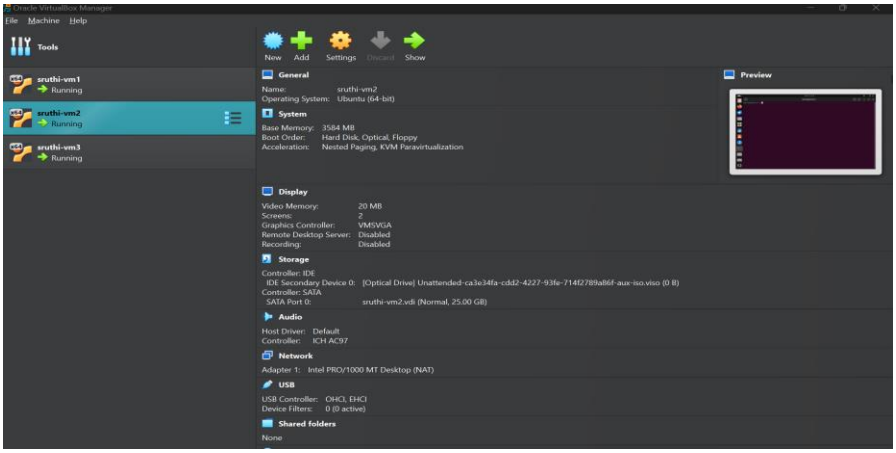
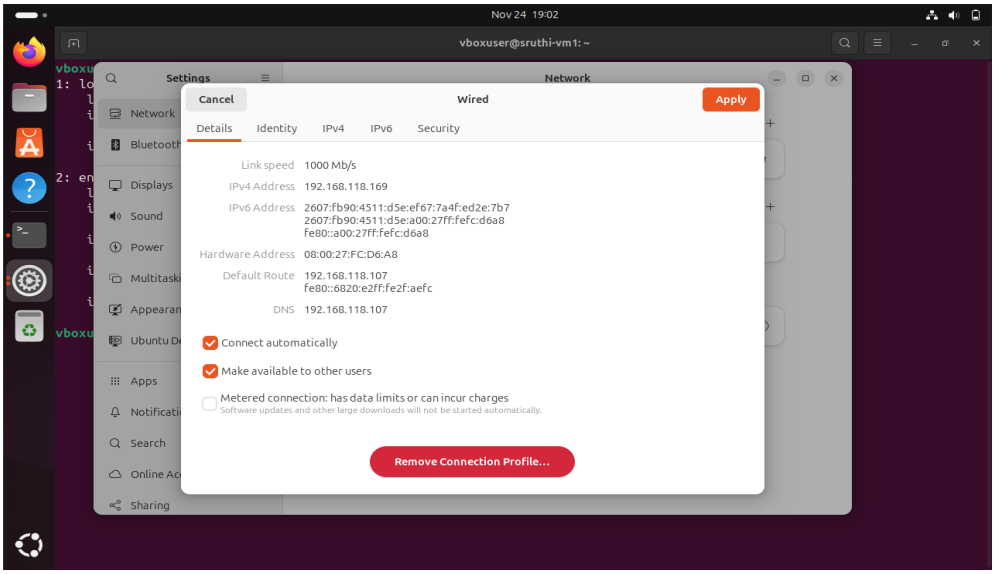
PART1:

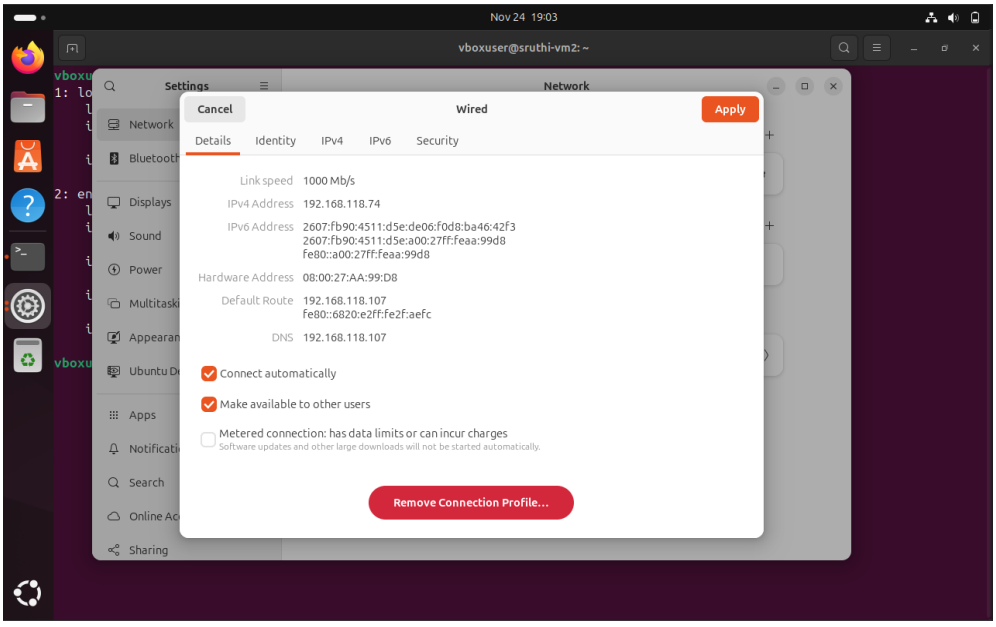
A: Configure Virtual Machines (VMs)



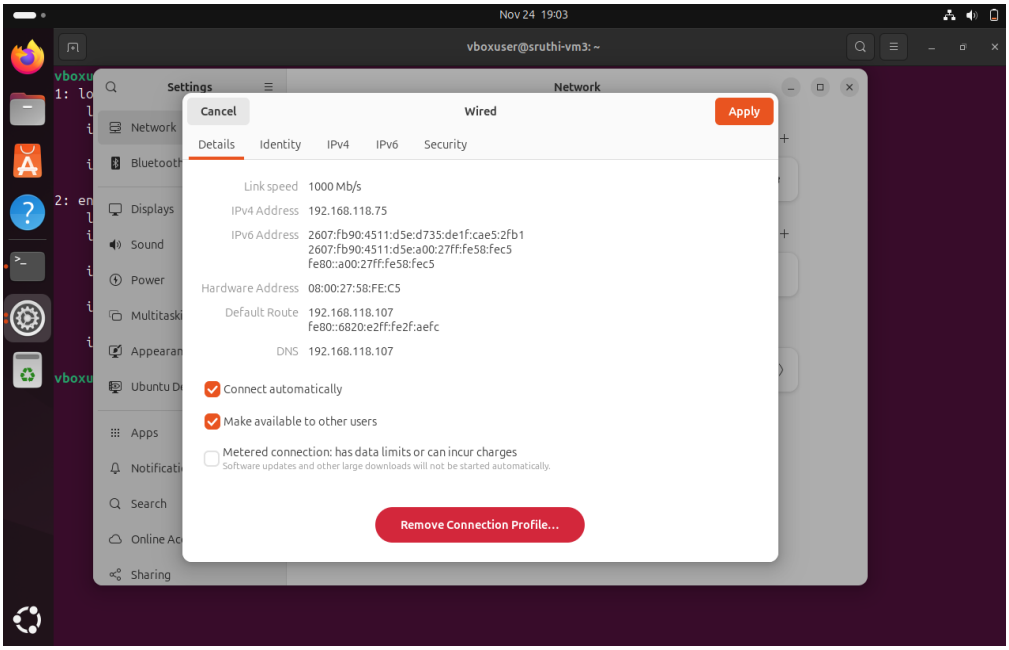
VM1 ip-address:



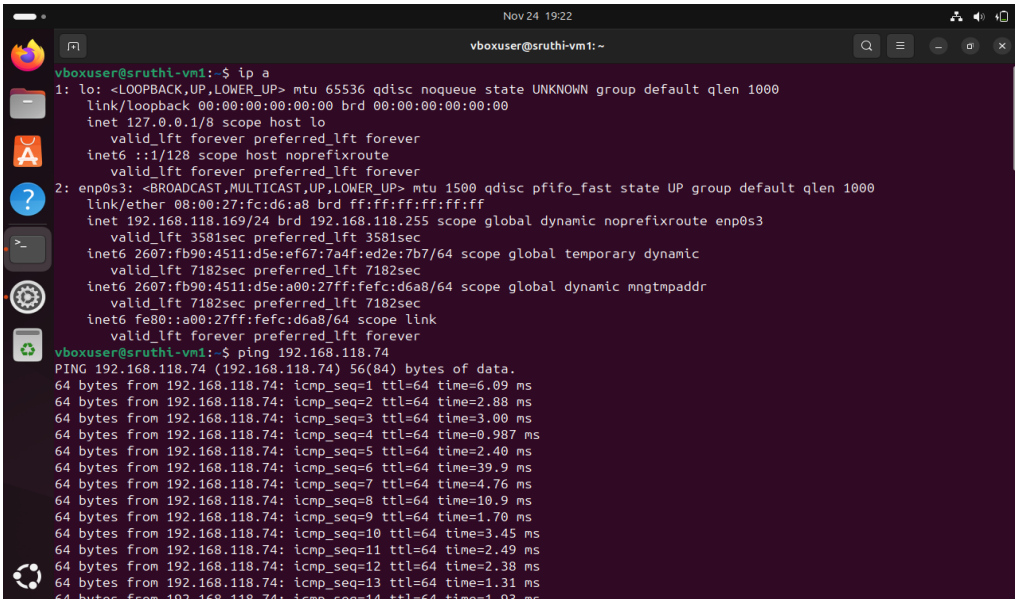
VM2 ip-address:

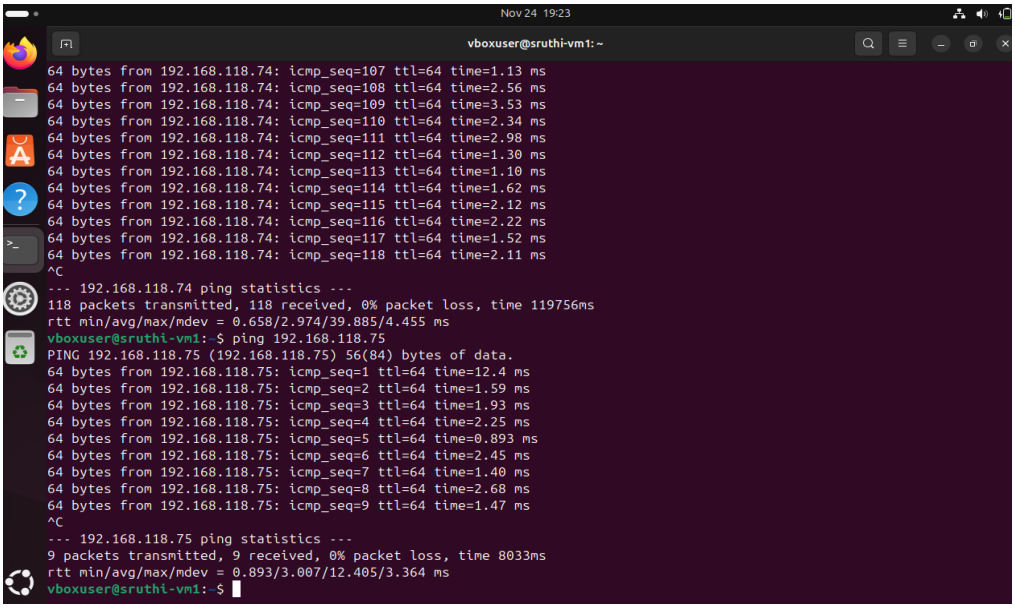


VM3 ip-address:

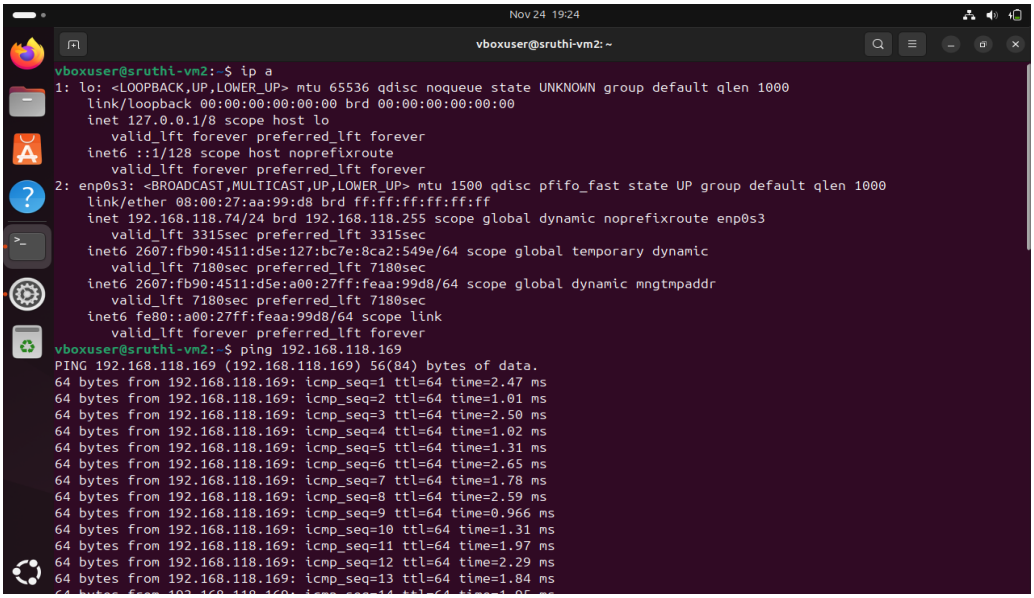


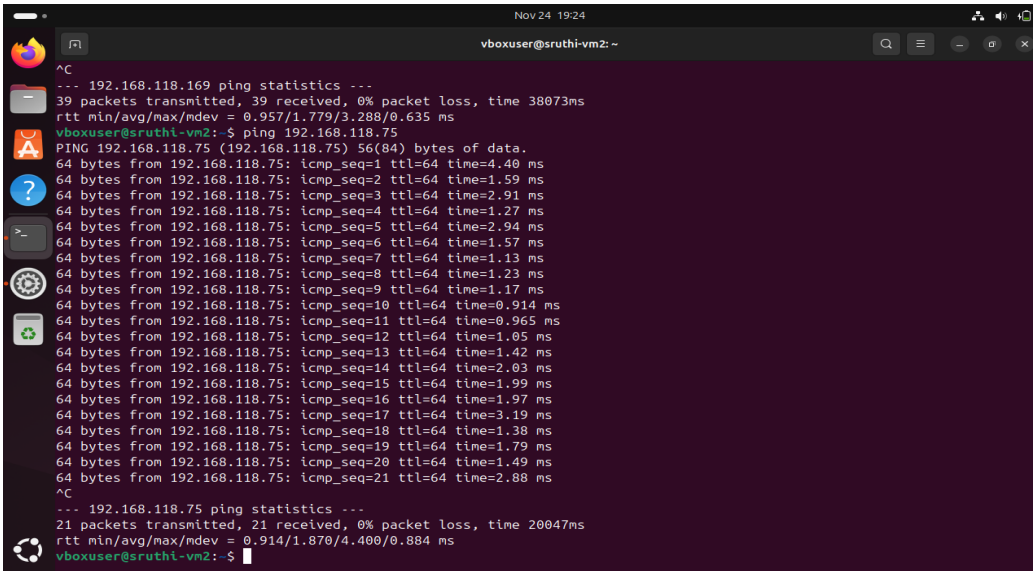
Ping requests – vm1



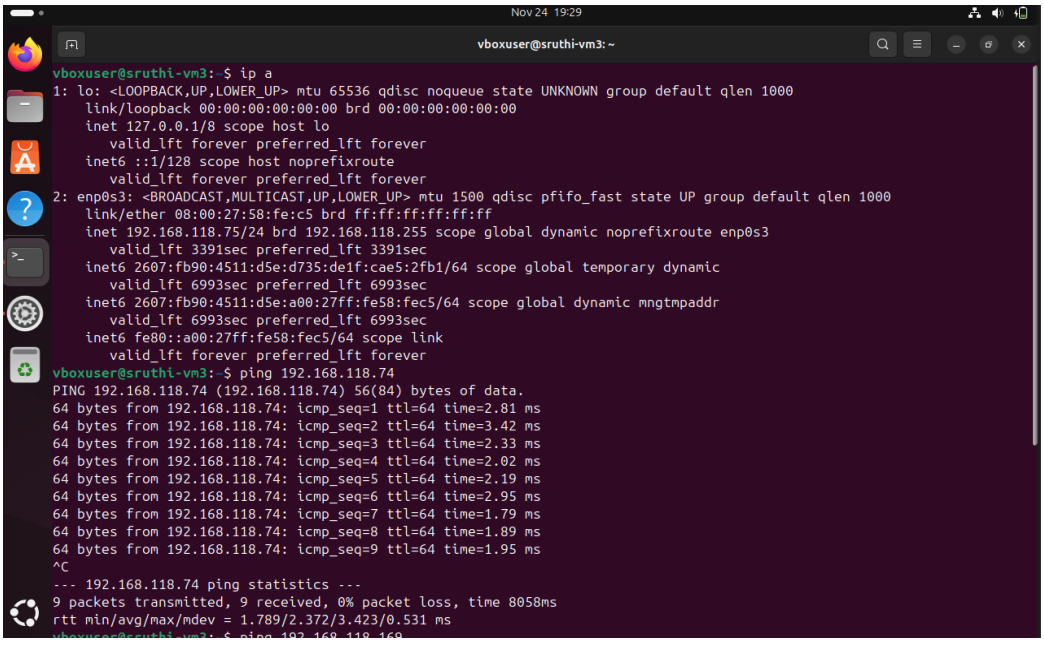


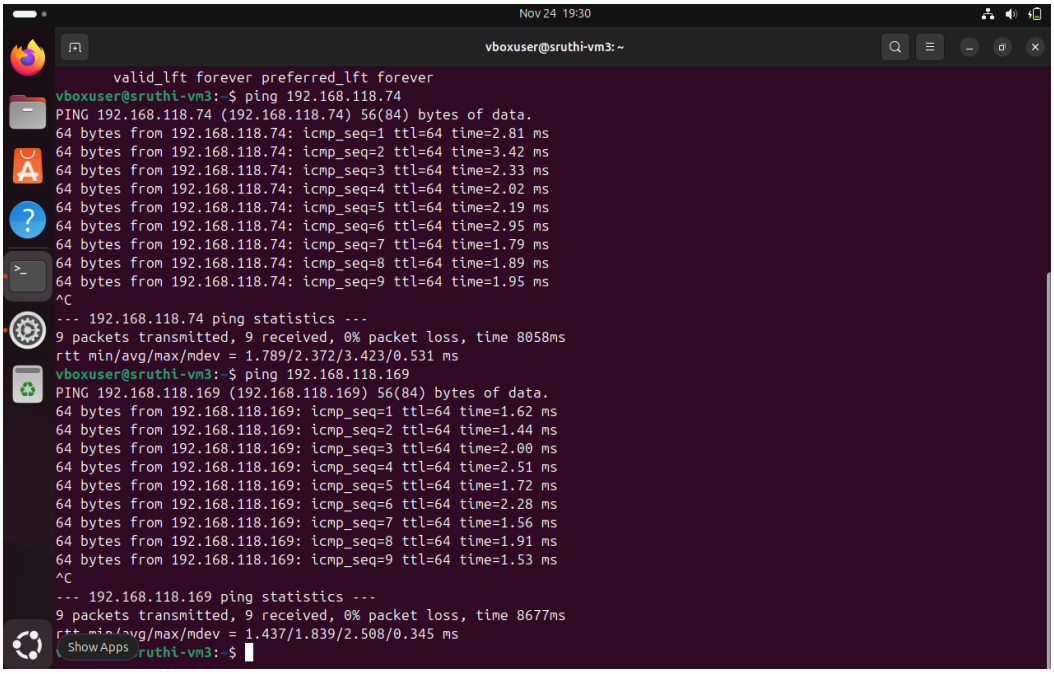
Ping requests - vm2:





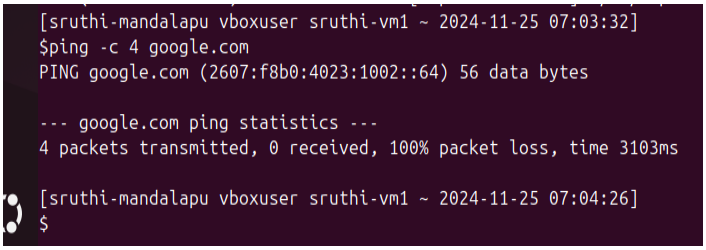
Ping requests - vm3:



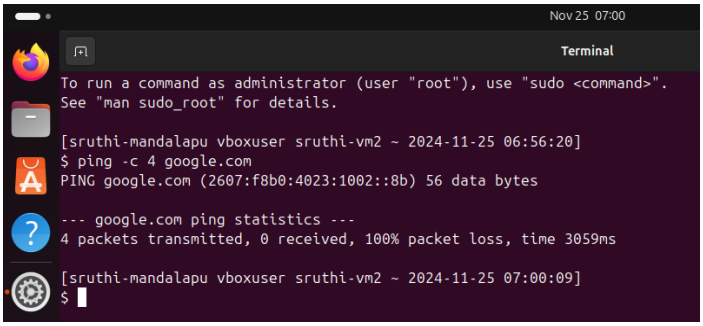


Each VM Internet Access:

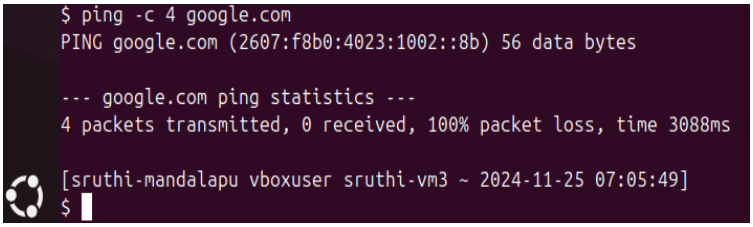
VM1:



VM2:



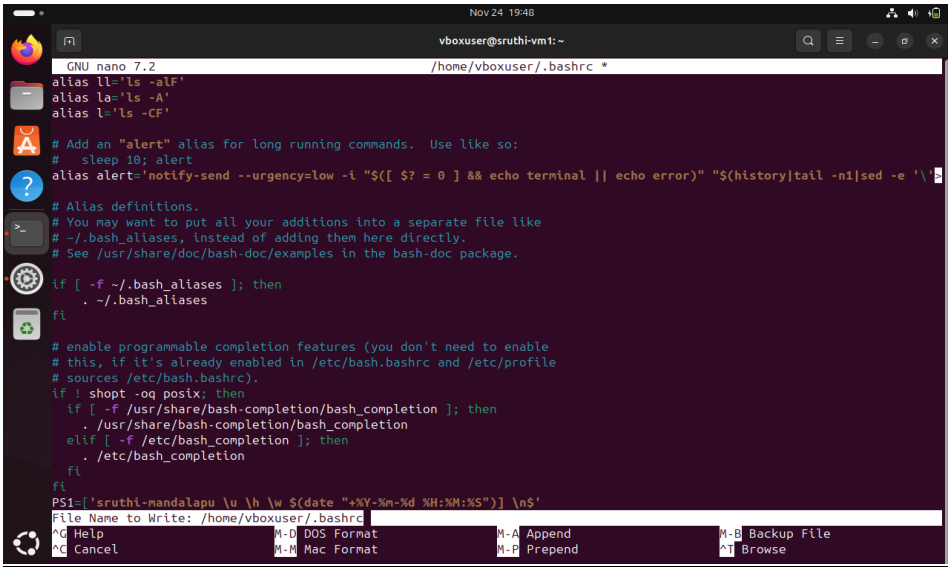
VM3:



B: Customize the bash prompt

VM1:

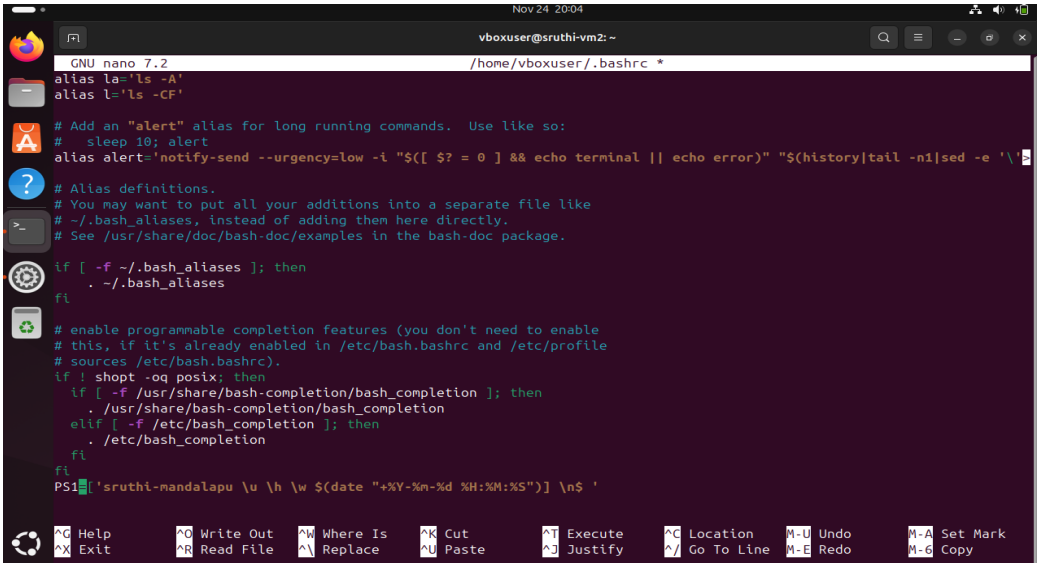
```
--- 192.168.118.74 ping statistics ---
118 packets transmitted, 118 received, 0% packet loss, time 119756ms
rtt min/avg/max/mdev = 0.658/2.974/39.885/4.455 ms
vboxuser@sruthi-vm1:~$ ping 192.168.118.75
PING 192.168.118.75 (192.168.118.75) 56(84) bytes of data.
64 bytes from 192.168.118.75: icmp_seq=1 ttl=64 time=12.4 ms
64 bytes from 192.168.118.75: icmp_seq=2 ttl=64 time=1.59 ms
64 bytes from 192.168.118.75: icmp_seq=3 ttl=64 time=1.93 ms
64 bytes from 192.168.118.75: icmp_seq=4 ttl=64 time=2.25 ms
64 bytes from 192.168.118.75: icmp_seq=5 ttl=64 time=0.893 ms
64 bytes from 192.168.118.75: icmp_seq=6 ttl=64 time=2.45 ms
64 bytes from 192.168.118.75: icmp_seq=7 ttl=64 time=1.40 ms
64 bytes from 192.168.118.75: icmp_seq=8 ttl=64 time=2.68 ms
64 bytes from 192.168.118.75: icmp_seq=9 ttl=64 time=1.47 ms
^C
--- 192.168.118.75 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8033ms
rtt min/avg/max/mdev = 0.893/3.007/12.405/3.364 ms
vboxuser@sruthi-vm1:~$ nano ~/.bashrc
```

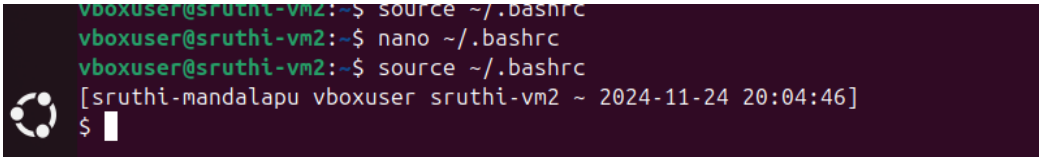



```
64 bytes from 192.168.118.75: icmp_seq=5 ttl=64 time=0.893 ms
64 bytes from 192.168.118.75: icmp_seq=6 ttl=64 time=2.45 ms
64 bytes from 192.168.118.75: icmp_seq=7 ttl=64 time=1.40 ms
64 bytes from 192.168.118.75: icmp_seq=8 ttl=64 time=2.68 ms
64 bytes from 192.168.118.75: icmp_seq=9 ttl=64 time=1.47 ms
^C
--- 192.168.118.75 ping statistics ---
9 packets transmitted, 9 received, 0% packet loss, time 8033ms
rtt min/avg/max/mdev = 0.893/3.007/12.405/3.364 ms
vboxuser@sruthi-vm1:~$ nano ~/.bashrc
vboxuser@sruthi-vm1:~$ source ~/.bashrc
[sruthi-mandalapu vboxuser sruthi-vm1 ~ 2024-11-24 19:49:30]
$
```

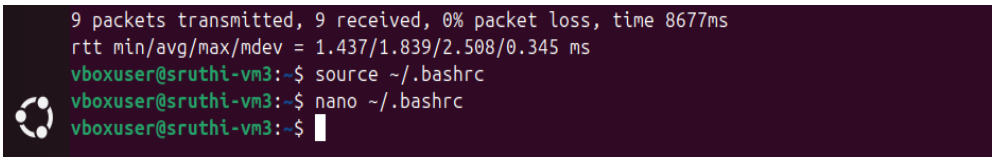
VM2:

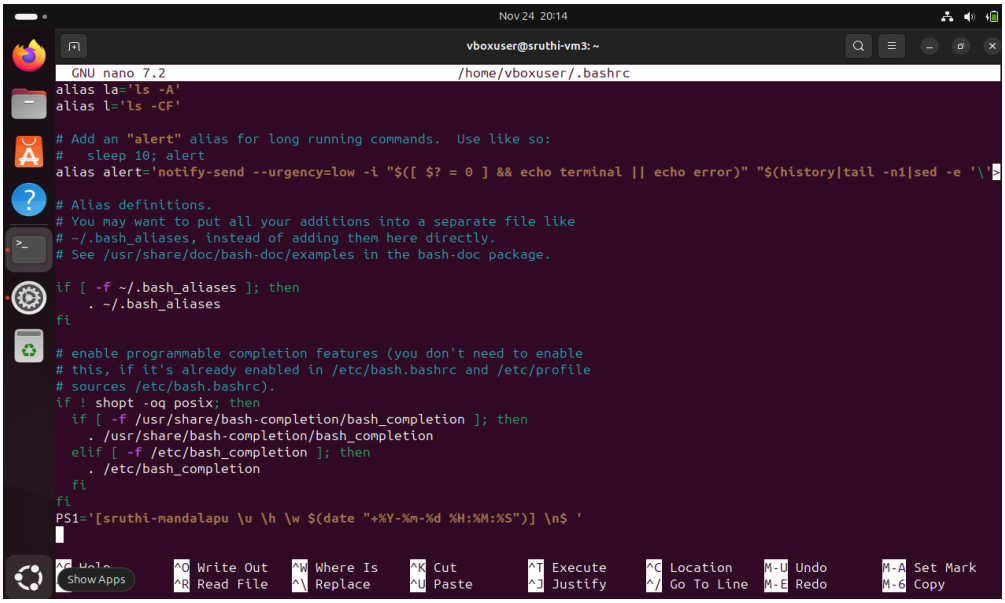
```
64 bytes from 192.168.118.75: icmp_seq=13 ttl=64 time=1.42 ms
64 bytes from 192.168.118.75: icmp_seq=14 ttl=64 time=2.03 ms
64 bytes from 192.168.118.75: icmp_seq=15 ttl=64 time=1.99 ms
64 bytes from 192.168.118.75: icmp_seq=16 ttl=64 time=1.97 ms
64 bytes from 192.168.118.75: icmp_seq=17 ttl=64 time=3.19 ms
64 bytes from 192.168.118.75: icmp_seq=18 ttl=64 time=1.38 ms
64 bytes from 192.168.118.75: icmp_seq=19 ttl=64 time=1.79 ms
64 bytes from 192.168.118.75: icmp_seq=20 ttl=64 time=1.49 ms
64 bytes from 192.168.118.75: icmp_seq=21 ttl=64 time=2.88 ms
^C
--- 192.168.118.75 ping statistics ---
21 packets transmitted, 21 received, 0% packet loss, time 20047ms
rtt min/avg/max/mdev = 0.914/1.870/4.400/0.884 ms
vboxuser@sruthi-vm2:~$ nano ~/.bashrc
```

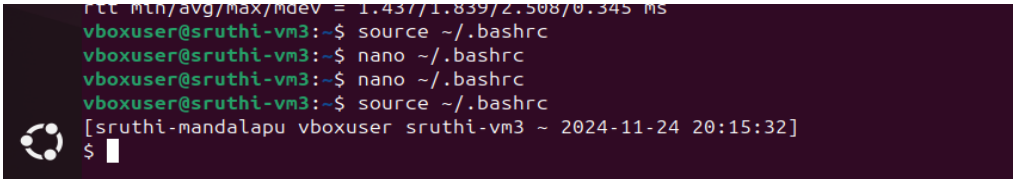




VM3:





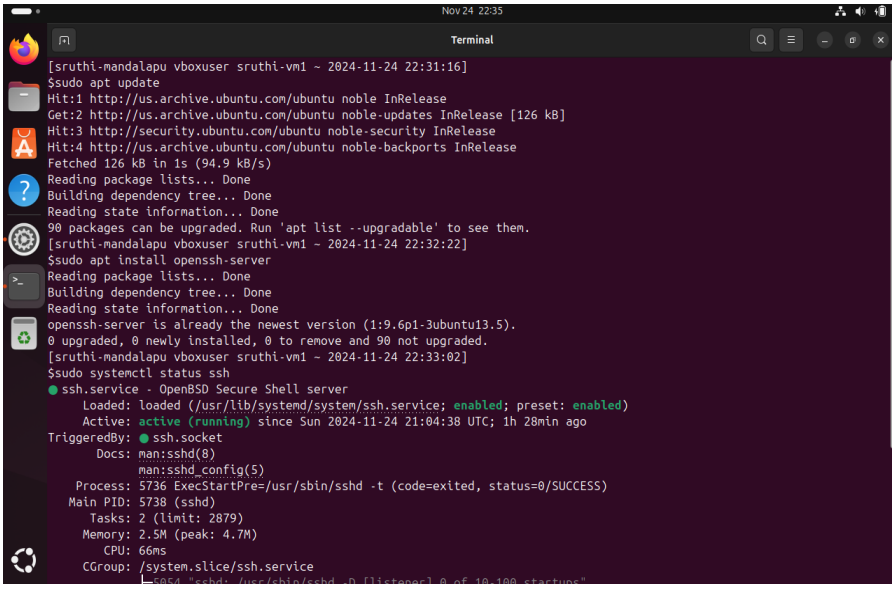


PART2: Baseline Analysis (Before SYN Flood Attack)

a)

Enable SSH on VM1: commands
sudo apt update
sudo apt install openssh-server

sudo systemctl status ssh

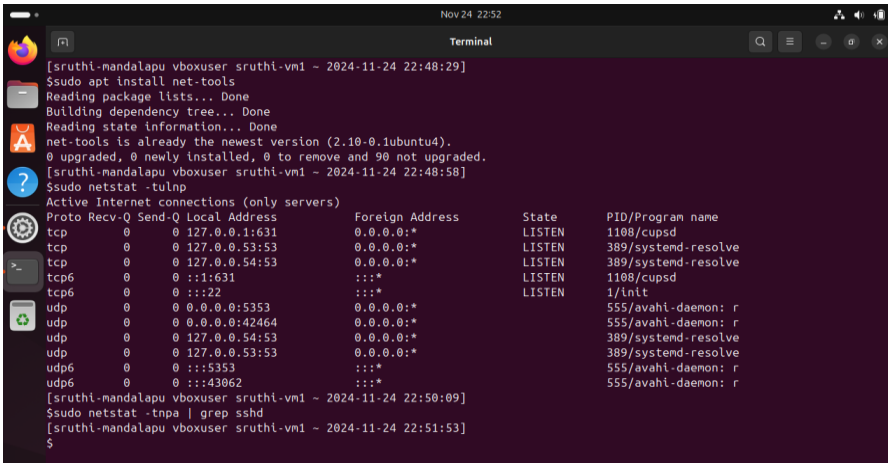


Initial State on VM1: commands

sudo apt install net-tools

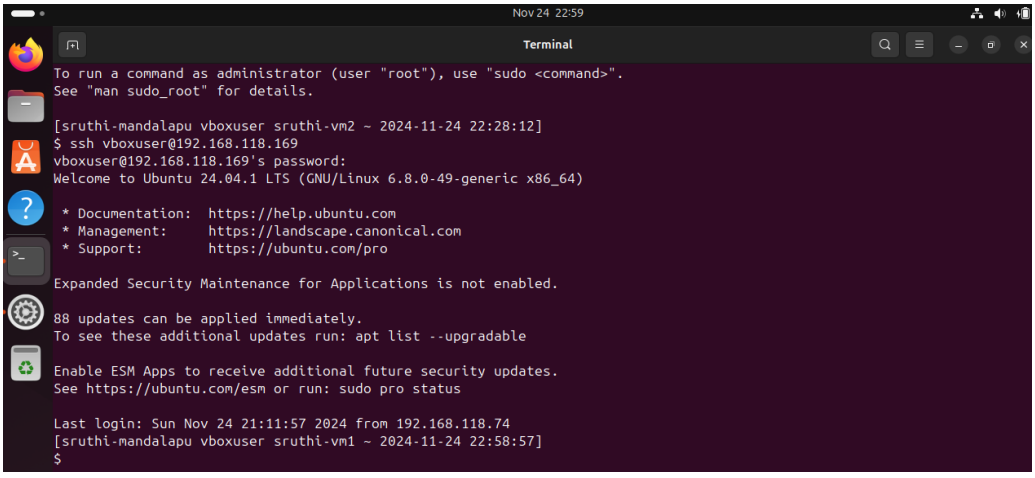
sudo netstat -tulnp → view active network connections

