ISSN No. 0976-5697
Volume 8, No. 5, May – June 2017
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Distributed Denial of Service Attacks – TCP Syn Flooding Attack Mitigation

Snehal Sathwara
M.Tech Cyber Security
Raksha Shakti University
Ahmedabad, India

Chandresh Parekh
Department of Telecommunication
Raksha Shakti University
Ahmedabad, India

Abstract: In this new era of digital science, networks and their capacities are significantly growing and increasing their market values. Attackers are gradually improving their skill sets by developing powerful tools to stay ahead in the world of black hat. Distributed Denial of Service Attacks (DDoS) are most dangerous attacks with the internet services and networks which is carried out in various forms such as server crashing, router crashing, slow performance of the CPU etc. Attackers implement various techniques to launch DDoS attacks on target computers or networks. In this paper, we discussed TCP syn flooding DDoS attack and its mitigation techniques to reduce attacks effect. We present a mitigation method of the TCP syn flood DDoS attacks on the Apache server by capturing attackers IP addresses and set the TCP – RST over the continues flow of SYN+ACK. It will reduce the effect of syn flooding with customised time duration. Through this method legitimate users can maintain their connection accessibility.

Keywords: DDoS Attacks, TCP syn flooding, SYN+ACK, RST, Cyber Security, IP tables, Mitigation Techniques

I. INTRODAUCTION

Confidentiality, Integrity and Availability are the main parameters of the Information Security. Distributed Denial of Service Attack is the key factor which breaks the availability parameter [1]. Transmission Control Protocol (TCP) is the protocol through which computers can connect to the internet [2]. To achieve the internet communication, TCP talks with client and server which is known as three-way handshake. In a SYN flooding attacks, the attacker sends a large number of SYN request to the victim server. The attack creates incomplete TCP connections that use up network resources [1]. Attacker exploits the three-way handshake method [1][2].
At the beginning of this paper, we describe the Distributed Denial of Service attacks followed by the TCP syn flooding DDoS attack. In the next section details involved with three-way handshake method and TCP syn flooding method. Further we describe detection methodology and analysis of network traffic and based on this we define mitigation method to reduce the impact of the TCP syn flooding DDoS attack. We monitor three-way handshake traffic of the live network and attacker's activity who exploits the three-way handshake method and cause server down with heavy flooding traffic. We implemented RST to reduce impact of TCP syn flooding on the server and block the IPs with specific time duration.

TCP Three-way handshake

When a client wants to begin a TCP connection to a server, the client and the server exchange a series of messages, as follows:

- A TCP SYN request is sent to a server
- The server sends back a SYN+ACK (acknowledgement) in response to the request and store the request information in the memory stack. This connection is known as SYN-RECV.
- The client sends a response ACK to the server to complete the session setup. This connection is moved from the SYN-RECV to ESTABLISHED.

The above method is known as TCP three-way handshake method [3].

In figure 1 it is shown that how connection is getting establish within three messages. The way of three-way handshake process is clearly captured in the Wireshark tool for the client and server which is shown in the figure 2.

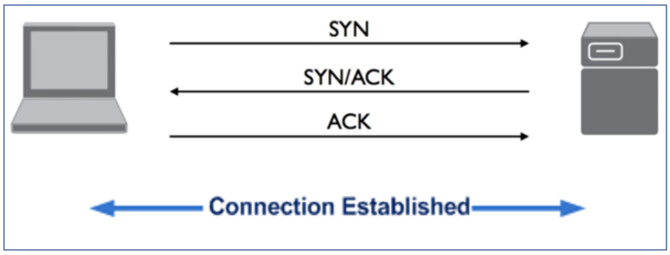


Figure 1. TCP Three-way handshake method

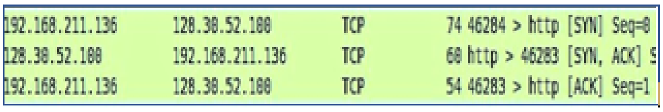


Figure 2. Example of TCP Three-way handshake in Wireshark

II. TCP SYNFLOODING ATTACK

Behavior of the Transmission Control Protocol (TCP) is exploited by the attackers that cause Syn flooding in the network. As per three-way handshake process client will send the request to the server for connection establishment and after receiving client request, server will acknowledge the request by sending acknowledge message to the client [3].

