and different types and which is a legitimate attack. To detect this type of attack the receiver has to analyze high voluminous traffic in real time and this also becomes a serious concern in the current reign of a very high-volume high velocity internet world. Even though many other techniques have been introduced it still becomes a challenge to detect and prevent these attacks from happening. In [3], authors give a detection method for such attacks based on Self similarity and the Hurst parameter

DDoS attacks severely impact Internet based services like e-commerce or net banking or transportation or any online services. Many hackers intentionally try to attack and try to comprise sensitive information and hence are highly vulnerable and we from the previous years we can also see that there has been an increase in these kinds of attacks. In spite of many solutions that were proposed in these years it's still very challenging to defend an DDoS attack. In [4], authors discuss and compare the tools and generators used by attackers and talks about on improving and having better defense mechanisms.

In [5], analysis was done on different kinds of slow DDoS attacks which targeted the HTTP layers. During each of these attacks, various network parameters such as window size and delta time of packets are varying continuously. The values of these parameters can be collected at the network gateway. The future scope of paper aims to identify various threshold values for the network parameters to detect and mitigate network attacks.

The methodology proposed by authors in [6] is based on identifying a threshold value for the number of connections being kept open for each IP Address and by monitoring the duration of time for which the connections are kept open. The results have shown that this method is significant to provide resistance against Slow DDoS attacks. This method can also prove to be significant against other network attacks when the right threshold values are identified.

Analysis of Slow Read DoS attack in a virtual environment was done by the authors int [7] to identify counter-effective measures. They have put forward the use of some of server security settings like ModSecurity to mitigate DDoS attacks. The authors have also designed an effective Slow Read DDoS attack to simulate and obtain results in with much better accuracy.

The authors of [8] have carried out an empirical study to analyse numerous vulnerabilities of a network to slow HTTP DoS attacks. They have also proposed a methodology for detecting DoS attacks using anomaly detection by calculating the Hellinger distance between two probability distributions generated during training and testing phases. The solution focusses on generating another distribution model during the slow HTTP attack which has a large proportion of incomplete messages and compares the model with already generated probability distribution model of the network.

Network security is always a concern, with new threats and vulnerabilities being reported on a regular basis. Denial of Service (DoS) assaults at the application layer are becoming increasingly widespread, creating network stress and service outages. Detection and mitigation strategies for such assaults have been a key priority for a long time, but new complex attack permutations are constantly being developed, leaving old protective techniques useless. The focus of [9] is on the detection of Slow HTTP POST DoS attacks. They conduct a variety of Slow HTTP POST attack configurations in a live network environment, with varied degrees of severity, to simulate a real-world attack situation. This research discusses the use of eight machine learning models to aid in the detection of Slow HTTP POST attacks. We did four 5-fold cross validation runs to generate their AUC performance statistic. Our findings showed that all eight learners correctly identified our attack traffic, demonstrating that Netflow characteristics are a good feature set for detecting Slow HTTP POST DoS attacks. In the future, we'll evaluate traffic patterns across a variety of HTTP DoS types and put different machine learning algorithms to the test.

Network programs that take a long time to perform are frequently targeted. Because the attackers follow the specification, it is difficult to detect them. Many network application servers are ill-equipped to deal with such a situation. attacks, either because to a lack of countermeasures or because of default settings. Settings are unconcerned about such assaults. The onus of proofing network services against such attacks is increasingly falling on the service provider. Slow HTTP attacks are different from flooding attacks, because detecting and mitigating them could take a lot of network infrastructure maintenance effort. In [10], the authors created a variety of concepts based on light-weight flow-based network traffic analysis that can assist in identifying intruders and assisting them in exiting the network. The research found that a network-based defence against delayed attacks is feasible and should be included in a network provider's defensive plan. The accuracy of the schemes isn't high enough to keep the system operating all the time, but it's valuable as part of a reactive protection system if ongoing attacks are found.

DDoS attacks (distributed denial of service) are a constant threat to Internet services. This year, the world record for the largest DDoS attack was set at 1.7 Tbps. On the other hand, detection and mitigation strategies are still insufficient. In order to mitigate attacks, many mitigation techniques require the victim's cooperation or the victim's administrator to become active. In [11], the authors built a system that can detect attacks, identify attackers, and neutralize them within the network architecture. Because of the enhanced flexibility of software-defined networks, new approaches to mitigate such attacks can be adopted.

