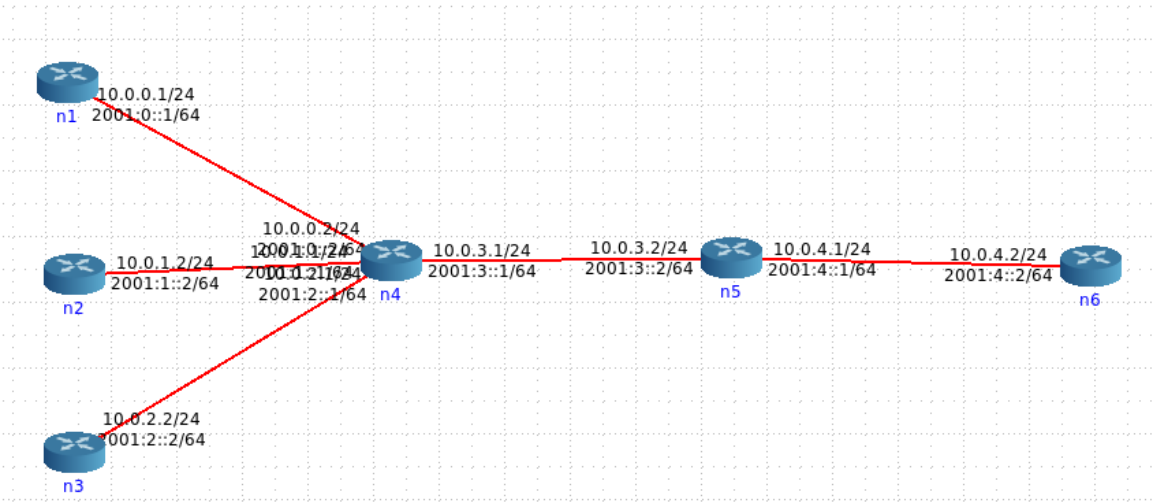
Network Security – CSCI_6541_80

Namana Y Tarikere – G21372717

Homework Assignment – 7

Firewall Section (10pts)



Useful commands:

- Flush the filter table: iptables -F
- Flush the NAT table: iptables -F -t NAT
- List the rules in the filter table: iptables -L
- List the rules in the filter table with #packet, #bytes matching each rule: iptables -L -n -v
- Change the default policy of the INPUT chain of the filter table to DROP: iptables -P INPUT DROP

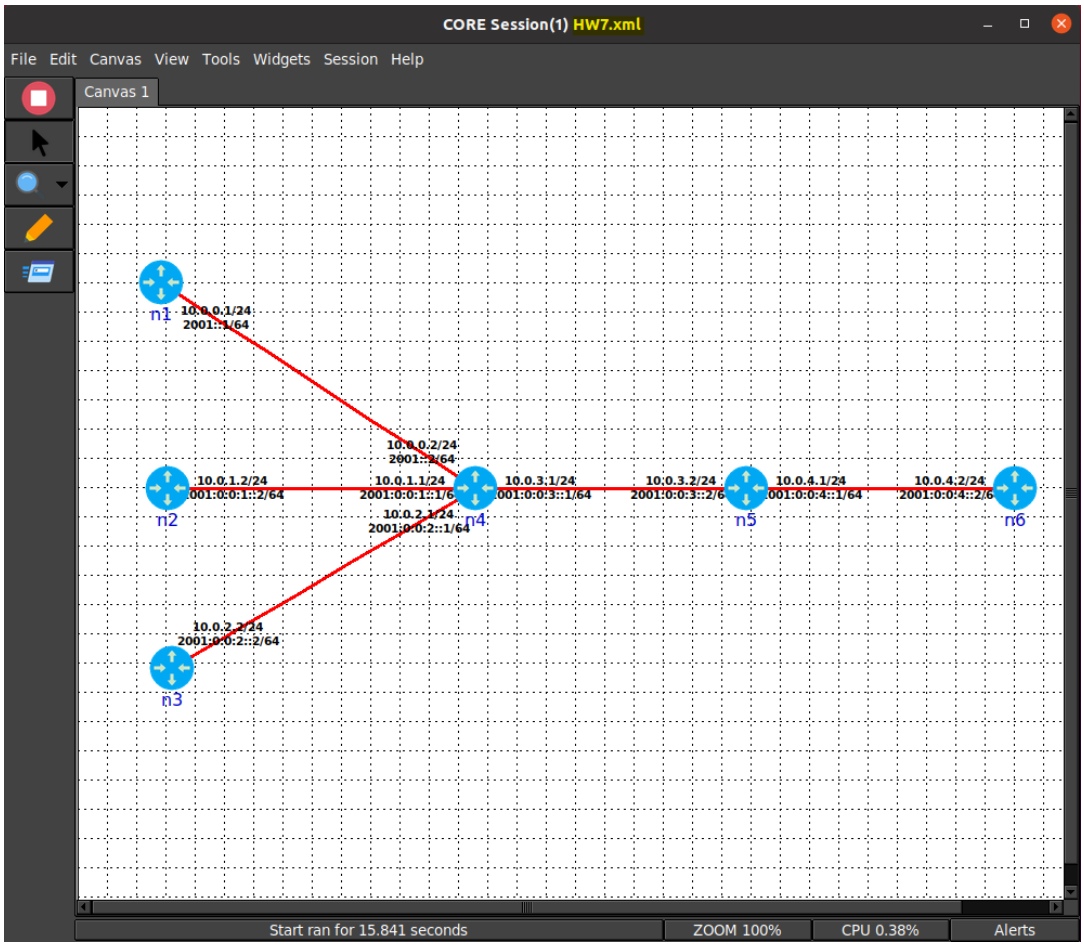
Use IPTables (http://www.netfilter.org/projects/iptables/index.html) to apply the following rules **at n4** to the network below.

For each question, provide a list of firewall rules to apply **at n4**. Associate rules with each of the requirements above. Please provide screenshots and make each expand to the width of the page.

Example:

1. a. iptables –A …. etc.

Core GUI:

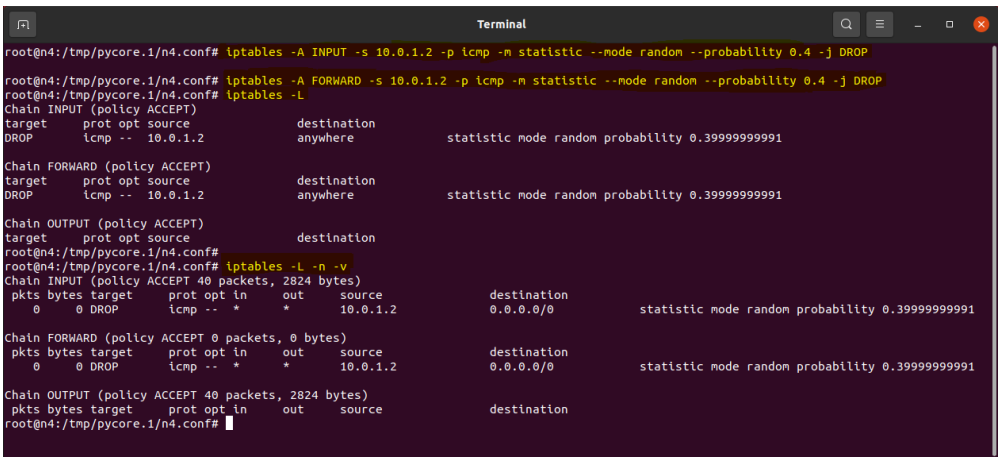


1) (2.5pts) Drop 40% of all ICMP packets from n2

a. Show the iptables command you used?