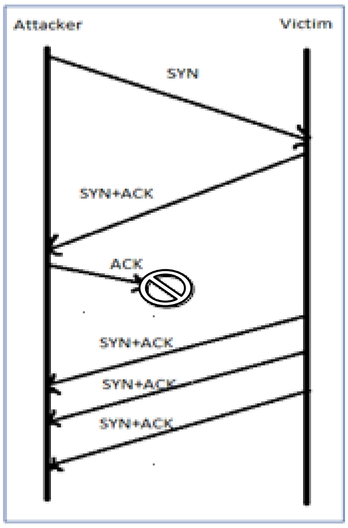


Figure 3. TCP SYN Flooding Attack

Same time, flow table entry is created and it will store receipt of the SYN packets. In the case of SYN flooding, flow table entry will be exhausted [3][4]. Attackers send connection establish request to the victim but they block the ACK as a response to the victim once they receive the SYN+ACK message from the victim which is shown in Figure 3. This is known as half open connection that cause heavy usage of victim resources resulting denial of service attack which will refuse the accessibility of the legitimate users or client.
This happens when malicious attackers send large number of SYN segments to a server by spoofing with their IP addresses. The server understand that the clients are asking for active open connection and allocates the necessary resources like creating communication tables and setting up the timer. If server is getting large number of SYN segments in short time then it runs out of resources and it may crash [5].
In order to mitigate the SYN flooding DDoS attack it is mandatory to detect and analyze the flow of flooding from the suspicious IP addresses. Mostly attackers use fake IP addresses to attack. It is necessary to monitor and captured this kind of suspicious activity in packet capture tool. Here we are going to use Wireshark packet capture tool.
In the next section, we are going to describe detection and mitigation process followed by generating DDoS SYN flooding attack.

III. METHODOLGY

A. Test bed

In order to practice DDoS SYN flood detection and mitigation we need following test environment:

- VMware Workstation to install test environment.
- Number of Ubuntu or Windows machines to generate DDoS SYN flood attacks (Acting as an attacker or legitimate clients/users).
- Ubuntu 14.04.1 machine with installed Apache2 server which will be acting as a server machine or victim.
- NMAP or ZENMAP tool to find open ports of the server machine or victim.
- Wireshark tool to capture suspicious traffic from the attacker and analysis of the captured traffic of the network after mitigation.
- PuTTY tool to check SSH connection with server after attack.
- Hping3 or Scapy script to generate TCP SYN flooding attack on the server or victim.
- Set of iptables chain.

Above environment consists of tools, operating systems and command based arguments or scripts. Operating Systems, Servers and required tools are installed in the VMware Workstation. SYN flood attacks are generated by Scapy script and with hping3 tool as per requirement of damage with respect to time duration.

In further section, we have described the generation of DDoS attack with detection analysis and mitigation results.