[12] gives us an overview of Bohatei, which uses SDN and NFV (Network Function Virtualization) to get rid of the scaling limitations of normal DDoS defenses. The results show that the proposed framework is scalable, capable of handling up to 500 Gbps attacks. It is also responsive, handling the mitigation of DDoS attacks in under a minute.

[13] gives us a SDN framework called FlowTrApp, which can detect traffic ranging from low to high rate, long lived and short-lived attacks. It is done by analyzing the flow rate and flow duration of a particular flow. The authors have used sFlow-RT, a sFlow based flow analytics engine.

[14] provides an overview of the ATLANTIC framework in SDN environment, which is used to categorize traffic anomalies. These details can be used to detect or mitigate any attacks that can harm the working of the environment

In [15]Muraleedharan et. Al have proposed a deep-learning based approach to mitigate slow DoS attacks. Flow data is analysed to make a model which can be used to implement a preventive system for the attack. The Intrusion Detection Evaluation Dataset available in the Canadian Institute of Cybersecurity (CICIDS2017) is used for the model training and evaluation. The results show that the model proposed have achieved 99.1 percent overall accuracy.

[16] tells us about what a Slow DDoS attack is, giving a brief about slow DDoS attacks, various existing defensive methods to prevent slow DDoS attacks and conclusions on the theoretical study made on the defensive mechanisms.

IV. Proposed Methodology

The proposed methodology involves communication between the Detection and Mitigation application and the controller of Software Defined Network for getting any data or for mitigation purposes. The proposed methodology contains the following modules:

A. Environment setup

Mininet is used to set up the SDN environment. Ryu controller acts as the control plane. A network of hosts, switches and controller is created.

B. Server setup

A php application which displays Hello World is created. Apache server is used to host the php application. The server is run on a host in the network.

C. Sending attack traffic

Two hosts are used to send attack traffic to the victim server. Slowhttptest library is used to send the attack traffic. Slowloris method of low and slow DDoS attack Is used.

D. Detection and Mitigation

Detecting attack hosts is done by using the flow data and by analysing the packets sent during the attack through Wireshark. The average of the number of connections is taken through APIs and is set as a threshold. If the number of connections of a particular host crosses this threshold, we put that host under suspected attack host. Further, by taking the count of the number of partial request packets sent, if the count passes a threshold, we classify the host as attack host.

After detecting the attack hosts, we drop the packets coming from the attack host to the network through REST APIs and mitigate the attack.

Algorithm 1 shows the detection and mitigation algorithm for the proposed solution.

Algorithm 1 Detection and Mitigation algorithm

```
Require: $totalPacketCount, numberOfHosts[]$
Require: $wiresharkAttackCapture[][], allHosts[]$
Require: $numberOfConnectionsPerHost[]$
0: $expectedAttackHosts \leftarrow []$
0: $attackHosts \leftarrow []$
0: $averagePacketCount \leftarrow \frac{totalPacketCount}{numberOfHosts}$
0: for $host \leftarrow allHosts$ do
0:   if $numberOfConnectionsPerHost[host] > averagePacketCount$ then
0:     $expectedAttackHosts \leftarrow host$
0:   end if
0: end for
0: for $hostexpectedAttackHosts$ do
0:   $i \leftarrow$
0:   $differenceof$
0:   $wiresharkAttackCapture[host][PSHPackets]$
0:   $wiresharkAttackCapture[host][FINPackets]$
0:   if $i > 100$ then
0:     $attackHosts \leftarrow host$
0:   end if
0: end for
0: $Mitigation$
0: for $host \leftarrow attackHosts$ do
0:   $Droppacketsfromhost$
0: end for=0
```

V. Implementation

A. System Configuration

SDN topology is setup on Mininet with RYU as the controller for testing. The system uses Ubuntu OS with Intel i5 processor and 16 GB of RAM. slowHTTPtest library is used to start the attack. Wireshark is used to capture the traffic.