sudo netstat -tnpa | grep sshd → to specifically view ssh connections



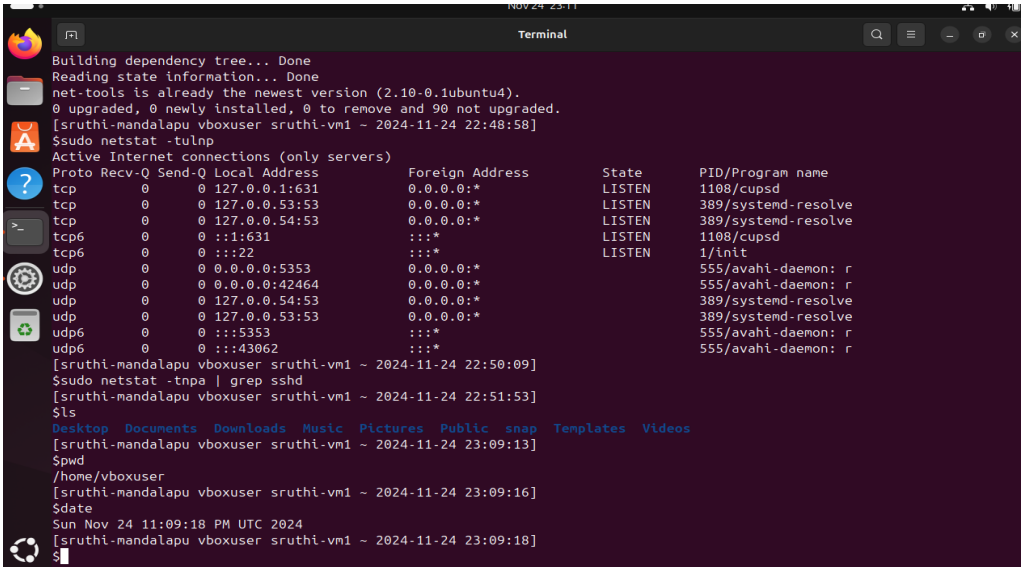
Use VM2 to Establish SSH connection with VM1: commands

ssh username_of_VM1@VM1_IPAddress



Execute series of commands during SSH session: commands

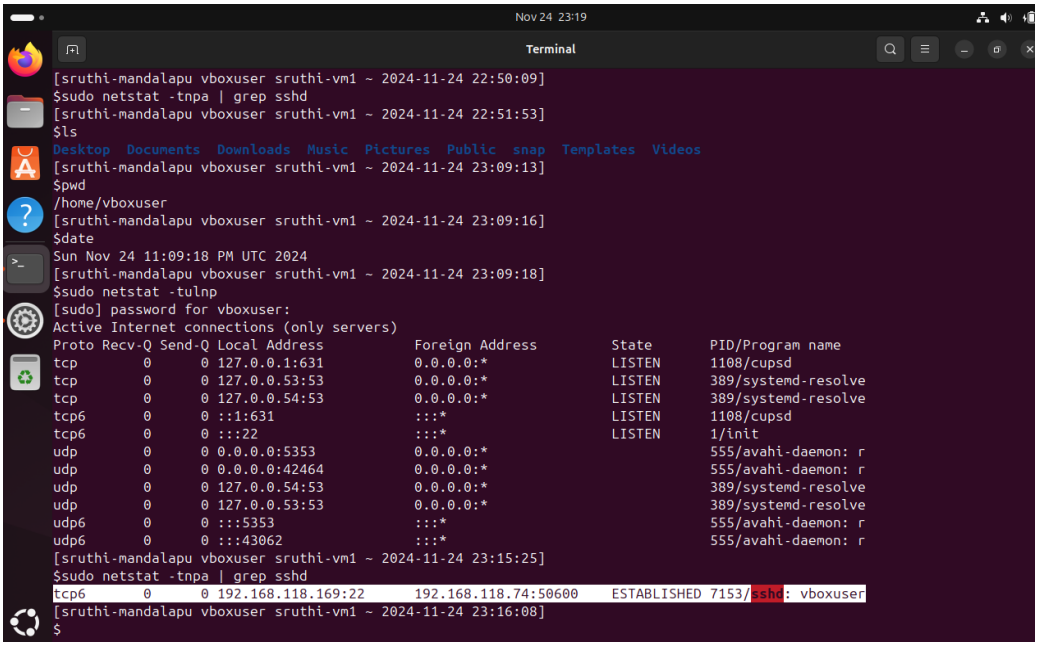
ls, pwd, date



Pre-Attack State on VM1: commands

sudo netstat -tulnp → view active network connections

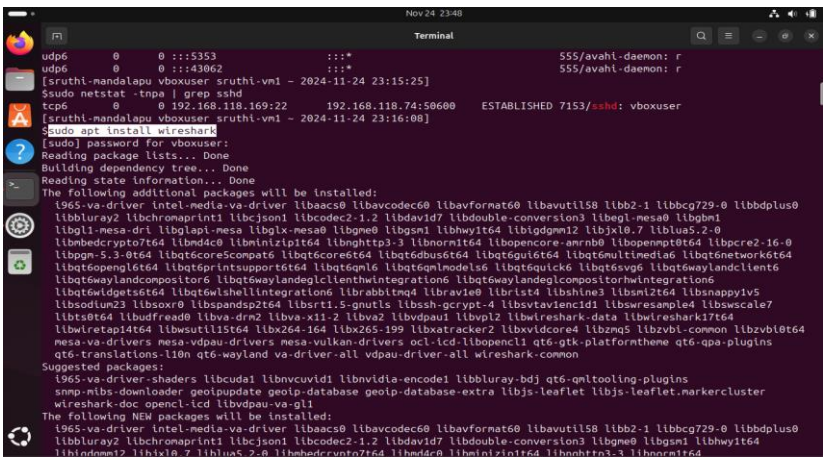
sudo netstat -tnpa | grep sshd → to specifically view ssh connections

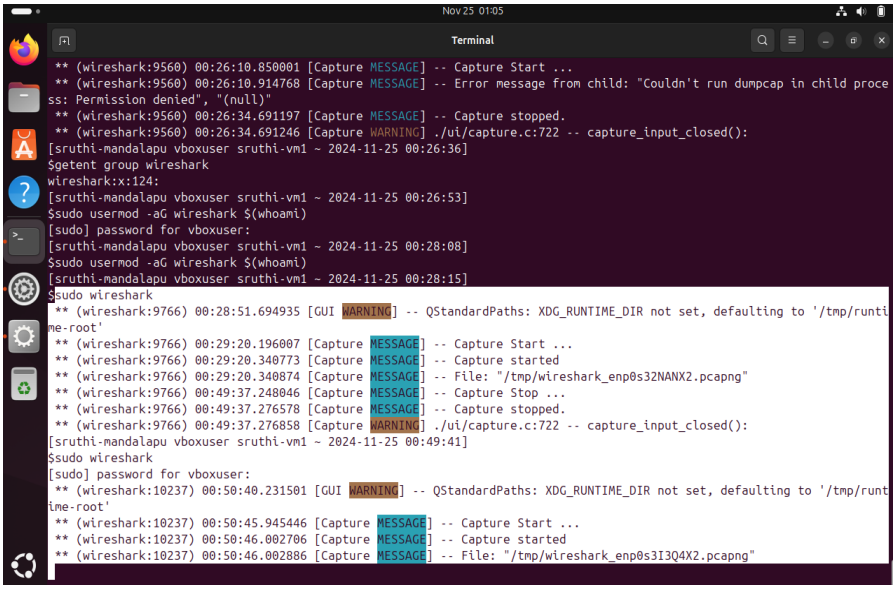


As we can observe there is a difference in initial state and pre-attack state for ssh connection. For initial state there are no ssh connection, if we observe there is a ssh connection established in pre-attack state. And the connection is between vm1 and vm2. We can find their ip addresses.

b)

Install wireshark:

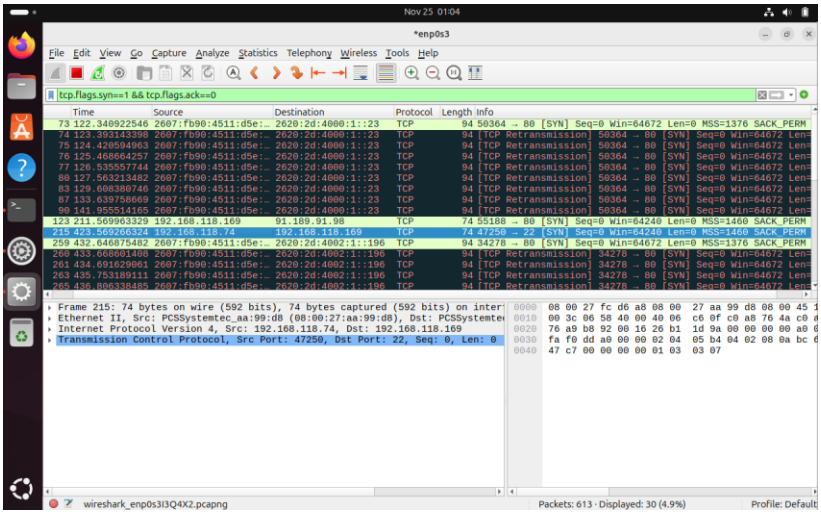




TCP Connection State: SYN_SENT

The main working of syn_sent is the client sends a syn (synchronize) packet to initiate a connection with the server. In wireshark look for SYN flag set in a packet with no ACK. It indicates that the client is attempting to establish a connection.

For SYN_SENT – the syn flag is set, but ack flag is not set which explains about syn_sent state. Syn sent is happens before getting acknowledgement from the receiver, hence ack is '0' which is not set, syn is '1' which is set. So, within Wireshark I have used filter command – "tcp.flags.syn==1 && tcp.flags.ack==0". This gives the syn_sent requests – my VM1 ip-address is 192.168.118.169, my VM2 ip-address is 192.168.118.74. As you can observe the blue color highlighted things VM2 made a syn_sent to VM1, and also we could observe their respective source and destination ip-addresses.

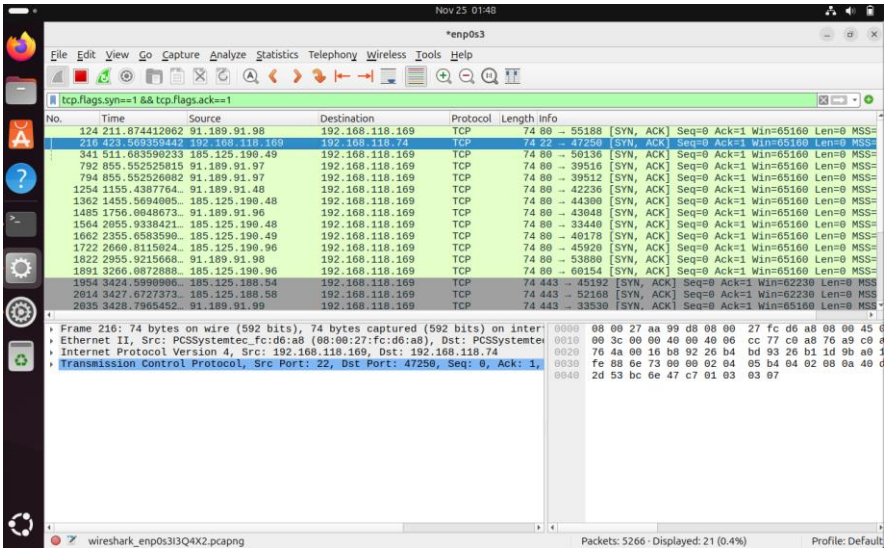


TCP Connection State: SYN_RECEIVED

The main working of syn_received is the server receives the syn packet and responds with a syn-ack packet. In wireshark look for a packet with both syn and ack flags are set. It shows the server is ready to establish the connection and awaits acknowledgement from the client.