B. Flow of procedure

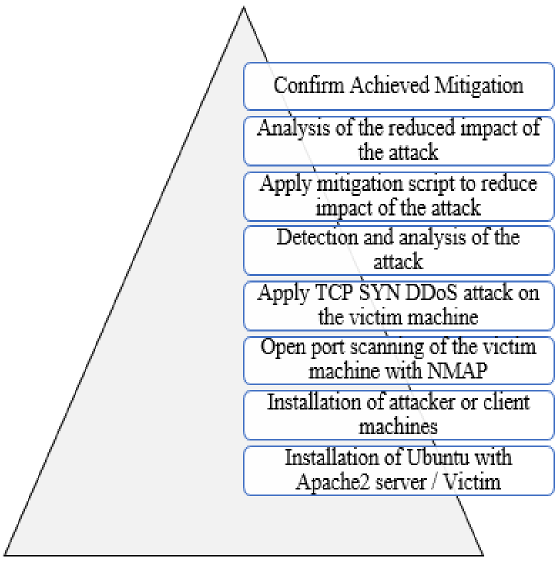


Figure 4. Flow of the Detection and Mitigation of the TCP SYN Flood

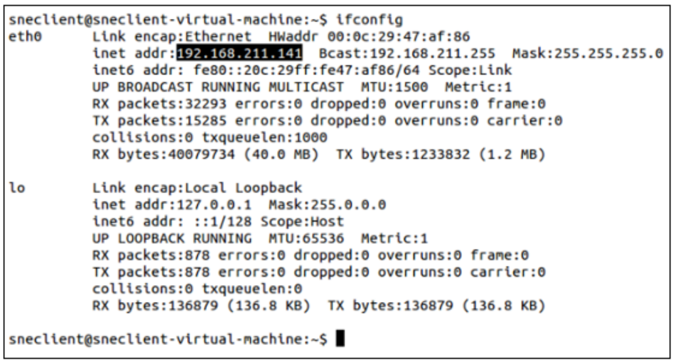
With reference of the Figure 3 we followed the steps to confirm achievement of the mitigation for TCP SYN flood attack on the top of the pyramid. We have generated SYN flood attacks on the victim server by conforming of the open ports on the victim server. By applying the mitigation method, we confirmed that reduce of the attack impact in network traffic. Mitigation script consists of following tasks:

- Set of iptables chain rules.
- Log generation with attacker IP addresses.
- Attackers identification mechanism and log into log file.
- Reducing attack impact by sending RST message.
- Setup sleeping time with respect to attack strength.

C. *TCP SYN Flood attack generation*

In this section, we have generated TCP SYN flood attack to flood the victim system. We have considered only two machines as an attacker. It may increase as per heavy flood requirement.

IP configuration of the attacker 1.



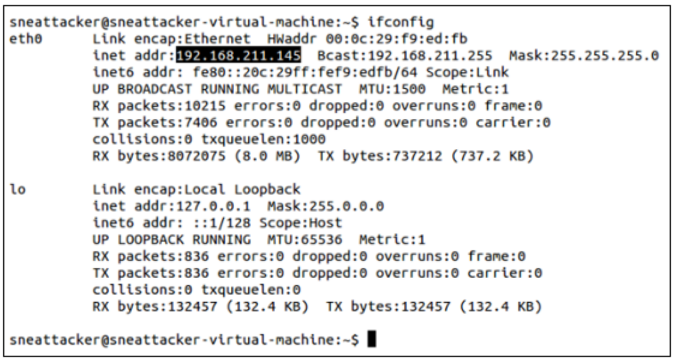
```
snecl1ent@snecl1ent-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:47:af:86
          inet addr:192.168.211.141  Bcast:192.168.211.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe47:af86/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:32293 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15285 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:40079734 (40.0 MB)  TX bytes:1233832 (1.2 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:878 errors:0 dropped:0 overruns:0 frame:0
          TX packets:878 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:136879 (136.8 KB)  TX bytes:136879 (136.8 KB)

snecl1ent@snecl1ent-virtual-machine:~$
```

Figure 5. Attacker 1 IP address

IP configuration of the attacker 2.



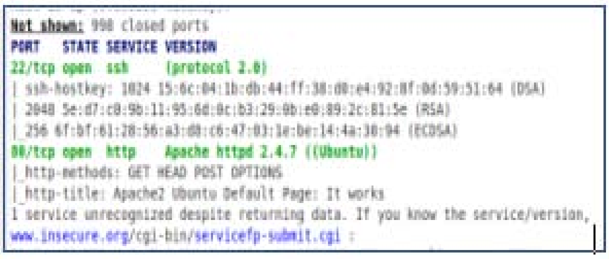
```
sneattacker@sneattacker-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:f9:ed:fb
          inet addr:192.168.211.145  Bcast:192.168.211.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fef9:edfb/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10215 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7406 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:8072075 (8.0 MB)  TX bytes:737212 (737.2 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:836 errors:0 dropped:0 overruns:0 frame:0
          TX packets:836 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:132457 (132.4 KB)  TX bytes:132457 (132.4 KB)

sneattacker@sneattacker-virtual-machine:~$
```