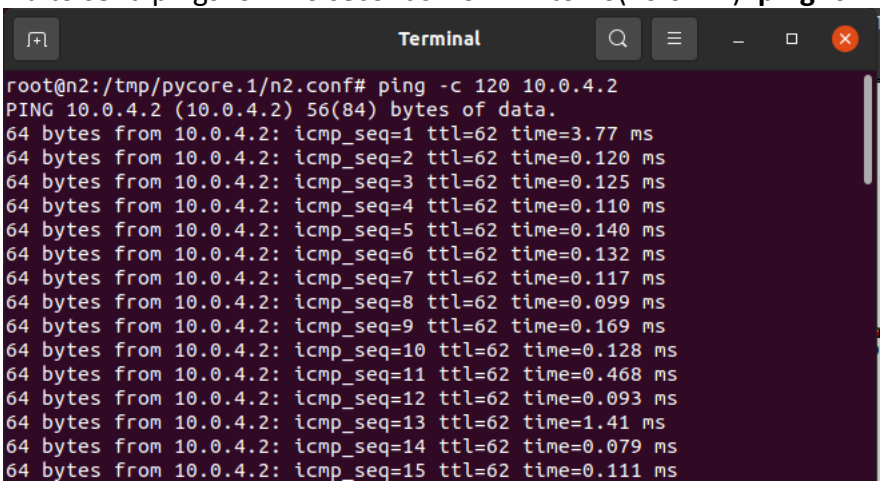
Command for dropping input and forward packets coming to n4 from n2:

- **iptables -A INPUT -s 10.0.1.2 -p icmp -m statistic --mode random --probability 0.4 -j DROP**
- **iptables -A FORWARD -s 10.0.1.2 -p icmp -m statistic --mode random - probability 0.4 -j DROP**
- **iptables -L**



b. Use ping command to ping from n2 to n6 for 120 seconds. Show the following:

Command to send pings for 120 seconds from n2 to n6(10.0.4.2): **ping -c 120 10.0.4.2**

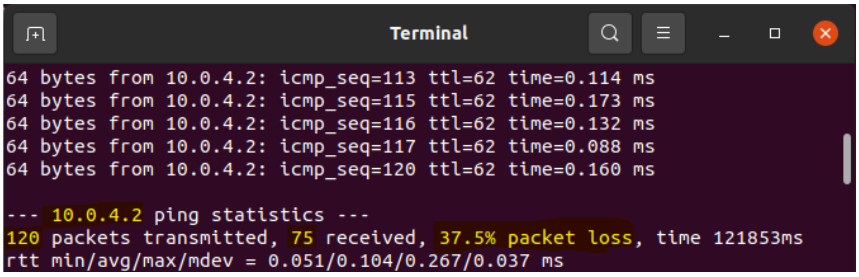


i. Show iptable counts to see # of packet drops corresponding to this rule.

1. You can use the command "iptables -L -n -v"

In the above screenshot, the output of **iptables -L -n -V** command showed **45** packets loss which matches with the forward drop rule as shown in the below ping statistics section.

ii. Show ping statistics reflecting the loss (lost packets should roughly match matched packets count in i.1)



The ping statistics show that only 75 packets were successfully received out of 120 packets transmitted. This results in a loss of **45** packets (120 - 75 = 45), leading to a **37.5%** packet loss. This closely aligns with the ~40% packet drop rate implemented by the iptables rule, as verified by the output of the command **iptables -L -n -v** as mentioned above.

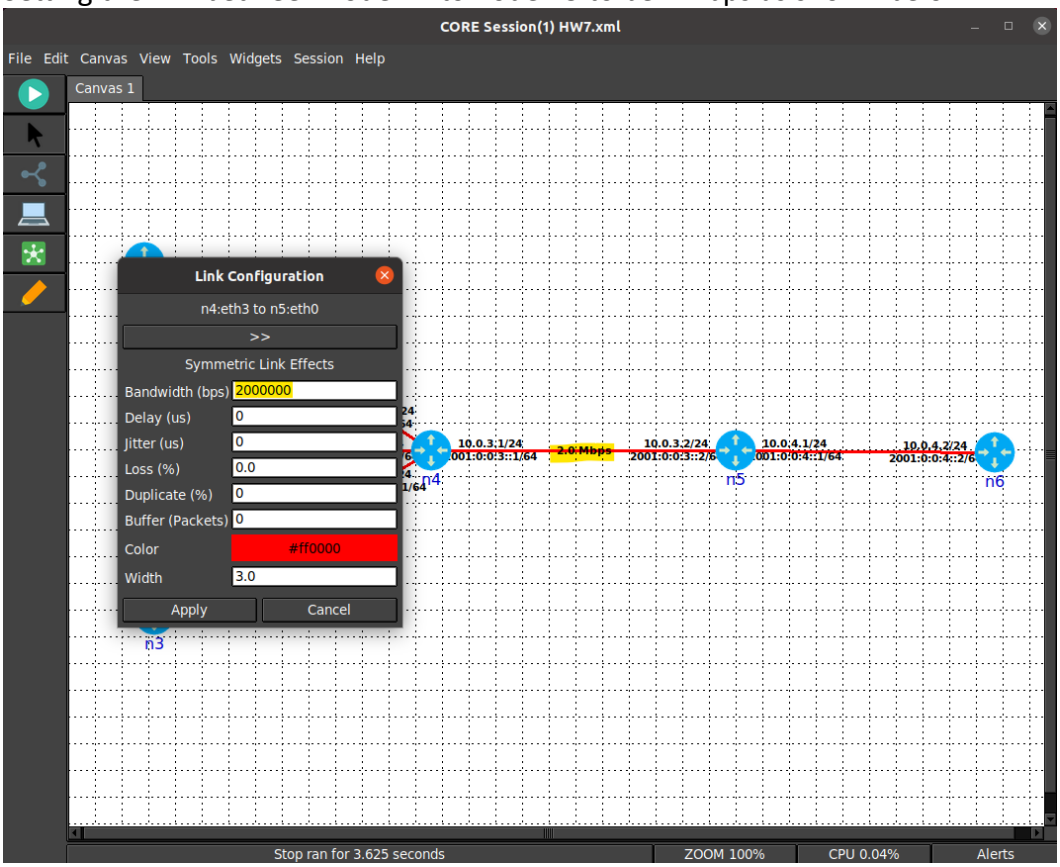
c. Reset iptables by flushing the rules as indicated in the "useful commands" section

Command to flush the iptables: **iptables -F**

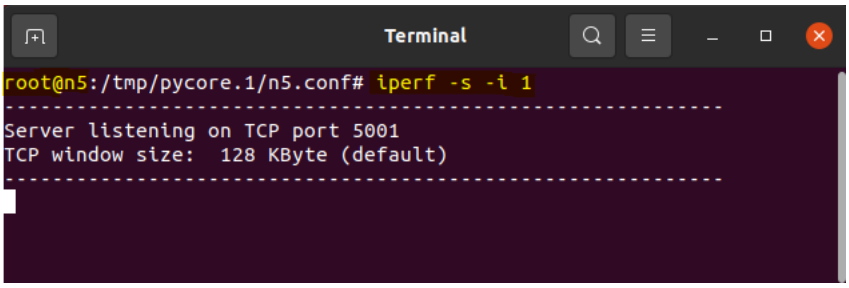
2. (2.5pts) Imagine n5 is a server that you are running and that n1, n2, and n3 are clients. Node n1 is an attacker who is using hping3 to run a UDP flooding attack against you. Nodes n2 and n3 are good nodes

a. Set the link between n4 and n5 to be 2Mbps.

Setting the link between node n4 to node n5 to be 2Mbps as shown below:

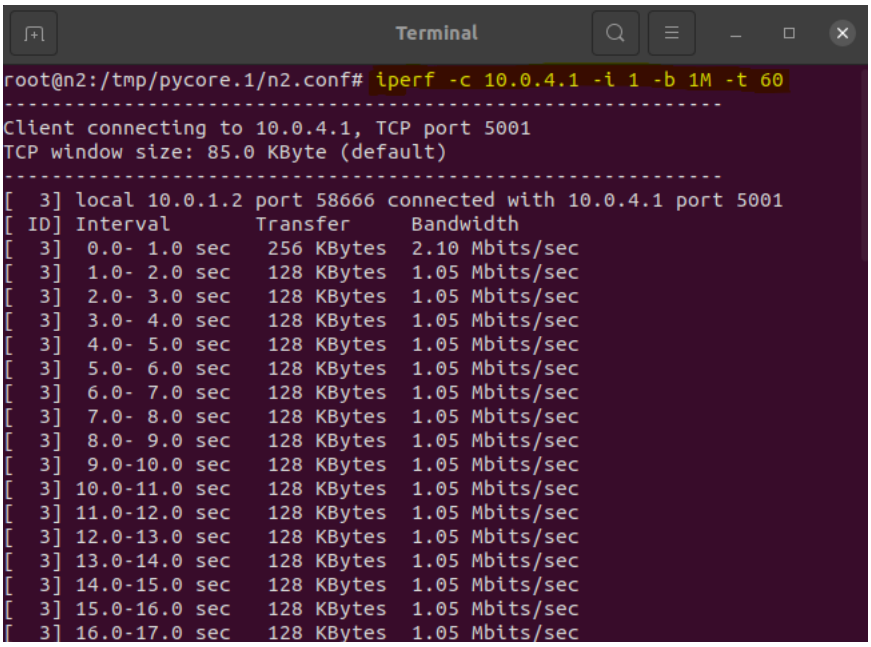


b. Run a TCP iperf server on node n5: iperf -s -i 1

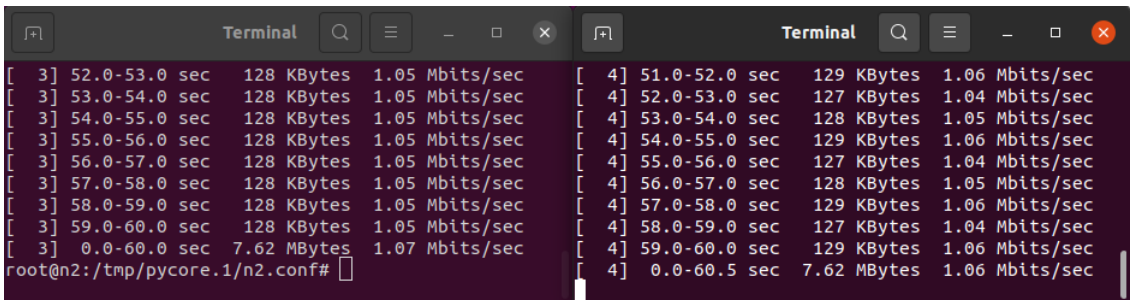


c. Try to run an iperf TCP client from node n2 to node n5. Set client to send 1mbps of data (-b option) for 1 minute (-t option).