B. Setting up SDN environment

Mininet is used to setup the SDN environment with RYU as the controller. DDoS simulation, detection and mitigation are all done in this environment. The TreeNet topology in Mininet is used to setup the network. The fanout (number of children per node) and depth (number of levels) parameters of this topology is set to 2 and 3. The network consists of 8 hosts and 7 switches. The hosts are labelled from H1 to H8 and the IP address range is from 10.0.0.1 (H1) to 10.0.0.8 (H8). The switches are labelled from S1 to S8. A single RYU controller is connected to the root switch (S1). In this experiment, H5 (10.0.0.5) hosts the server containing the php application. H2 and H3 are the DDoS attack hosts. RYU's Northbound APIs is used to get data from the controller and to add flow rules. Figure 1 shows the Mininet topology setup.

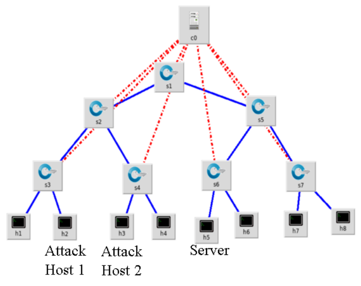


Fig. 1. TreeNet Topology

As shown in figure 1, H5 (10.0.0.5) acts as the server with the php application running on it (on port 3000) and H2 (10.0.0.2) and H3 (10.0.0.3) act as the attack hosts.

C. Attacking the server using Slowloris

After setting up the network and running the application on the server, attack is initiated on the server. slowHTTPtest library is used to attack the server. Attack command is sent from the terminals of the attack hosts (H2 and H3). The attack is run for 60 seconds to gather data required for analysis.

D. Detection and Mitigation

Using the Northbound REST APIs in RYU, the number of connections per host and the total number unique hosts are retrieved. Average number of connections (total number of connections / total number of hosts) is calculated. If the number of connections of a host exceeds the average number of connections calculated, we place the host under suspected attack hosts list. Wireshark is used to capture the entire flow of traffic. The data captured from Wireshark is analyzed and patterns are observed. The number of partial request packets sent from each host in the suspected attack hosts list is calculated from Wireshark. While establishing a connection, there is one PSH packet sent for a request. The difference between PSH packets and FIN packets for each host gives the number of partial request packets. If the number of such requests sent is greater than the threshold, the host is classified as attack host and is placed under the attack hosts list.

To mitigate the attack, for each host in the attack hosts list, flow rules are modified using the Northbound RYU APIs instructing the switches to drop any traffic coming from these attack hosts.

The flow diagram of the proposed solution is shown in figure 2.

VI. Results and Analysis

The attack was successfully executed and the detection and mitigation was also done. It is observed that a minimum of 700 consecutive connections are required to DDoS the server and the server is unavailable after a few seconds of the attack.

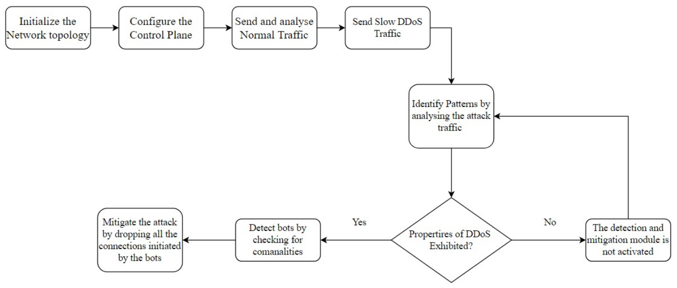

Initialize the Network topology
Configure the Control Plane
Send and analyse Normal Traffic
Send Slow DDoS Traffic
Identify Patterns by analysing the attack traffic
Propertires of DDoS Exhibited?
Yes
No
Detect bots by checking for comanalities
Mitigate the attack by dropping all the connections initiated by the bots
The detection and mitigation module is not activated

Fig. 2. Flow diagram

The graph for the entire duration of the attack is generated by the slowHTTPtest library (figure 3). It is seen that roughly after 700 connections, the server goes down and remains unavailable for the rest of the attack duration.