Figure 6. Attacker 2 IP address

Open port scan of the victim server by ZENMAP tool

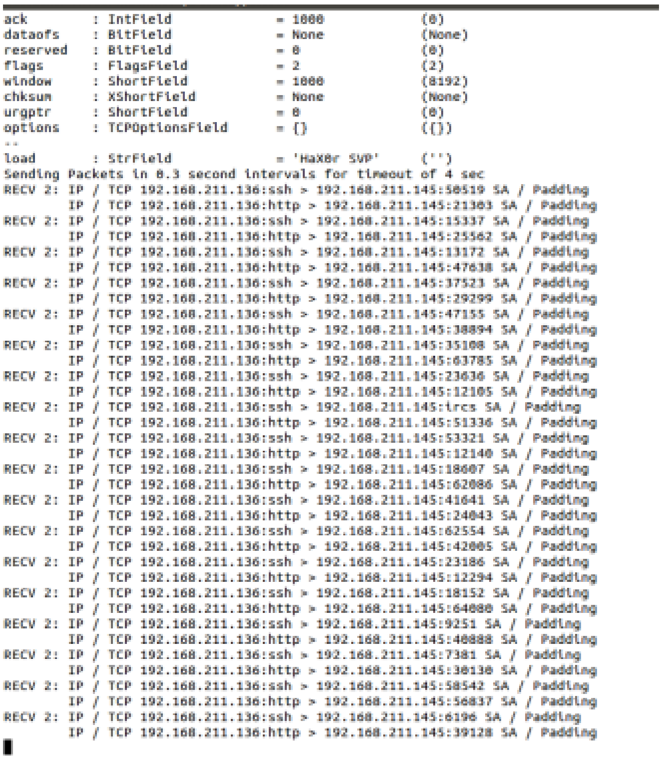


```
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     (protocol 2.0)
| ssh-hostkey: 1024 15:6c:04:1b:db:44:ff:38:d0:e4:92:8f:0d:59:51:64 (DSA)
| 2048 5e:d7:c8:9b:11:95:6d:0c:b3:29:0b:e0:89:2c:81:5e (RSA)
|_256 6f:bf:61:28:56:a3:d8:c6:47:03:1e:be:14:4a:38:94 (ECDSA)
80/tcp open  http    Apache httpd 2.4.7 ((Ubuntu))
|_http-methods: GET HEAD POST OPTIONS
|_http-title: Apache2 Ubuntu Default Page: It works
1 service unrecognized despite returning data. If you know the service/version,
www.insecure.org/cgi-bin/servicefp-submit.cgi :
```

Figure 7. Port no. 22 and Port no. 80 are open for victim machine

Attackers have applied iptables rules to stop the ACK for the victim server that cause production of unnecessary SYN+ACK to the attacker for the establishment of the connection. After setting up the iptables rules attacker will generate the flood attacks.

Applied TCP SYN flood attack by Scapy script. We can generate from hping3 tool by option.



```
ack        : IntField              = 1000          (0)
dataofs    : BitField              = None          (None)
reserved   : BitField              = 0             (0)
flags      : FlagsField            = 2             (2)
window     : ShortField            = 1000          (8192)
chksum     : XShortField           = None          (None)
urgptr     : ShortField            = 0             (0)
options    : TCPOptionsField       = {}            ({})
--
load       : StrField              = 'HaX0r SVP'   ('')
Sending Packets in 0.3 second intervals for timeout of 4 sec
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:50519 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:21303 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:15337 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:25562 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:13172 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:47638 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:37523 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:29299 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:47155 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:38894 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:35108 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:63785 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:23636 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:12105 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:ircs SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:51336 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:53321 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:12140 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:18607 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:62086 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:41641 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:24043 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:62554 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:42005 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:23186 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:12294 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:18152 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:64080 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:9251 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:40888 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:7381 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:30130 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:58542 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:56837 SA / Padding
RECV 2: IP / TCP 192.168.211.136:ssh > 192.168.211.145:6196 SA / Padding
        IP / TCP 192.168.211.136:http > 192.168.211.145:39128 SA / Padding
```

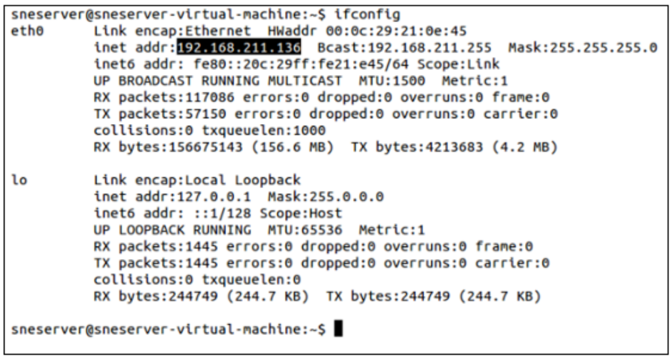
Figure 8. Flood generated on the Victim Server machine