i) iperf -c 10.0.4.1 -i 1 -b 1M -t 60



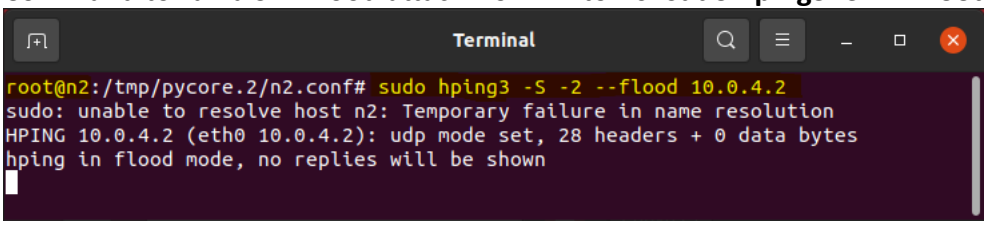
d. What performance does iperf report?



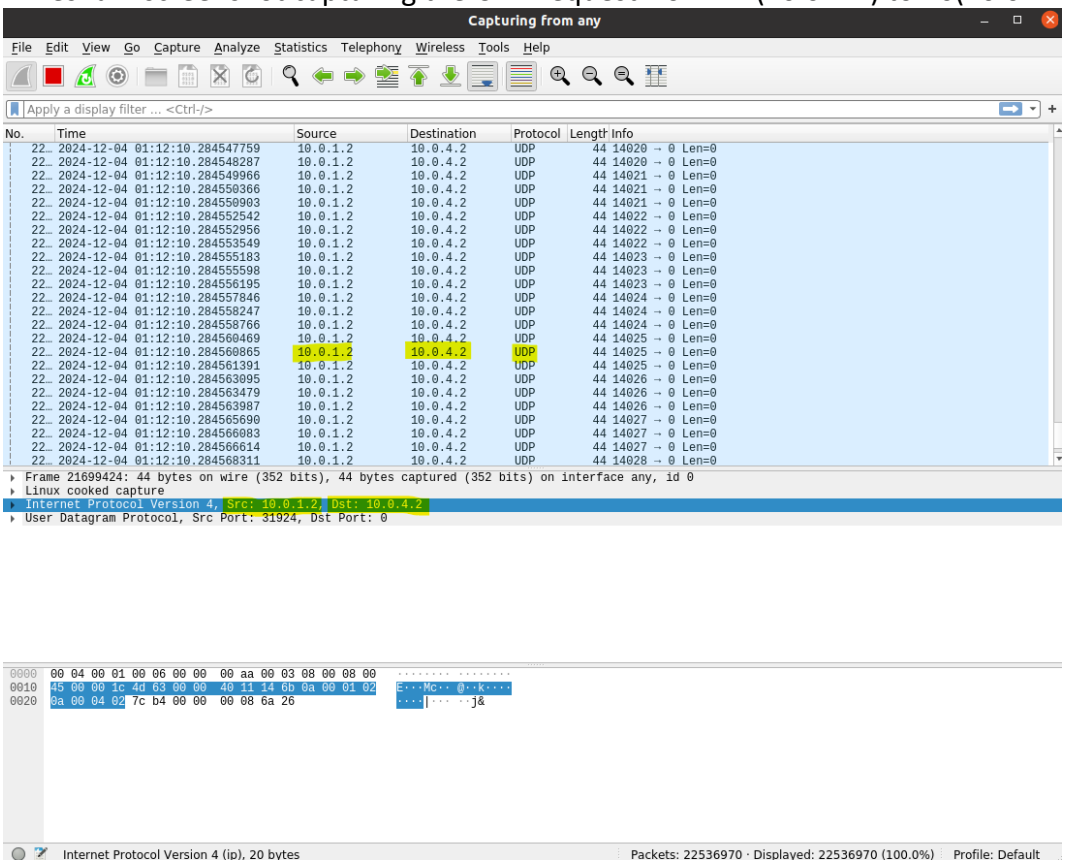
The iperf client reported a performance of 128 Kbytes of data transfers on a 1.05 Mbits/sec bandwidth.

e. Now, use hping3 to run a SYN flood attack from node n2 to node n6

Command to run a SYN flood attack from n2 to n6: **sudo hping3 -S -2 --flood 10.0.4.2**

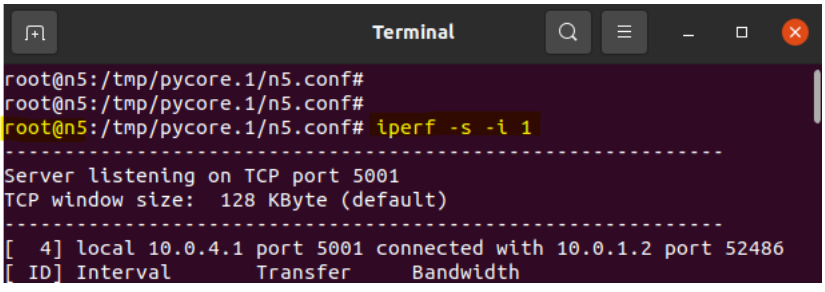


Wireshark screenshot capturing the UDP request from n2 (10.0.1.2) to n6(10.0.4.2):