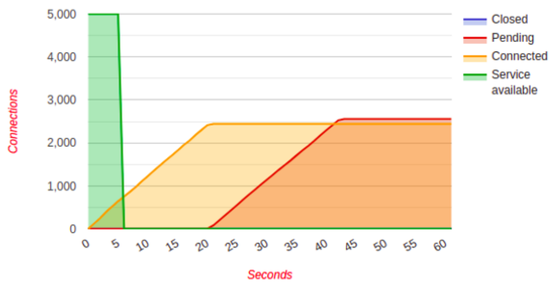


Fig. 3. Attack graph

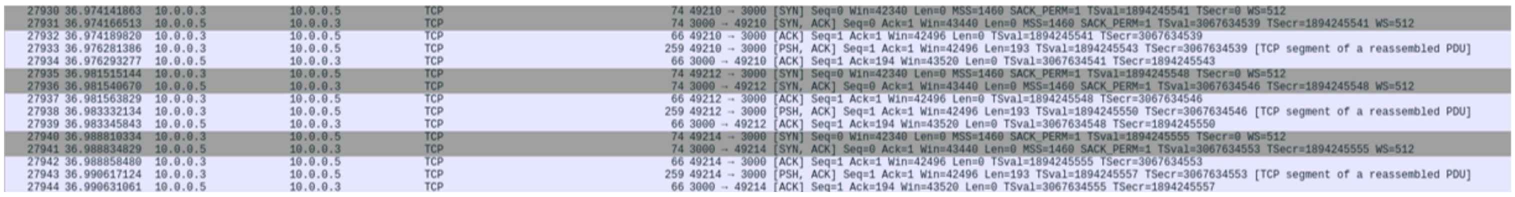
Wireshark analysis shows that the attacking hosts exhibit a pattern while establishing connections. After the usual three way TCP handshake and data retrieval request, there are no FIN packets sent (figure 4). FIN packets tell the server no further data is required and to close the connection.

The detection and mitigation module is run as a separate application and communicates with the controller through Northbound APIs. It is observed that a minimum of 100 partial request packets are sent after the connection has been established in order to keep the connection alive. These partial request packets have fake HTTP header content sent. Genuine traffic have their TCP buffer filled before a request can be sent. However, the partial request packets' buffer is not completely filled (shown in figure 5). These packets are sent with PSH (push) flag set to 1 so that the request is pushed even though the TCP buffer is not full.

These partial request packets contain fake HTTP header content which can be seen in figure 6.

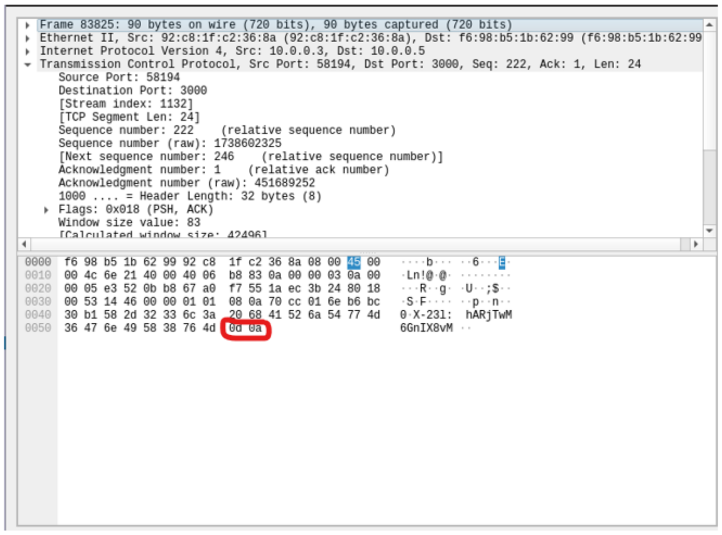
After the attack is complete, the detection mitigation module is run. The Wireshark capture file for the whole duration of the attack is loaded into the module. Figure 7 shows the output of the detection mitigation module after detecting the attack hosts and mitigating the attack.

The detection and mitigation module outputs the total number of PSH packets and the total number of FIN packets for each host. The attack hosts are also output by the module and flow rules are added for each attack host. The output