IV. RESULT & DISCUSSION

As result of the TCP SYN flood attack, we have captured the traffic to analyze the flood attack as following:

A. *Detection of TCP SYN flood attacks on victim machine*

IP configuration of the victim/server machine.



```
sneserver@sneserver-virtual-machine:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:21:0e:45
          inet addr:192.168.211.136  Bcast:192.168.211.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe21:e45/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:117086 errors:0 dropped:0 overruns:0 frame:0
          TX packets:57150 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:156675143 (156.6 MB)  TX bytes:4213683 (4.2 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1445 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1445 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:244749 (244.7 KB)  TX bytes:244749 (244.7 KB)

sneserver@sneserver-virtual-machine:~$
```

Figure 9. IP Address of the victim/server

After DDoS flood attack server status

```
tcp        0      0 192.168.211.136:ssh         192.168.211.141:64535       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:48390       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:16348       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:22711       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:47372       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:914         SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:11647       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:14593       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:12303       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:46109       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:35272       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:46984       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:41917       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:48876       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:24310       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:8502        SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:51918       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:10275       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:44079       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:24265       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:38703       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:15573       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:53470       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:45861       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:42287       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:56867       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:28559       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:56415       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:27703       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:50529       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:9186        SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:18496       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:56197       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:31983       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:593         SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:41026       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:6805        SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:14618       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:17188       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:37609       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:30677       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:39524       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:39563       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:17249       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:47952       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:22570       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.145:52487       SYN_RECV
tcp        0      0 192.168.211.136:ssh         192.168.211.141:58026       SYN_RECV
```

Figure 10. SYN_RECV status on server after attack

Overall traffic performance of the victim machine in kbps bandwidth

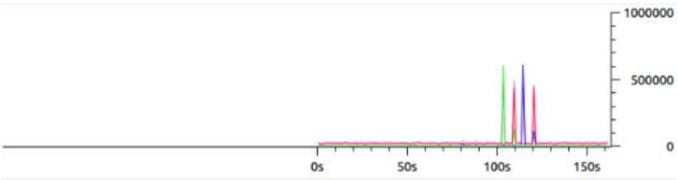


Figure 11. Victim machine performance approx. 500 kbps

B. Mitigation of TCP SYN flood attacks on victim machine

Applying mitigation method on the victim machine.

2149	53.15711100	192.168.211.136	192.168.211.141	TCP	54 ssh > 8735 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2150	53.16435100	192.168.211.145	192.168.211.136	SSH	63 Encrypted request packet len=9
2151	53.16439200	192.168.211.136	192.168.211.145	TCP	54 ssh > 32028 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2152	53.16496600	192.168.211.141	192.168.211.136	TCP	63 [TCP segment of a reassembled PDU]
2153	53.16499700	192.168.211.136	192.168.211.141	TCP	54 http > 9759 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2154	53.17090900	192.168.211.145	192.168.211.136	TCP	63 [TCP segment of a reassembled PDU]
2155	53.17093800	192.168.211.136	192.168.211.145	TCP	54 http > 19874 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2156	53.44011100	192.168.211.145	192.168.211.136	SSH	63 Encrypted request packet len=9
2157	53.44015200	192.168.211.136	192.168.211.145	TCP	54 ssh > 58142 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2158	53.44343800	192.168.211.145	192.168.211.136	TCP	63 [TCP segment of a reassembled PDU]
2159	53.44348100	192.168.211.136	192.168.211.145	TCP	54 http > 35133 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2160	53.44803200	192.168.211.141	192.168.211.136	SSH	63 Encrypted request packet len=9
2161	53.44809500	192.168.211.136	192.168.211.141	TCP	54 ssh > 60102 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2162	53.45022400	192.168.211.141	192.168.211.136	TCP	63 [TCP segment of a reassembled PDU]
2163	53.45025400	192.168.211.136	192.168.211.141	TCP	54 http > 34645 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2164	53.49365400	192.168.211.1	239.255.255.250	UDP	698 Source port: 63736 Destination port: ws-discover
2165	53.74701100	192.168.211.141	192.168.211.136	SSH	63 Encrypted request packet len=9
2166	53.74708400	192.168.211.136	192.168.211.141	TCP	54 ssh > 53866 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2167	53.75213500	192.168.211.141	192.168.211.136	TCP	63 [TCP segment of a reassembled PDU]
2168	53.75216800	192.168.211.136	192.168.211.141	TCP	54 http > 53536 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2169	53.75310300	192.168.211.145	192.168.211.136	SSH	63 Encrypted request packet len=9
2170	53.75312300	192.168.211.136	192.168.211.145	TCP	54 ssh > 60860 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2171	53.75811400	192.168.211.145	192.168.211.136	TCP	63 [TCP segment of a reassembled PDU]
2172	53.75815000	192.168.211.136	192.168.211.145	TCP	54 http > 14912 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0
2173	54.04144700	192.168.211.141	192.168.211.136	SSH	63 Encrypted request packet len=9
2174	54.04148100	192.168.211.136	192.168.211.141	TCP	54 ssh > 10842 [RST, ACK] Seq=1 Ack=10 Win=0 Len=0