For SYN_RECEIVED – the syn flag is set, but ack flag is set which explains about syn_received state. Syn received happens to give acknowledgement to the sender, it is generated from receiver, hence ack is '1' which is set, syn is '1' which is set. So, within Wireshark I have used filter command – "tcp.flags.syn==1 && tcp.flags.ack==1". This gives the syn_received requests – my VM1 ip-address is 192.168.118.169, my VM2 ip-address is 192.168.118.74. As you can observe the blue color highlighted things VM1 made a syn_received acknowledgement to VM1, and also we could observe their respective source and destination ip-addresses. As,

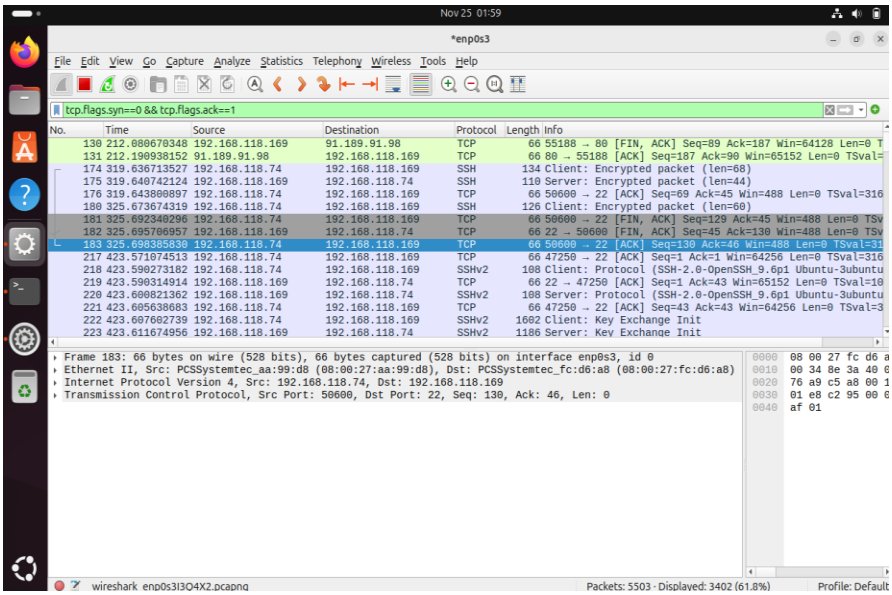
the acknowledgement need to be sent from vm1 – hence source ip-address is vm1, and vm2 as destination ip-addresses.



TCP Connection State: ESTABLISHED

The main working of established is the client acknowledges the server's SYN_ACK, completing the handshake. And then data transfer begins. In wireshark there is continuous flows of packets between client and server. This indicates an active connection.

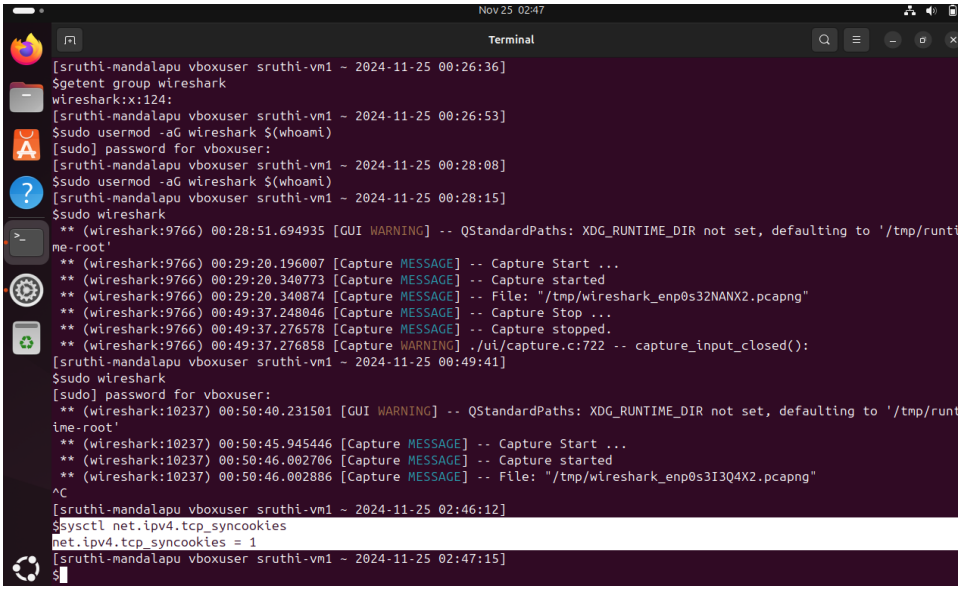
For ESTABLISHED – the syn flag is not set, but ack flag is set which explains about established state. Established state happens when the connection is maintained between the source and destination, it is generated as ack is '1' which is set, syn is '0' which is not set. So, within Wireshark I have used filter command – "tcp.flags.syn==0 && tcp.flags.ack==1". This gives the connection established requests – my VM1 ip-address is 192.168.118.169, my VM2 ip-address is 192.168.118.74. As you can observe the blue color highlighted things VM2 made a established acknowledgement to VM1, and also we could observe their respective source and destination ip-addresses. As, the acknowledgement need to be sent from vm2 – hence source ip-address is vm2, and vm1 as destination ip-addresses.



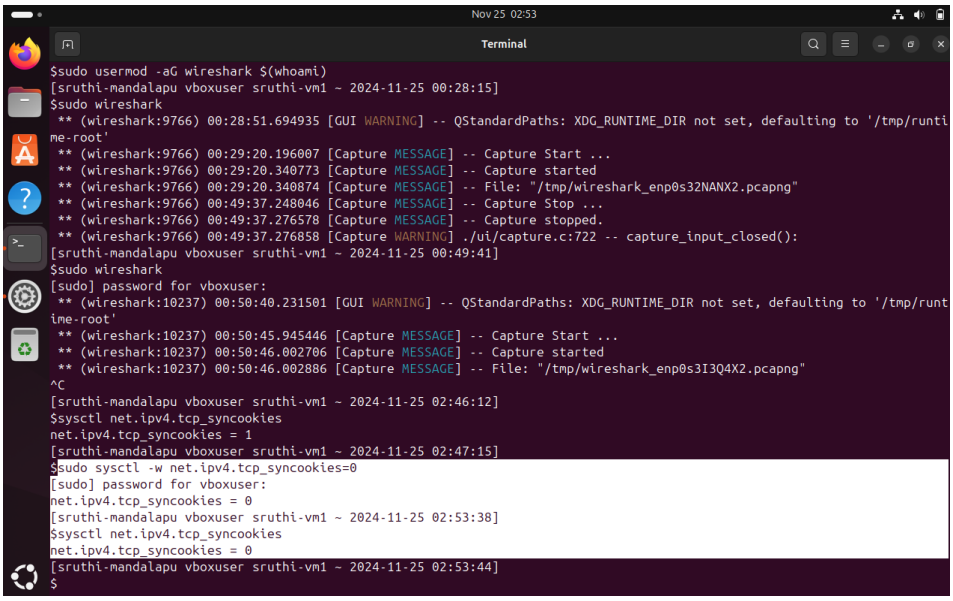
PART3: Launching a SYN Flood Attack

a)

Command to check if cookies are enabled or not: sysctl net.ipv4.tcp_syncookies, as it is 1 – the cookies are enabled



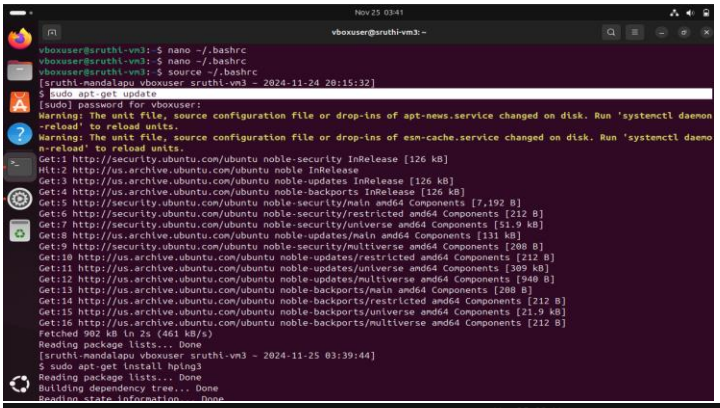
To disable cookies: sudo sysctl -w net.ipv4.tcp_syncookies=0

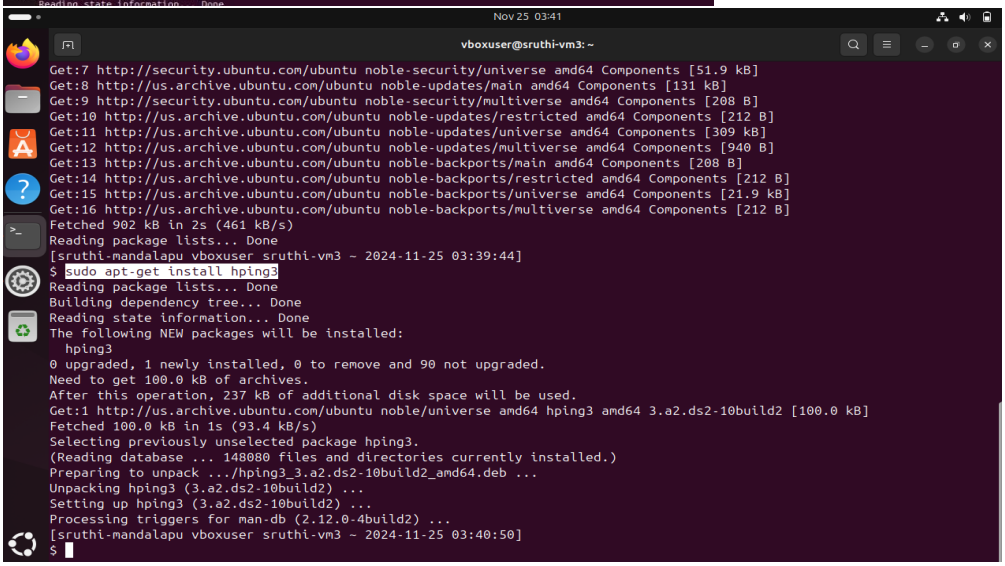


b)

DOS (Deniel-of-Service) attack software is designed to flood a target system with an overwhelming volume of network requests, rendering it unable to respond to legitimate traffic. Among many tools available, hping3 stands out due to its flexibility and wide adoption in network testing. This hping can be used it in a flexible and customizable way. The main features are: It supports customizable packet crafting – means it allows precise control over packet headers and flags. Multi-protocol support – Works with TCP, UDP, ICMP, and raw IP Packets. High throughput – Can send packet at very high rate, making it suitable for SYN flood attacks.
The setup:
VM1: Target Machine running on server TCP
VM3: Attacker machine equipped with hping3
VM Network: Both Machines are on same sub network

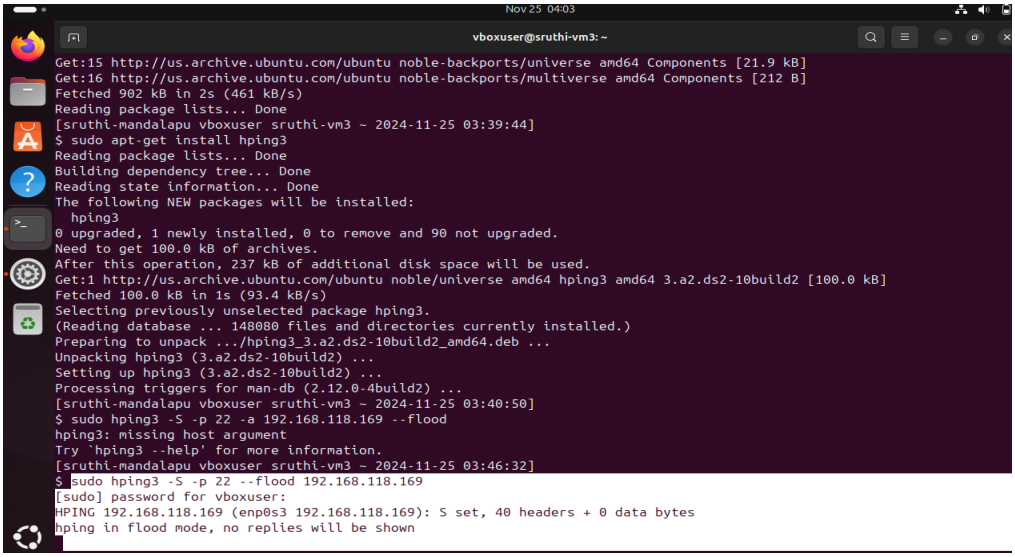
Install DOS Attack Software: commands
In vm3:
sudo apt-get update
sudo apt-get install hping3





Launch the SYN Flood Attack: commands

sudo hping3 -S -p 22 --flood 192.168.118.169 → where ip-address of vm1 is 192.168.118.169



c)

Software Used:

The software I have used for the attack is hping software. Hping is a popular command-line tool often used for network testing. It can craft custom TCP, UDP, ICMP and raw IP packets. Hping is helpful in Firewall testing, Network Analysis. Misusing hping sends large volumes of packets can cause DOS attacks.