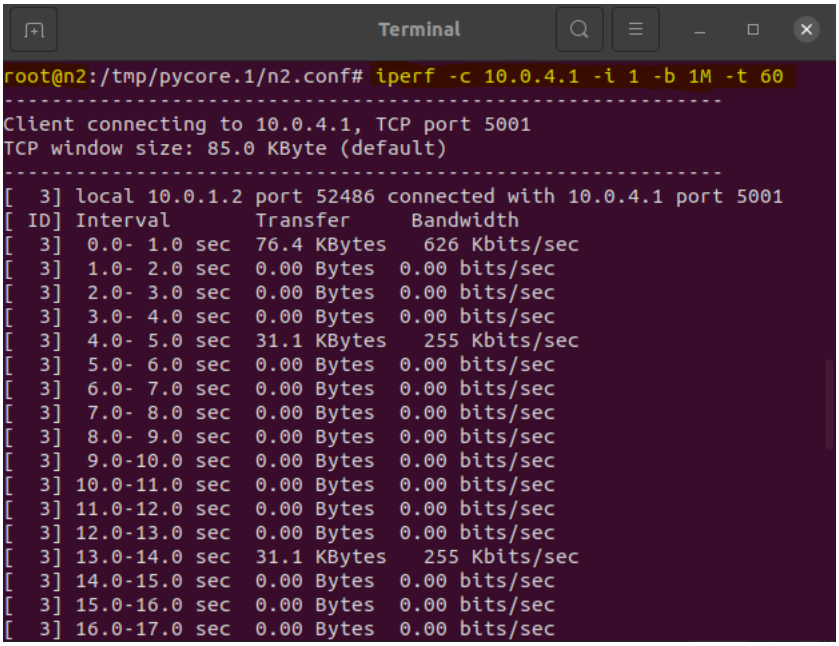


f. Repeat c, d

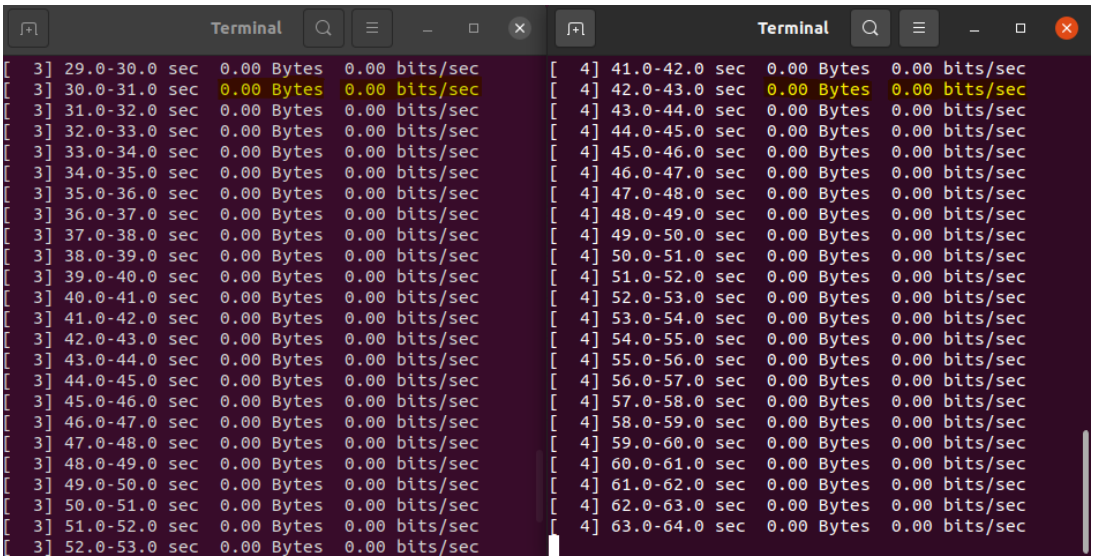
- Starting the iperf server on n5 using the command: **iperf -s -I 1**



- Starting iperf client on n2 while running a SYN flood attack using the command: **iperf -c 10.0.4.1 -I 1 -b 1M -t 60**



- In the below screenshot the **[3] is representing iperf client on n2** and **[4] is representing iperf server on n5**. We can observe that as the SYN flood attack is in progress, the iperf client could not transfer information to the server.

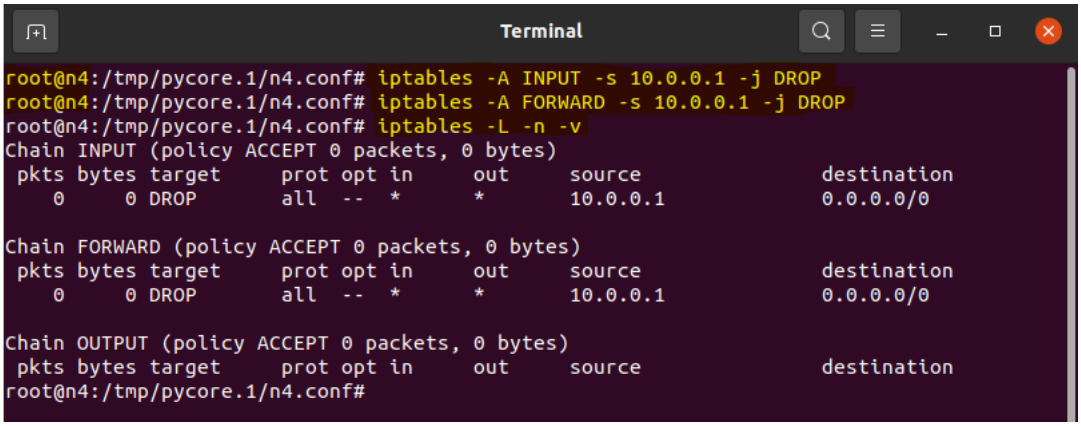


The iperf client reported a performance of 0 Bytes of data transfers on a 0.00 bits/sec bandwidth as shown above.

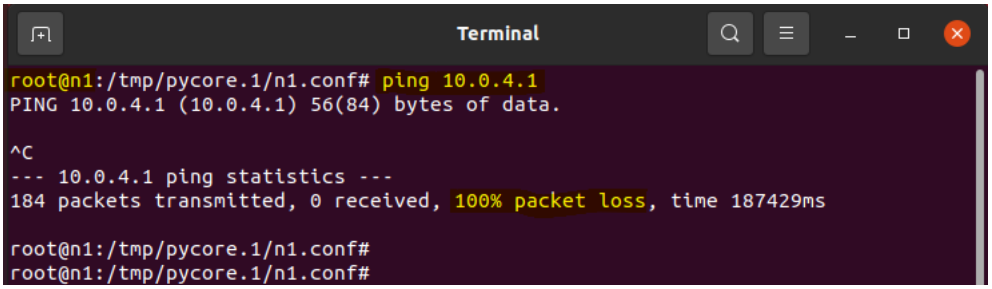
g. Now add a firewall rule at node n4 to drop all packets coming in from n1. What rule did you add?

Command to add a firewall at node n4 to drop all input and forward packets coming in from n2 and view the output of iptables:

- **iptables -A INPUT -s 10.0.0.1 -j DROP**
- **iptables -A FORWARD -s 10.0.0.1 -j DROP**
- **iptables -L -n -v**



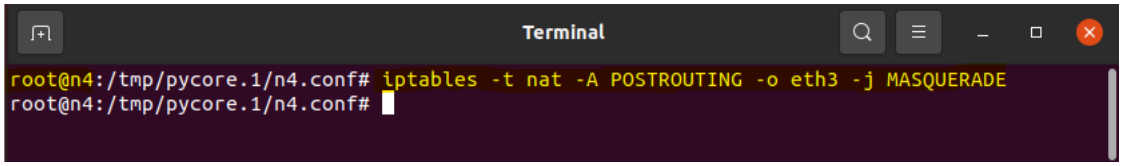
The ping statistics from n1 to n4 reveals a 100% loss of packets since all packets from n1 were dropped at the firewall at n4 as shown below:



3. (2.5pts) Perform a NAT for all traffic forwarded by n4 to n5

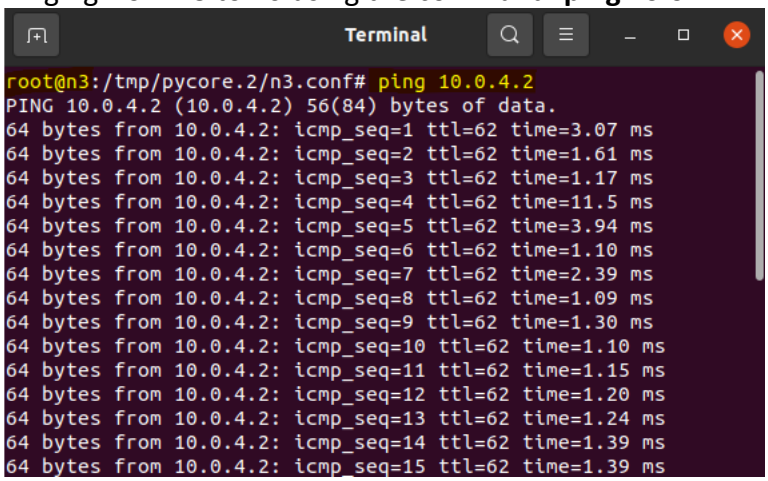
a. Show the iptables command you used?

Command used: **iptables -t nat -A POSTROUTING -o eth3 -j MASQUERADE**



b. Use ping command to ping from n3 to n6. Show the following:

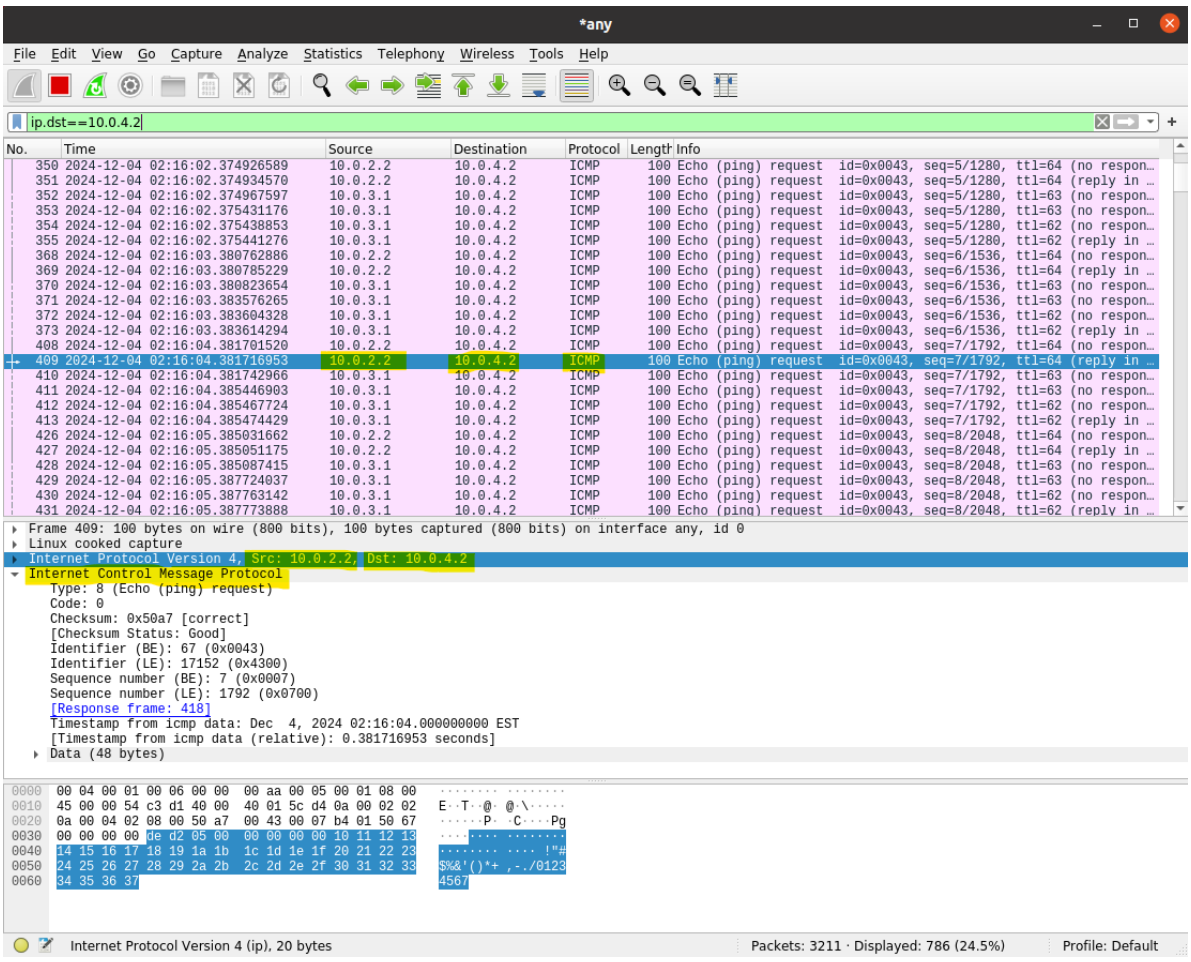
Pinging from n3 to n6 using the command: **ping 10.0.4.2**



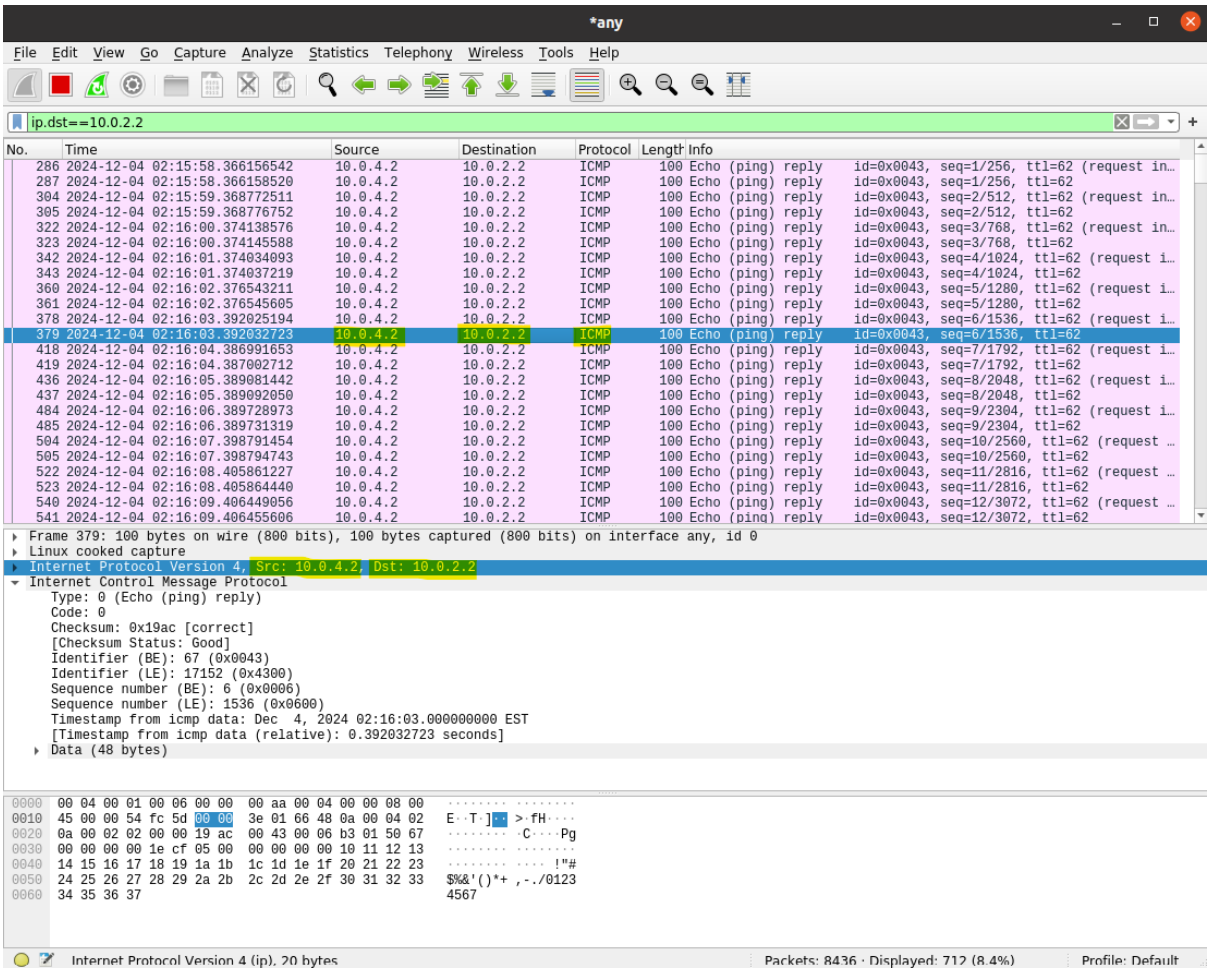
i. Screenshot of ICMP packets arriving at node n4 (i.e., link between n3 and n4). Highlight the source and destination IP addresses

The source IP address is 10.0.2.2 and the destination IP address is 10.0.4.2

The Wireshark packet capture arriving at node 4 (the link between n3 and n4) is shown:



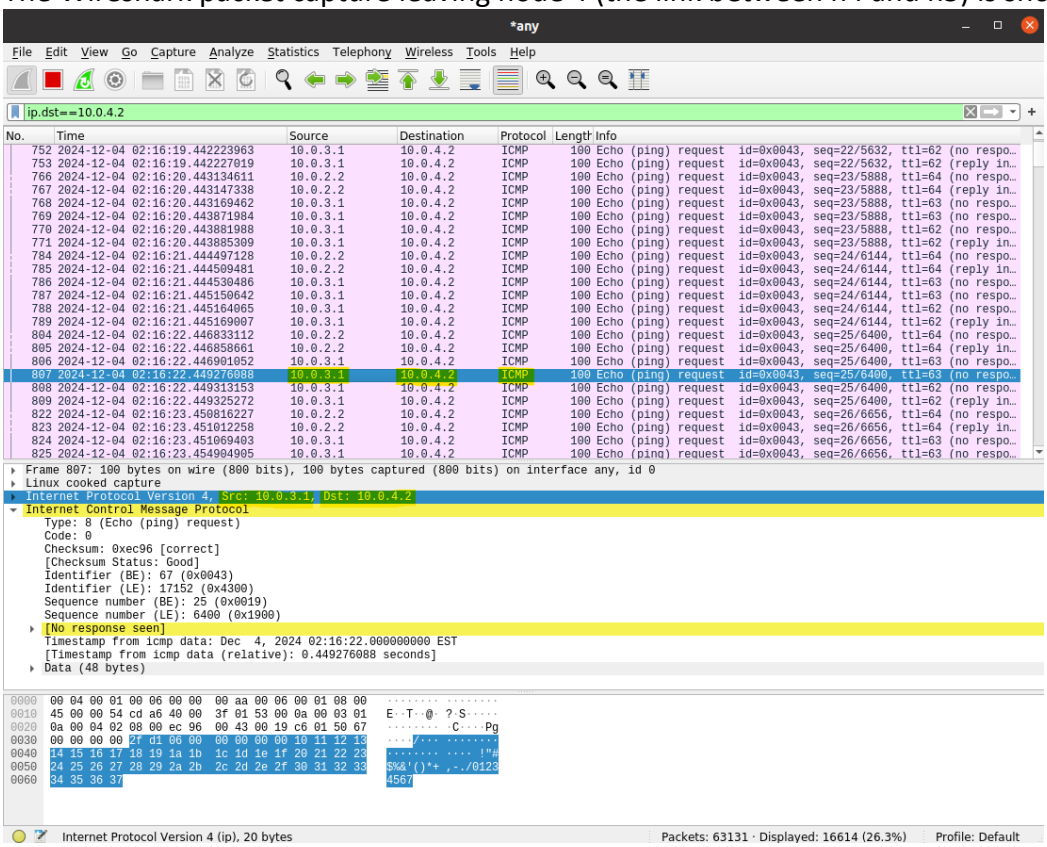
The Wireshark packet capture leaving node 4 (the link between n3 and n4) is shown:



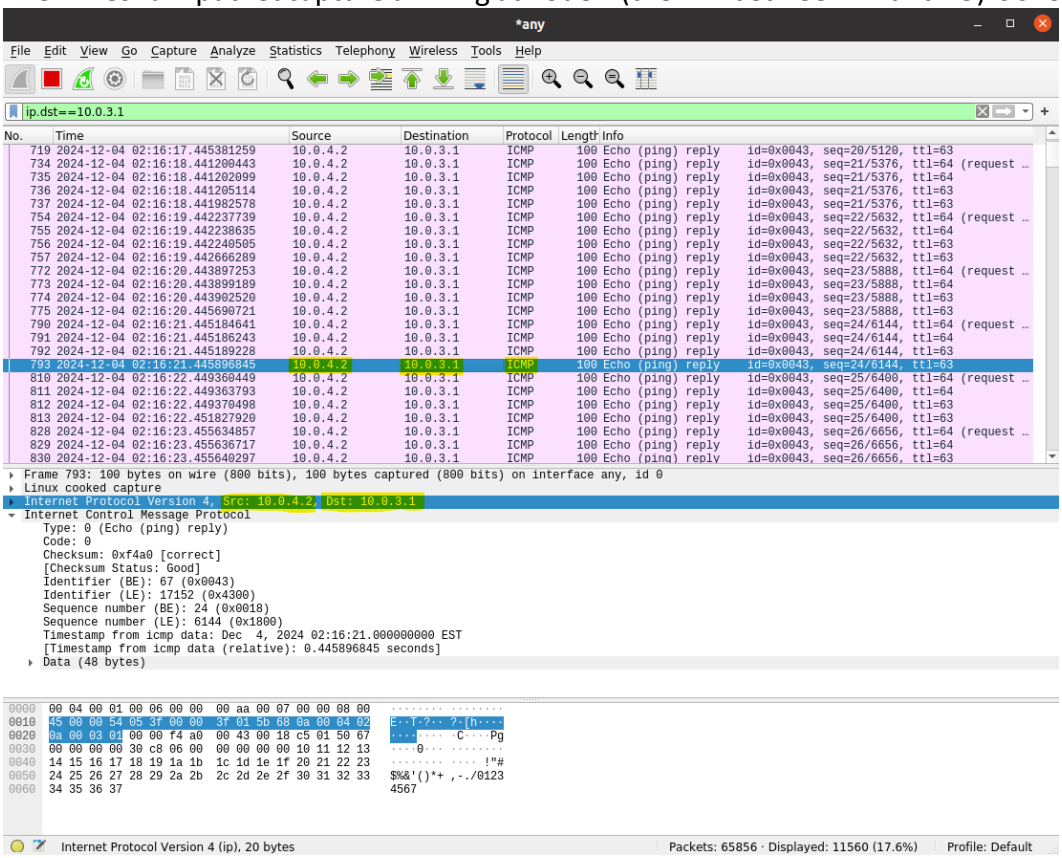
ii. Screenshot of ICMP packets leaving node n4 (i.e., link between n4 and n5). Highlight the source and destination IP addresses

The source IP address is 10.0.3.1 (Due to the masquerading rule)
The destination IP address is 10.0.4.2

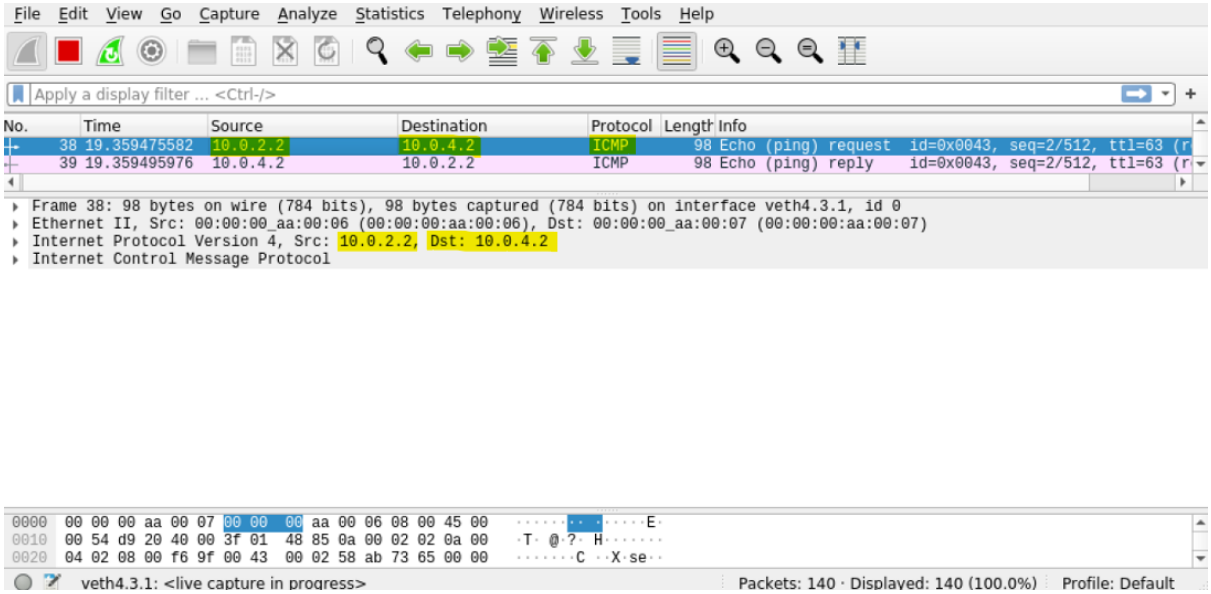
The Wireshark packet capture leaving node 4 (the link between n4 and n5) is shown:



The Wireshark packet capture arriving at node 4 (the link between n4 and n5) is shown:

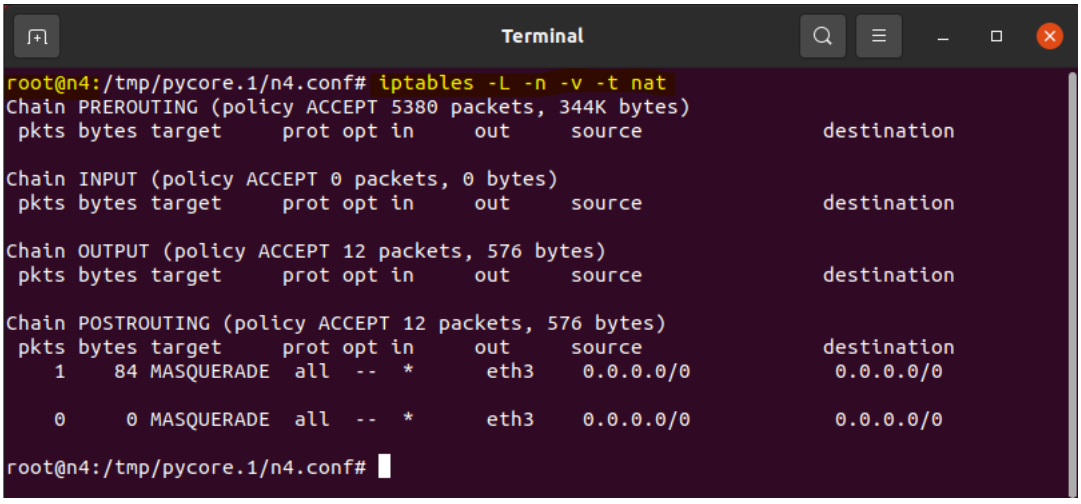


Without masquerading, the Wireshark packet capture at node n4 (on the link between n4 and n5) shows the source IP address as 10.0.2.2, indicating that the rule is working correctly as shown below:



iii. Show the output of iptables -L -n -v -t NAT

The output of the command **iptables -L -n -v -t nat** is shown below:

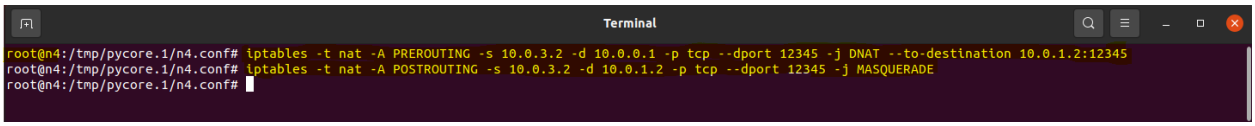


Here, the masquerade rule is being matched whenever messages are initiated from node n4 to n5.