Figure 12. Set up of the RST message for the attackers

Overall traffic performance of the server after the mitigation.

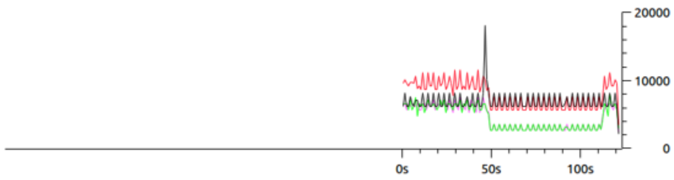


Figure 13. Impact of mitigation is reduced by approx. 20 kbps

Log captured in log files.

```
Thu May  4 04:17:56 IST 2017
     20 192.168.211.141
     23 192.168.211.145
Thu May  4 04:18:58 IST 2017
```

Figure 14. Captured attackers IP addresses in the log file

V. CONCLUSION & FUTURE WORK

Internet service provider and network capacities are increasing their market space. Skilled hackers are damaging the revenue cycle of the big corporate by applying various DDoS attack. In which TCP SYN flooding attack is the most dangerous and long-time attack that cause unavailability of servers to the legitimate users or clients. This attack is very strong to interrupt server activities for that we implemented our detection technique using shell script. So, this shell code can be used to make application which can run to mitigate the TCP SYN attack that cause the reducing of the attack impact. In this method, we continuously applying RST signal and removing all iptables chain and based on requirement we can make sleep the system for the specific time duration.

We studied one of the DDoS attack that is "TCP SYN flooding DDoS attack" in the network. We have studied about the existing techniques on detection of TCP SYN flood attack. So, in our dissertation work, we tried to achieve the defined objectives and explore some dedicated work on that.

VI. REFERENCES

[1] Gilang Ramadhan, Yusuf Kurniawan, Chang-Soo Kim "Design of TCP SYN Flood DDoS Attack Detection Using Artificial Immune Systems", 2016 IEEE 6th International Conference on System Engineering and Technology (ICSET), October 3-4, 2016 Bandung – Indonesia, Pg. 72-76

[2] Abdulaziz Aborujilah, Mohd Nazri Ismail, Shahrulniza Musa, "Detecting TCP SYN Based Flooding Attacks by Analyzing CPU and Network Resources Performance", 2014 3rd International Conference on Advanced Computer Science Applications and Technologies, Pg. 157-161.

[3] Pie-E Liu, Zhong-Hua Sheng, "Defending against tcp syn flooding with a new kind of syn-agent" Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008, Pg. 1218-1221

[4] Samad S. Kolahi, Amro A. Alghalbi, Abdulmohsen F. Alotaibi; Saarim S. Ahmed; Divyesh Lad 2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops

(ICUMT) "Performance comparison of defense mechanisms against TCP SYN flood DDoS attack" Pages: 143 - 147

[5] L. Kavisankar; C. Chellappan 2011 International Conference on Recent Trends in Information Technology (ICRTIT) "A mitigation model for TCP SYN flooding with IP spoofing" Pages: 251 - 256