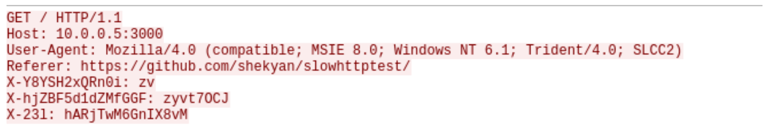
```
27930 36.974141863  10.0.0.3        10.0.0.5        TCP     74 49210 → 3000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=1894245541 TSecr=0 WS=512
27931 36.974166513  10.0.0.5        10.0.0.3        TCP     74 3000 → 49210 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=1460 SACK_PERM=1 TSval=3067634539 TSecr=1894245541 WS=512
27932 36.974189820  10.0.0.3        10.0.0.5        TCP     66 49210 → 3000 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=1894245541 TSecr=3067634539
27933 36.976281386  10.0.0.3        10.0.0.5        TCP    259 49210 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=42496 Len=193 TSval=1894245543 TSecr=3067634539 [TCP segment of a reassembled PDU]
27934 36.976293277  10.0.0.5        10.0.0.3        TCP     66 3000 → 49210 [ACK] Seq=1 Ack=194 Win=43520 Len=0 TSval=3067634541 TSecr=1894245543
27935 36.981515144  10.0.0.3        10.0.0.5        TCP     74 49212 → 3000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=1894245548 TSecr=0 WS=512
27936 36.981540670  10.0.0.5        10.0.0.3        TCP     74 3000 → 49212 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=1460 SACK_PERM=1 TSval=3067634546 TSecr=1894245548 WS=512
27937 36.981563829  10.0.0.3        10.0.0.5        TCP     66 49212 → 3000 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=1894245548 TSecr=3067634546
27938 36.983332134  10.0.0.3        10.0.0.5        TCP    259 49212 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=42496 Len=193 TSval=1894245550 TSecr=3067634546 [TCP segment of a reassembled PDU]
27939 36.983345843  10.0.0.5        10.0.0.3        TCP     66 3000 → 49212 [ACK] Seq=1 Ack=194 Win=43520 Len=0 TSval=3067634548 TSecr=1894245550
27940 36.988810334  10.0.0.3        10.0.0.5        TCP     74 49214 → 3000 [SYN] Seq=0 Win=42340 Len=0 MSS=1460 SACK_PERM=1 TSval=1894245555 TSecr=0 WS=512
27941 36.988834829  10.0.0.5        10.0.0.3        TCP     74 3000 → 49214 [SYN, ACK] Seq=0 Ack=1 Win=43440 Len=0 MSS=1460 SACK_PERM=1 TSval=3067634553 TSecr=1894245555 WS=512
27942 36.988858480  10.0.0.3        10.0.0.5        TCP     66 49214 → 3000 [ACK] Seq=1 Ack=1 Win=42496 Len=0 TSval=1894245555 TSecr=3067634553
27943 36.990617124  10.0.0.3        10.0.0.5        TCP    259 49214 → 3000 [PSH, ACK] Seq=1 Ack=1 Win=42496 Len=193 TSval=1894245557 TSecr=3067634553 [TCP segment of a reassembled PDU]
27944 36.990631061  10.0.0.5        10.0.0.3        TCP     66 3000 → 49214 [ACK] Seq=1 Ack=194 Win=43520 Len=0 TSval=3067634555 TSecr=1894245557
```

Fig. 4. Pattern exhibited



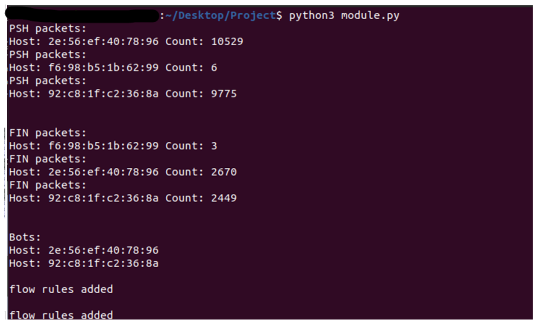
```
Frame 83825: 90 bytes on wire (720 bits), 90 bytes captured (720 bits)
Ethernet II, Src: 92:c8:1f:c2:36:8a (92:c8:1f:c2:36:8a), Dst: f6:98:b5:1b:62:99 (f6:98:b5:1b:62:99
Internet Protocol Version 4, Src: 10.0.0.3, Dst: 10.0.0.5
Transmission Control Protocol, Src Port: 58194, Dst Port: 3000, Seq: 222, Ack: 1, Len: 24
   Source Port: 58194
   Destination Port: 3000
   [Stream index: 1132]
   [TCP Segment Len: 24]
   Sequence number: 222    (relative sequence number)
   Sequence number (raw): 1738602325
   [Next sequence number: 246    (relative sequence number)]
   Acknowledgment number: 1    (relative ack number)
   Acknowledgment number (raw): 451689252
   1000 .... = Header Length: 32 bytes (8)
 Flags: 0x018 (PSH, ACK)
   Window size value: 83
   [Calculated window size: 42496]

0000  f6 98 b5 1b 62 99 92 c8  1f c2 36 8a 08 00 45 00   ····b··· ··6···E·
0010  00 4c 6e 21 40 00 40 06  b8 83 0a 00 00 03 0a 00   ·Ln!@·@· ········
0020  00 05 e3 52 0b b8 67 a0  f7 55 1a ec 3b 24 80 18   ···R··g· ·U··;$··
0030  00 53 14 46 00 00 01 01  08 0a 70 cc 01 6e b6 bc   ·S·F···· ··p··n··
0040  30 b1 58 2d 32 33 6c 3a  20 68 41 52 6a 54 77 4d   0·X-231:  hARjTwM
0050  36 47 6e 49 58 38 76 4d  0d 0a                     6GnIX8vM ··
```