Working of Hping:

For a SYN Flood attack, it sends a huge number of TCP SYN packets to the target. Hence, leading to make resources by creating a half-open connections. TCP Handshake works as: SYN – Initiates a connection, SYN-ACK – Server responds to the request. ACK – Client acknowledges, completing the handshake. In SYN Flood

attack only SYN packets are sent, leaving the server waiting indefinitely for the final ACK, leading to resource exhaustion.

Steps for illustrating functionality of hping:

1. Setting up: Install hping – use sudo apt-get install hping3;
2. Crafting packets:
    a. Protocols: hping allows you to customize various aspects of network packets – TCP: -s for SYN, -A for ACK; UDP --udp.
    b. Source/Destination IPs and Ports: -a <spoofed_ip> : Spoof the source ip, -p <port> : Set destination port.
    c. Flags: SYN, ACK, FIN, RST, PSH (used for TCP handshakes and teardown)
    d. Payload: Attach data or customize packet size with -d
    e. Rate control: Limit packet sending speed using --flood or -i <interval>
3. Ping and Trace Routes: Simulate high traffic to analyse

Parameters used for hping:

sudo hping3 -S -p 22 --flood 192.168.118.169 – This is the command used for hping request

-S → Specifies sending TCP SYN packets

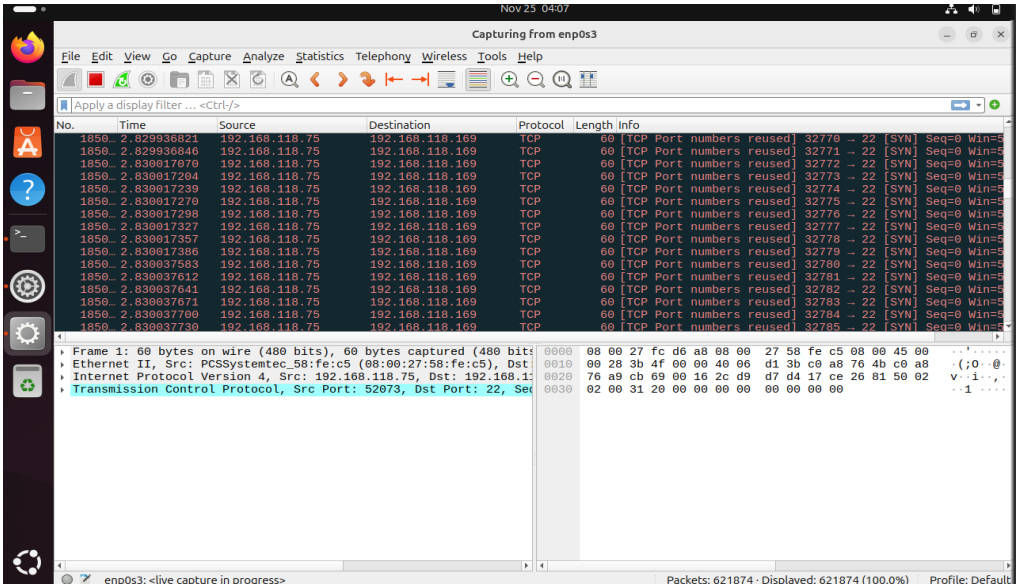
-p 22 → Target port 22 on the victim

--flood → Sends packets as fast as possible without waiting for replies

192.168.118.169 → IP Address of the target Machine (VM1)

d)

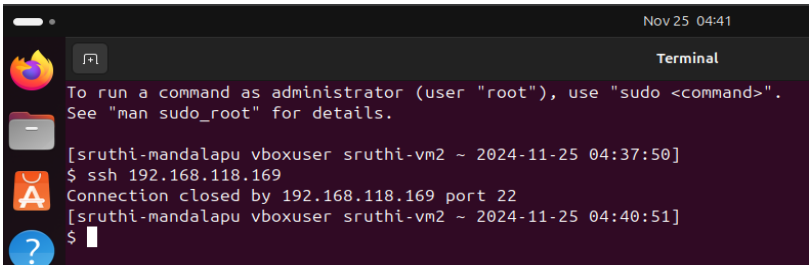
VM3 ip is 192.168.118.75, VM1 ip is 192.168.118.169 → we can observe continuous syn flood in VM1, and we are receiving from VM3. source ip - VM3, destination ip – VM1



e)

While attempting to establish an SSH session from VM2 to VM1, during ongoing SYN Flood attack, I have observed a delay and taking time waiting to connect, after sometime it couldn't able to connect finally shown up connection closed. Here, what happened is - SSH Session failed to connect with VM1. This is because the SYN Flood attack overwhelms VM1's networking stack, consuming resources like CPU and memory. It couldn't be completed as VM1 is too busy in processing or rejecting the flood of incoming SYN packets, leading unable to handle VM2 request.

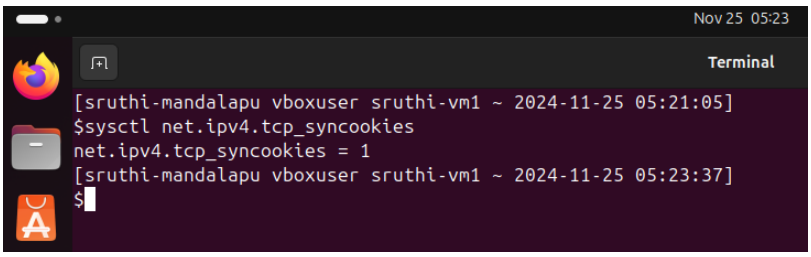
ssh 192.168.118.169 → the ip-address of VM1

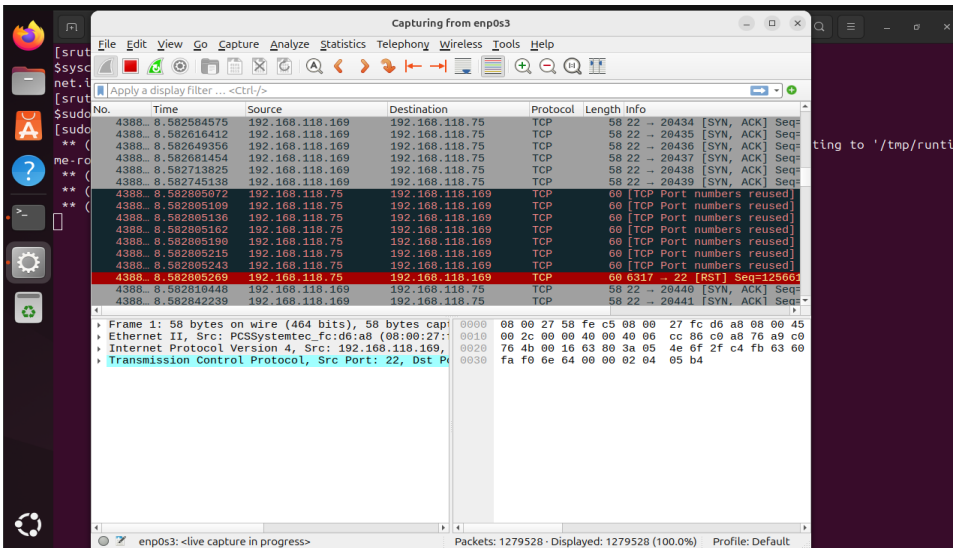


f)

In first scenario when cookies are disabled, VM1 allocated resources for every incoming SYN packet. The backlog quickly fills up under SYN flood causing system unable to work. We can see in wireshark, it shows a large number of connections stuck in SYN_RECV state. In other scenario, when cookies are enabled, VM1 doesn't allocate resources for incoming SYN packets until handshake is completed. We can see in wireshark, it shows a few connections remain in SYN_RECV state.
With cookies disabled: High volume of SYN packets, delayed or missing SYN-ACK responses, SYN-ACK packets often go unanswered.
With cookies enabled: High volume of SYN packets but still visible, SYN-ACK responses are sent promptly for all packets, ACK responses are processed allowing connections to complete.

When you see the screenshots in 1st scenario – most of the packets are SYN packets. In 2nd scenario – every SYN packet is completing a handshake protocol by producing acknowledgement responses as SYN-ACK.



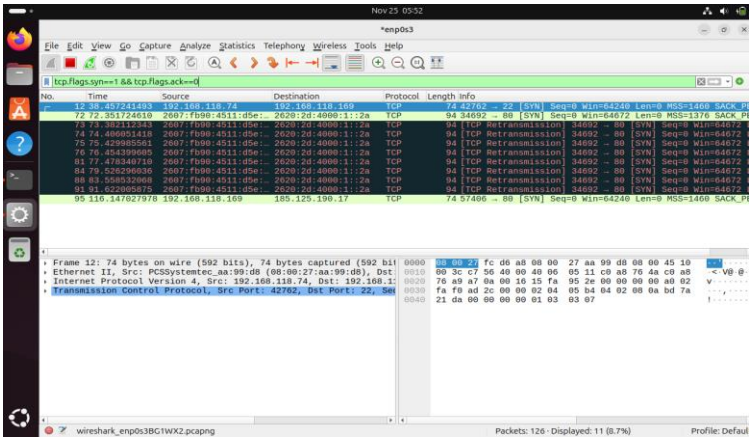


PART4: Traffic Analysis

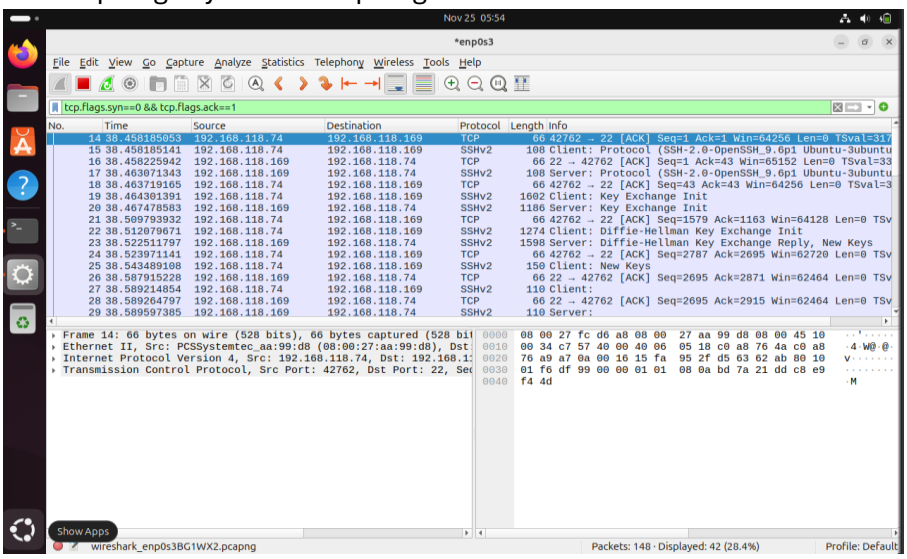
a)

Before SYN Flood:

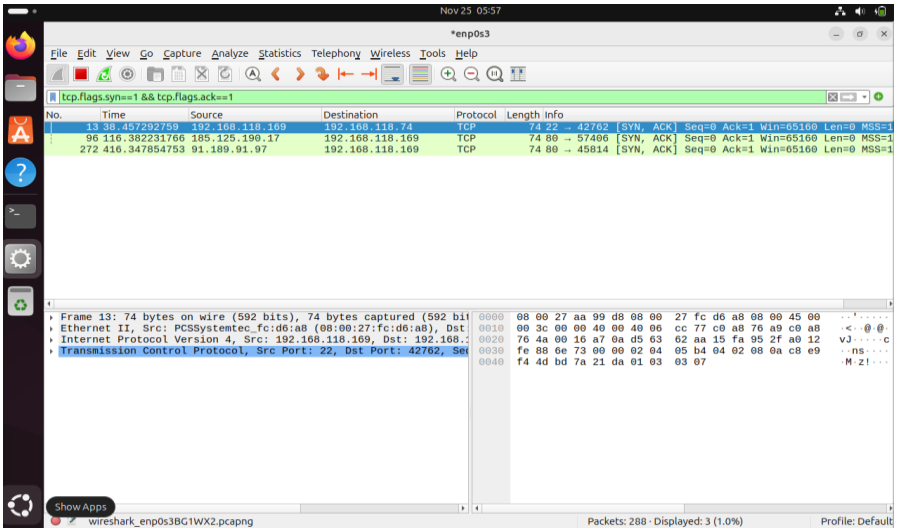
SYN: tcp.flags.syn==1 && tcp.flags.ack==0



ACK: tcp.flags.syn==0 && tcp.flags.ack==1



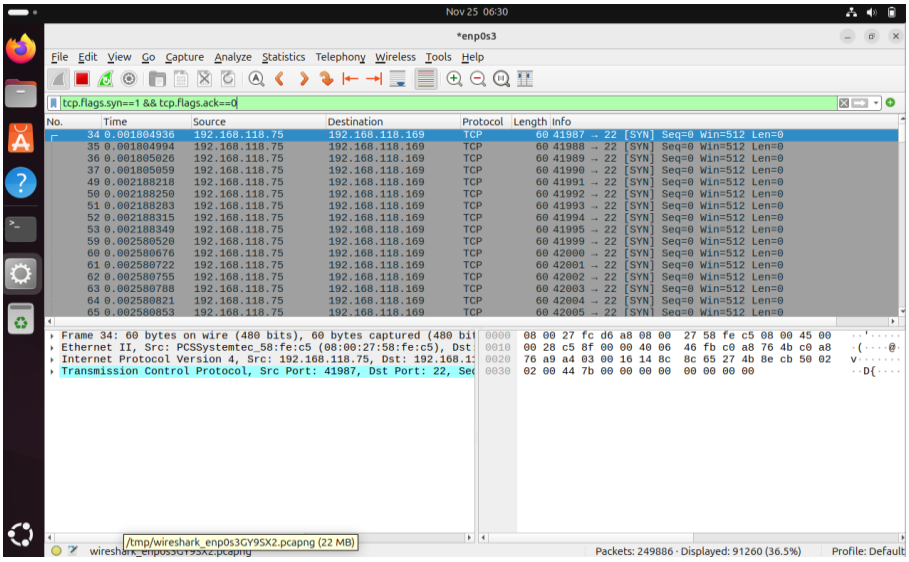
SYN-ACK: tcp.flags.syn==1 && tcp.flags.ack==1



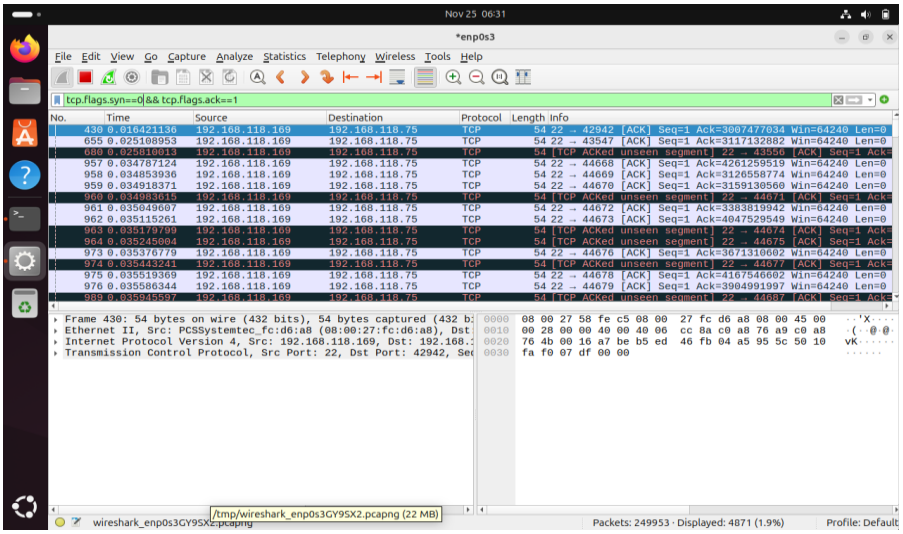
SYN Flood when SYN Cookies enabled:

By default, the cookies are enabled

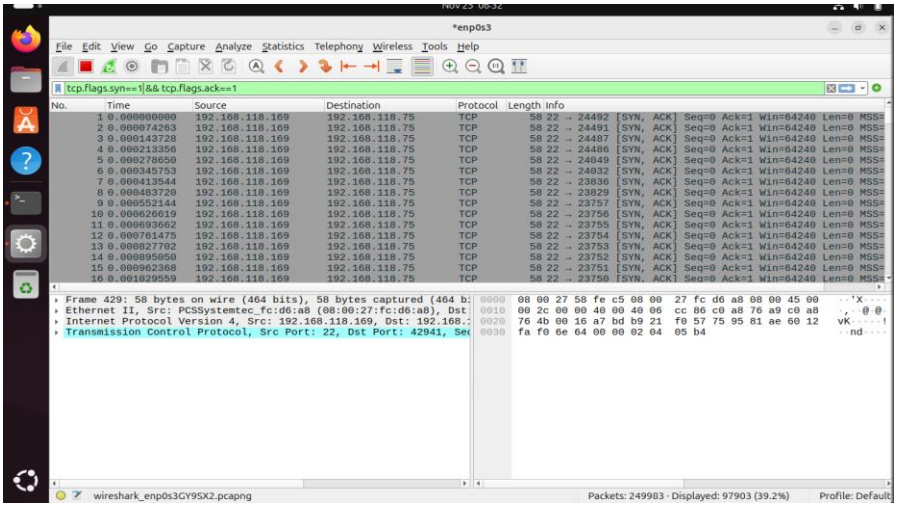
SYN: tcp.flags.syn==1 && tcp.flags.ack==0



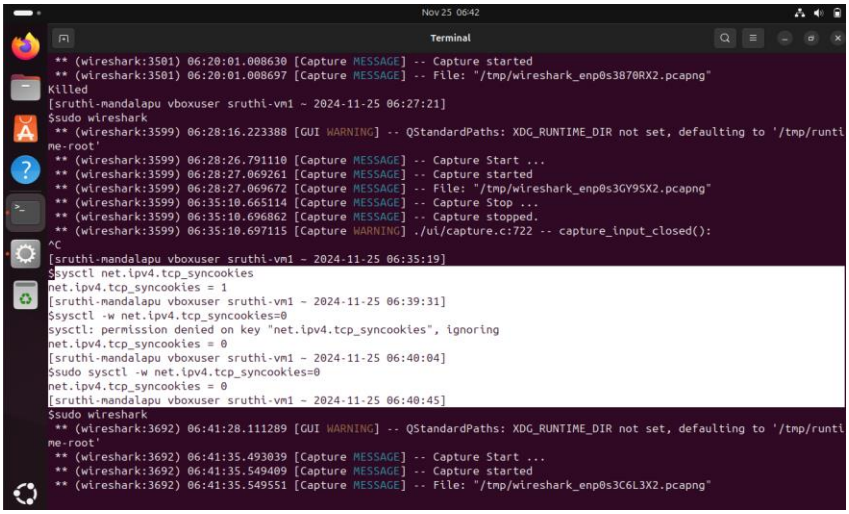
ACK: tcp.flags.syn==0 && tcp.flags.ack==1



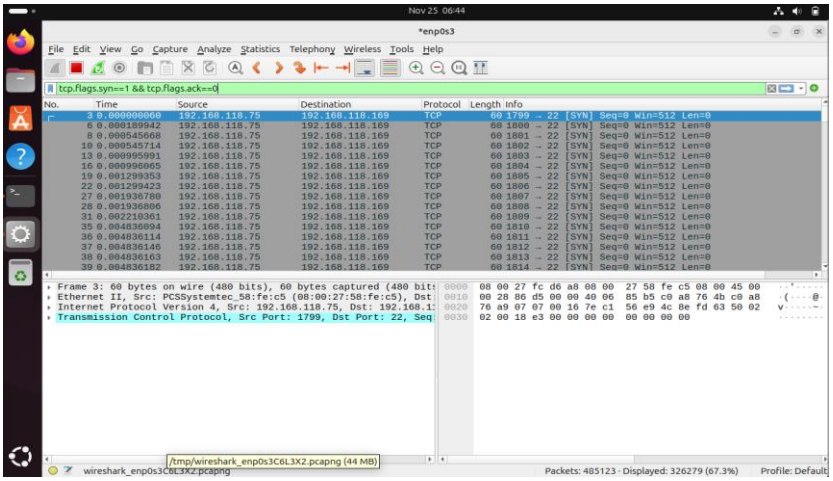
SYN-ACK: tcp.flags.syn==1 && tcp.flags.ack==1



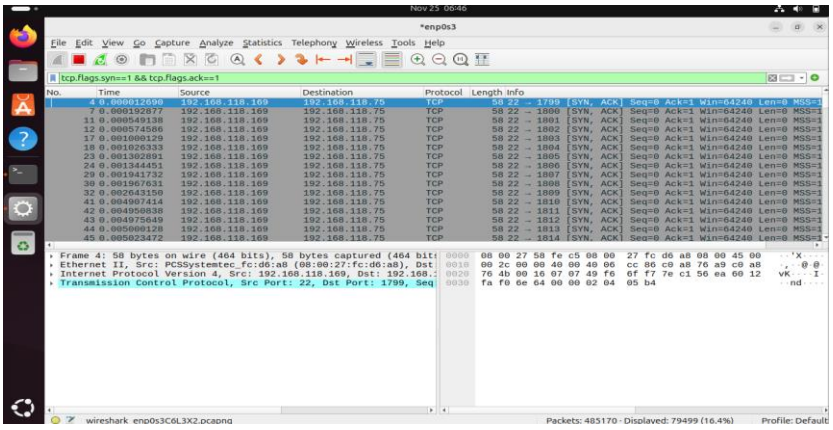
SYN Flood when SYN Cookies disabled:



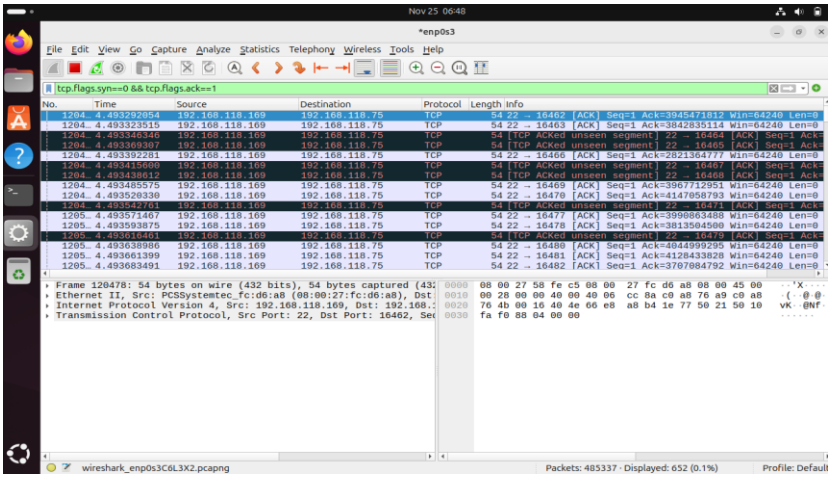
SYN: tcp.flags.syn==1 && tcp.flags.ack==0



SYN-ACK: tcp.flags.syn==1 && tcp.flags.ack==1



ACK: tcp.flags.syn==0 && tcp.flags.ack==1



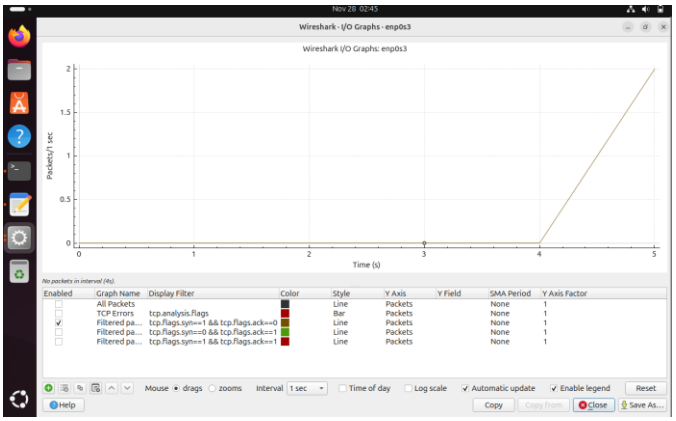
As you can see in the below table – Displayed percentage in wireshark at bottom – among all the three scenarios:

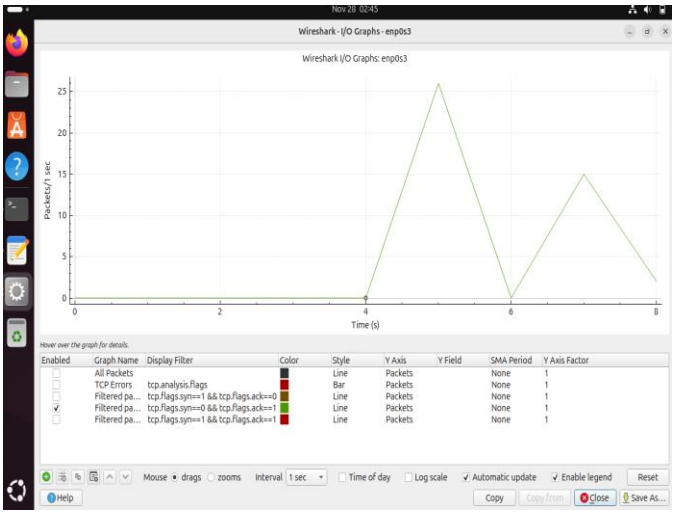
	Before Flood Attack	Cookies Enabled - Flood	Cookies Disabled - Flood
SYN	8.7%	36.25%	67.3%
SYN-ACK	1%	39.2%	16.4%
ACK	28.4%	1.9%	0.1%

As you can see before flood attack, almost equal proportions for syn, syn-ack, ack. As it is not a flood attack you could only observe 8 percent as SYN packets. In case of Cookies enabled and flood attack. As this is flood attack – the SYN is increased from 8 percent to 36.25 percent. Until the acknowledgement is received, the vm waits to finish up each SYN packet. Hence, SYN and SYN-ACK almost have equal proportions (each packet waits to finish handshake protocol). When cookies is disabled, the SYN packet keeps on generating – hence there is increase in SYN packets received than before – it is from 36% to 67%. And it couldn't able to give the responses hence there is low SYN-ACK (which is 16% reduced than before).