Fig. 5. Partial request packet



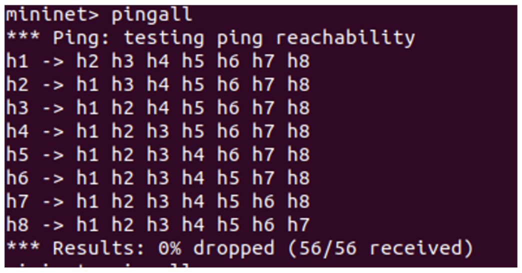
```
GET / HTTP/1.1
Host: 10.0.0.5:3000
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2)
Referer: https://github.com/shekyan/slowhttptest/
X-Y8YSH2xQRn0i: zv
X-hjZBF5d1dZMfGGF: zyvt7OCJ
X-231: hARjTwM6GnIX8vM
```

Fig. 6. Fake header content



```
[redacted]:~/Desktop/Project$ python3 module.py
PSH packets:
Host: 2e:56:ef:40:78:96 Count: 10529
PSH packets:
Host: f6:98:b5:1b:62:99 Count: 6
PSH packets:
Host: 92:c8:1f:c2:36:8a Count: 9775


FIN packets:
Host: f6:98:b5:1b:62:99 Count: 3
FIN packets:
Host: 2e:56:ef:40:78:96 Count: 2670
FIN packets:
Host: 92:c8:1f:c2:36:8a Count: 2449


Bots:
Host: 2e:56:ef:40:78:96
Host: 92:c8:1f:c2:36:8a

flow rules added

flow rules added
```

Fig. 7. Output from Detection and Mitigation module



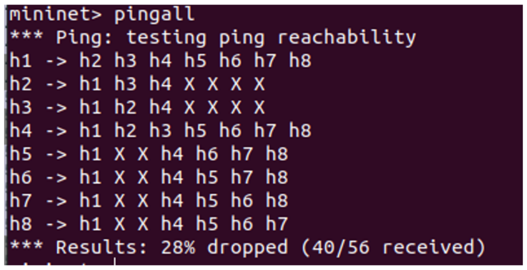
```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 h5 h6 h7 h8
h3 -> h1 h2 h4 h5 h6 h7 h8
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 h2 h3 h4 h6 h7 h8
h6 -> h1 h2 h3 h4 h5 h7 h8
h7 -> h1 h2 h3 h4 h5 h6 h8
h8 -> h1 h2 h3 h4 h5 h6 h7
*** Results: 0% dropped (56/56 received)
```

Fig. 8. Pinging all hosts before mitigation

of the pingall command in Mininet (where each host pings every other host) before and after the detection and mitigation module is started is shown in figures 8 and 9 respectively. As shown in figure 9, H2 and H3 are not able to ping hosts after the mitigation module.