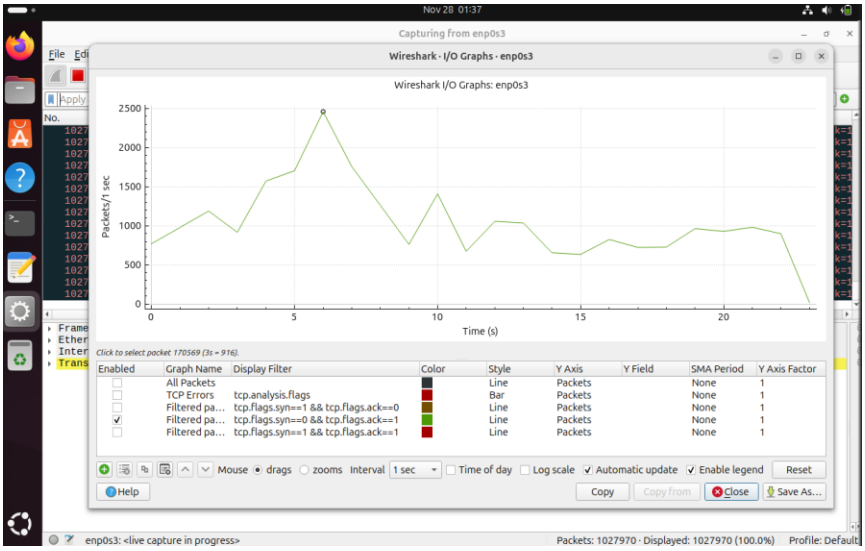
b)

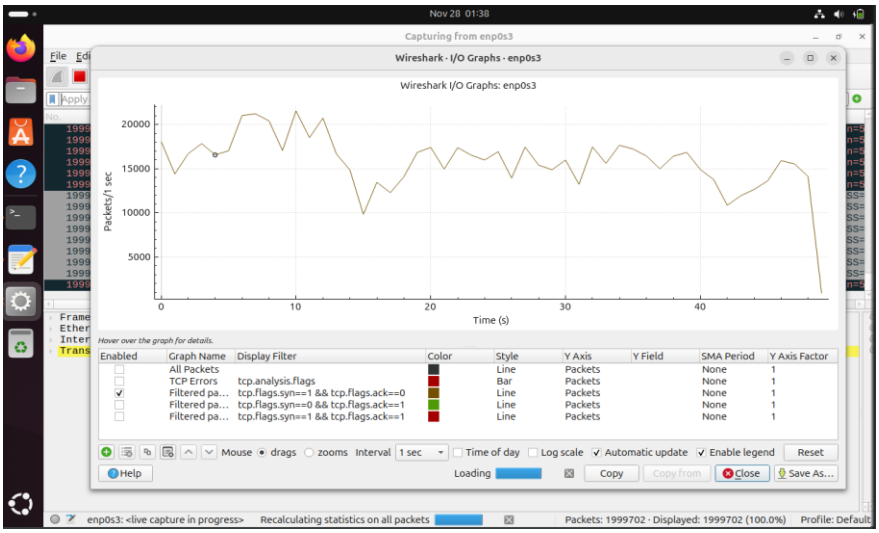
Before Flood Attack: (Made ssh connection from vm2 to vm1)

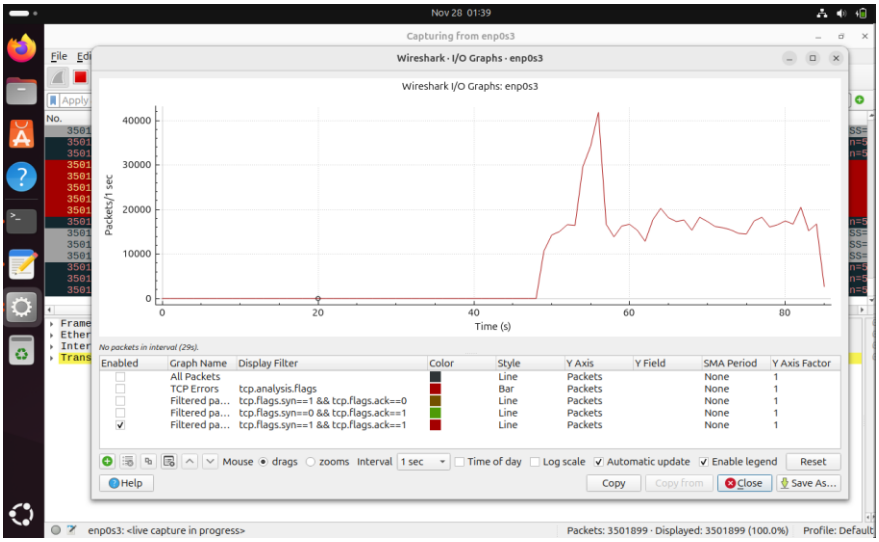




During Flood Attack when cookies is enabled:

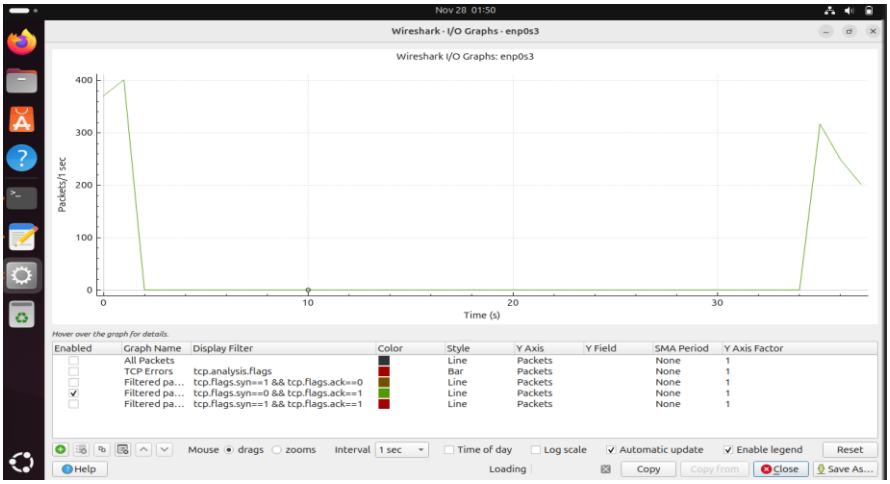


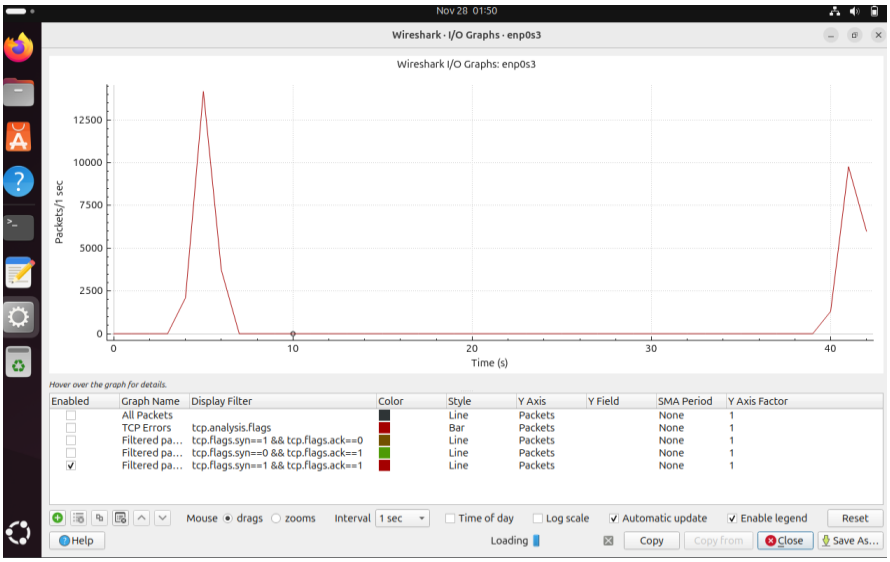




When the cookies are enabled – we will finish handshake protocol for every packet, rather than allocating all the packets that was sent from the attacker. So, we can observe the SYN, SYN-ACK and ACK packets capturing over times. Firstly, SYN is sent then SYN-ACK is sent. As it is a flood attack – attacker focuses on sending the packets, hence there are no ACK first few seconds, after sometime we could find ACK packets.

During flood attack when cookies is disabled:





When cookies are enabled, the main focus is on transmitting the packets, it allocates all the packets firstly rather than completing the connections. Which explains leaving too many open connections. Hence, we find high data for SYN packets. Coming to SYN-ACK, as it focuses completely on allocating all the receiving packets, hence we have less data rate for SYN-ACK. And almost very much less packets that is almost negligible packets for ACK.

Data Rate:

Data Rate = Total number of packets transmitted/Time (in seconds)

- Data Rate for before attack: (for first 10 seconds)
  Data Rate = (0+0+0+0+0+2+0+0+0+0)/10 = 0.2 packets/sec
- During Attack, when cookies is enabled: (for first 10 seconds)
  Data Rate = (750+1000+1200+1500+1000+1700+2500+2200+1700+750+1400)/10 = 1500 packets/sec (approximation – as a rough values from graph)
- During Attack, when cookies is disabled: (for first 5 seconds)
  Data Rate = (0+0+0+0+60000+65000+65000+70000+60000+60000+65000)/10 = 44000 packets/sec (approximation – as a rough values from the graph)

Data Rates for incoming packets to VM1 – this explains about the SYN packets. As you can see from above data rates, before attack the data rate is too low, when the attack but when cookies is enabled the data rate is little higher than before attack scenario. Lastly, when we check when cookies are disabled it was too high, projecting maximal data rates comparison to the above 2 conditions.

c)

As you can see in the below table – among all the three scenarios:
Table taken from Part (a) – figures (at the bottom – there is a packet count and filter count and the percentage)

	Before Flood Attack	Cookies Enabled - Flood	Cookies Disabled - Flood
Total no of packets	200 (Round figure)	2,50,000 (Round figure)	5,00,000 (Round Figure)
SYN Packets	11 (8.7%)	92,000 (36.25%)	3,30,000 (67.3%)
SYN-ACK Packets	3 (1%)	97,000 (39.2%)	80,000 (16.4%)

The above table explains there is a baseline data (before flood attack), during flood attack – cookies enabled, during flood attack – cookies disabled. The table data explains about count of packets and the percentage. SYN packets explain about SYN_SENT, SYN-ACK packets explain about SYN_RECEIVED. Data is displayed based on before attack, attack on cookies enabled, attack on cookies disabled at specified instance that is (SYN, SYN-ACK) is displayed. We can find the number packets at each state (that is SYN/SYN-ACK) and also the percentage. We can observe clearly before attack, followed by cookies enabled, followed by cookies disabled the total number of packets is strictly increasing. When SYN packets – even in this case the number of packets are strictly increasing. When SYN-ACK packets when its disabled (this state gives slight lower when compared to the state of enabled). Overall, the attack when cookies are disabled affects the most receiving the highest number of packets.