VII. Conclusion

In this paper, an overview of the experiment carried out to detect and mitigate a low and slow DDoS attack was given. The attack, Slowloris is simulated in and SDN environment with RYU as the controller. A module is created to detect and mitigate the attack. The module uses



```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4 h5 h6 h7 h8
h2 -> h1 h3 h4 X X X X
h3 -> h1 h2 h4 X X X X
h4 -> h1 h2 h3 h5 h6 h7 h8
h5 -> h1 X X h4 h6 h7 h8
h6 -> h1 X X h4 h5 h7 h8
h7 -> h1 X X h4 h5 h6 h8
h8 -> h1 X X h4 h5 h6 h7
*** Results: 28% dropped (40/56 received)
```

Fig. 9. Pinging all hosts after mitigation

Wireshark for analysis and RYU's Northbound APIs to get the flow data and modify flow rules.The proposed solution is only for the low and slow DDoS attack,namely slowloris. Future scope can include detection and mitigation modules for other types of low and slow DDoS attacks such as slow READ,slow POST and range header attacks.

REFERENCES

[1] Lysenko, Sergii, Kira Bobrovnikova, Serhii Matiukh, Ivan Hurman, and Oleh Savenko. "Detection of the botnets' low-rate DDoS attacks based on self-similarity." International Journal of Electrical Computer Engineering (2088-8708) 10, no. 4 (2020).

[2] Cusack, Brian, Raymond Lutui, and Reza Khaleghparast. "Detecting slow ddos attacks on mobile devices." In The 27th Australasian Conference on Information Systems. Australasian Conference on Information Systems (ACIS), 2016.

[3] Deka, Rup K., and Dhruba K. Bhattacharyya. "Self-similarity based DDoS attack detection using Hurst parameter." Security and Communication networks 9, no. 17 (2016): 4468-4481.

[4] Behal, Sunny, and Krishan Kumar. "Characterization and Comparison of DDoS Attack Tools and Traffic Generators: A Review." Int. J. Netw. Secur. 19, no. 3 (2017): 383-393.

[5] Muraleedharan, N., and B. Janet. "Behaviour analysis of HTTP based slow denial of service attack." In 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), pp. 1851-1856. IEEE, 2017.

[6] Hirakawa, Tetsuya, Kanayo Ogura, Bhed Bahadur Bista, and Toyoo Takata. "A defense method against distributed slow http dos attack." In 2016 19th international conference on network-based information systems (NBiS), pp. 152-158. IEEE, 2016.

[7] Park, Junhan, Keisuke Iwai, Hidema Tanaka, and Takakazu Kurokawa. "Analysis of slow read DoS attack and countermeasures on web servers." International Journal of Cyber-Security and Digital Forensics 4, no. 2 (2015): 339-353.

[8] Tripathi, Nikhil, Neminath Hubballi, and Yogendra Singh. "How secure are web servers? An empirical study of slow HTTP DoS attacks and detection." In 2016 11th International Conference on Availability, Reliability and Security (ARES), pp. 454-463. IEEE, 2016.

[9] Calvert, Chad, Clifford Kemp, Taghi Khoshgoftaar, and Maryam Najafabadi. "Detecting slow http post dos attacks using netflow features." In The thirty-second international FLAIRS conference. 2019.

[10] Lukaseder, Thomas, Lisa Maile, Benjamin Erb, and Frank Kargl. "SDN-assisted network-based mitigation of slow DDoS attacks." In International Conference on Security and Privacy in Communication Systems, pp. 102-121. Springer, Cham, 2018.

[11] Lukaseder, Thomas, Shreya Ghosh, and Frank Kargl. "Mitigation of flooding and slow DDoS attacks in a software-defined network." arXiv preprint arXiv:1808.05357 (2018).

[12] Fayaz, Seyed K., Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. "Bohatei: Flexible and Elastic DDoS Defense." In 24th USENIX security symposium (USENIX Security 15), pp. 817-832. 2015.

[13] C. Buragohain, and N. Medhi, "FlowTrApp: An SDN based Architecture for DDoS Attack Detection and Mitigation in Data Centers," in Proc. Int. Conf. SPIN, 2016, pp. 519-524.

[14] A. S. da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, "ATLANTIC: A Framework for Anomaly Traffic Detection, Classification, and Mitigation in SDN," in IEEE/IFIP NOMS, 2016, pp.27-35.

[15] Muraleedharan, N., and B. Janet. "A deep learning based HTTP slow DoS classification approach using flow data." ICT Express 7, no. 2 (2021): 210-214.

[16] Suroto, Suroto. "A review of defense against slow HTTP attack." JOIV: International Journal on Informatics Visualization 1, no. 4 (2017): 127-134.