4. (2.5pts) Imagine that nodes n1 and n2 are running the same server: nc -k -l 12345. Nodes n5 and n6 are both running "nc" clients and we want to load balance connections between n1 and n2.

a. **Assume nc clients on both n5 and n6 initially send traffic to node n1.**
b. **Use a combination of DNAT and NAT (like question 2) to change traffic from n5 so it goes to n2 instead. Ensure bidirectional traffic is possible.**

The commands used in combination of DNAT and NAT to change traffic and allow bidirectional traffic to go from n5 to n2 is shown below:



c. **Use any combination of screenshots to convince me that your solution works.**

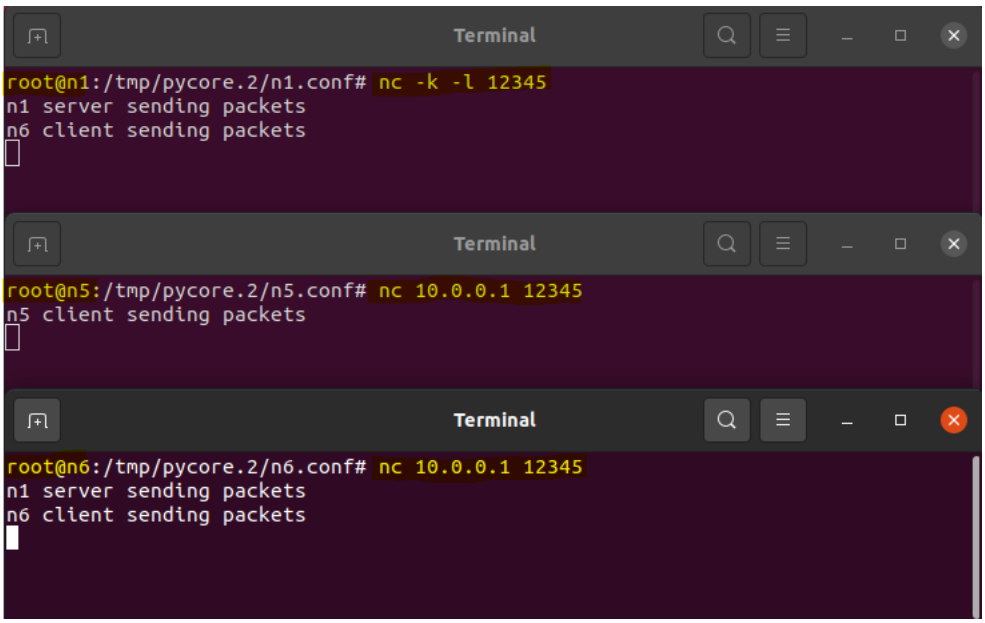
<u>Answers to a, b and c parts:</u>

<u>Before Load Balancing:</u>

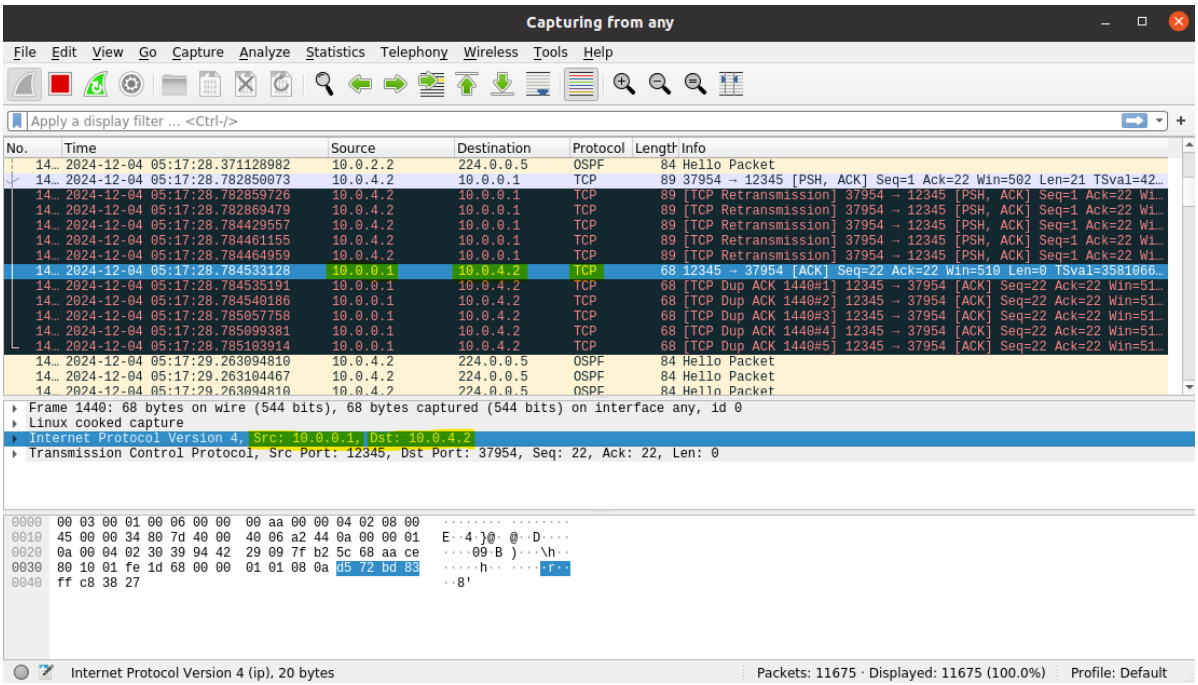
Commands used:

- At n1 server: **nc -k -l 12345**
- At n5 client: **nc 10.0.0.1 12345**
- At n6 client: **nc 10.0.0.1 12345**

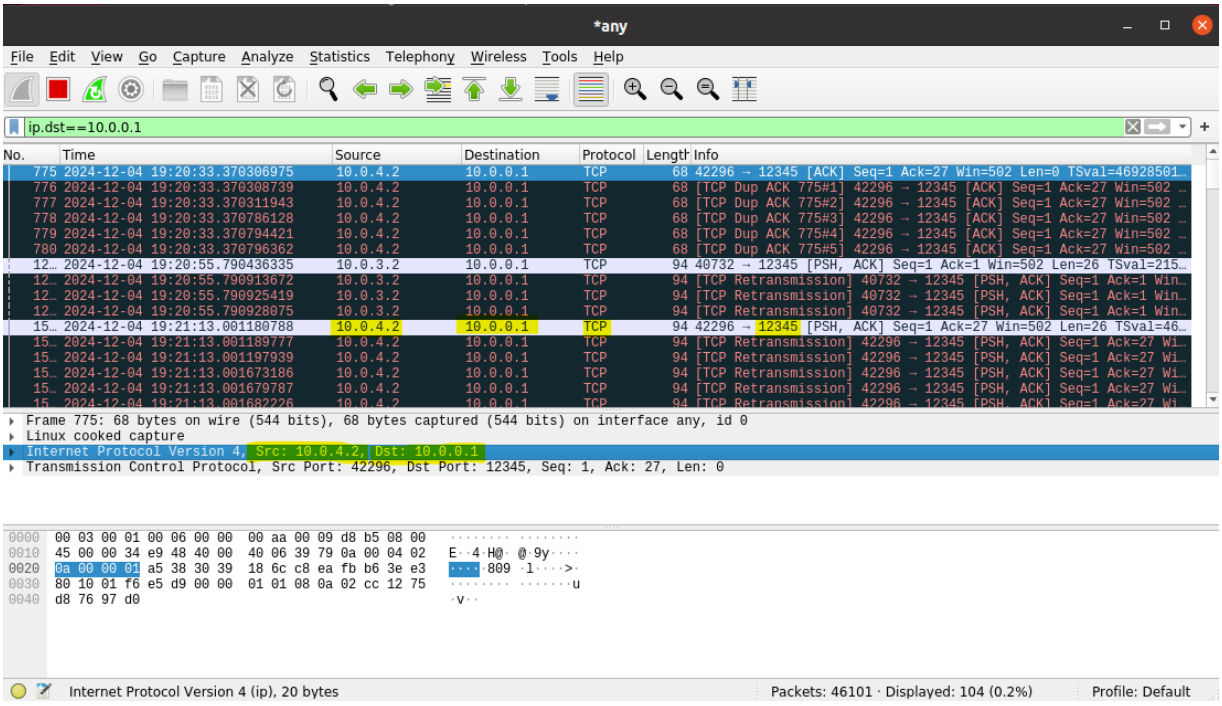
The n1 server can listen for multiple connections but can only maintain a single connection due to its single-threaded design. Consequently, it establishes and maintains a bidirectional connection with the client at node n6, as the command for client at node n6 was executed first, even though it receives packets from the client at node n5. The screenshots below confirm the same:



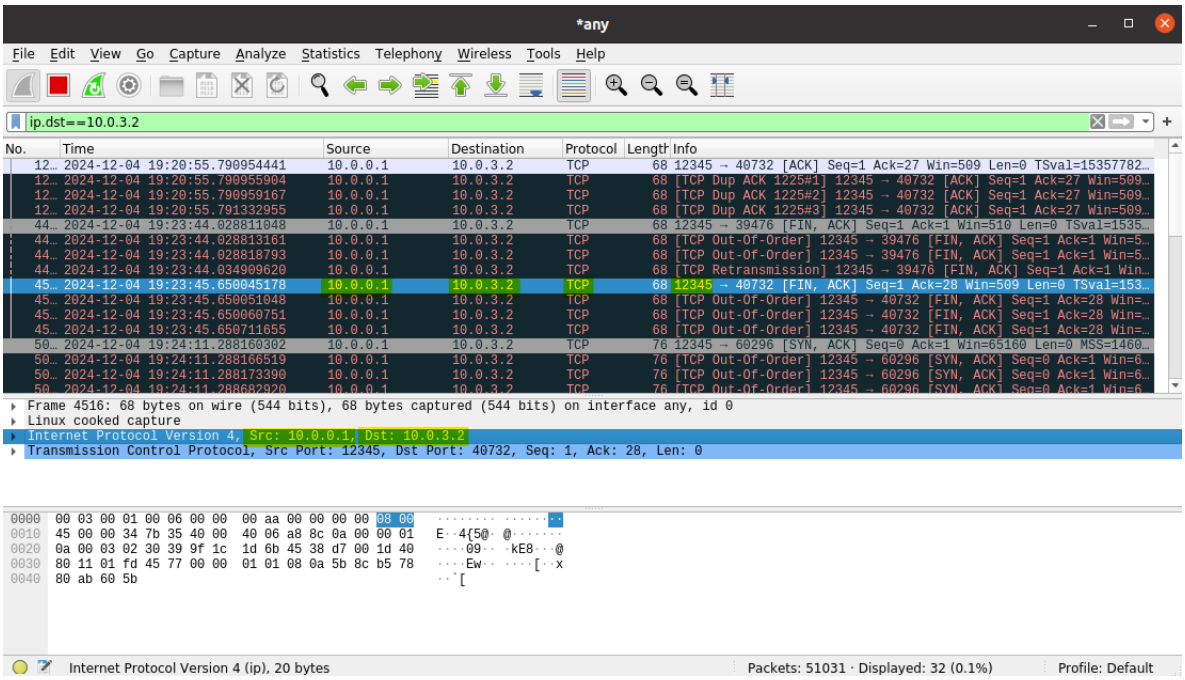
The Wireshark capture showing packet transfer from n1 to n6 is shown below:



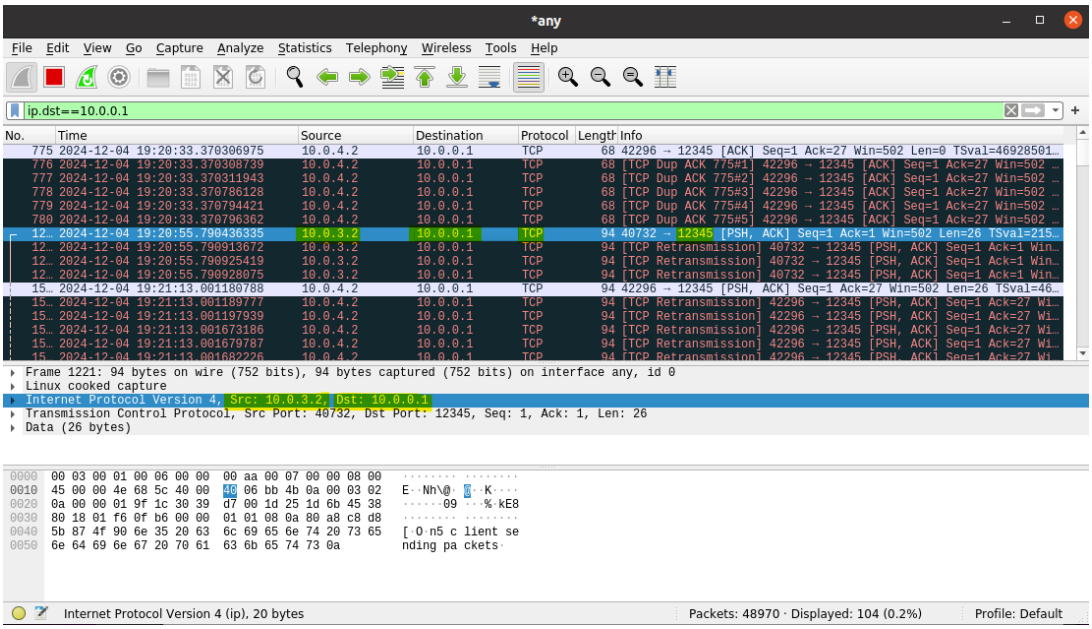
The Wireshark capture showing packet transfer from n6 to n1 proving bidirectional connection is shown below:



The Wireshark capture showing packet transfer from n1 to n5 is shown below:



The Wireshark capture showing packet transfer from n5 to n1 proving bidirectional connection is shown below:

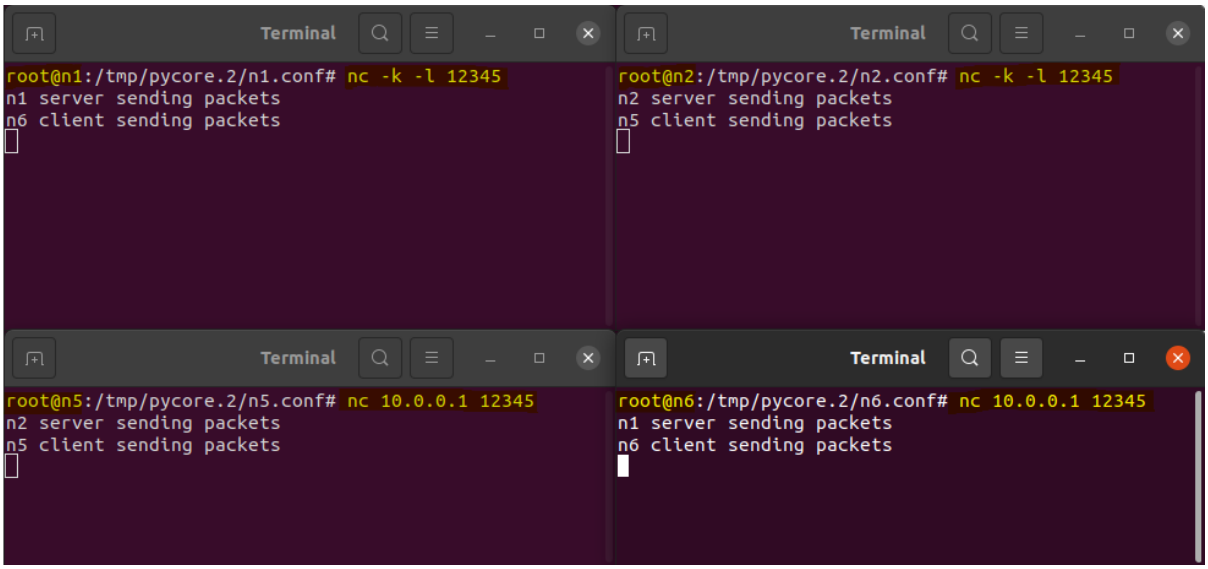


After Load Balancing:

Commands used:

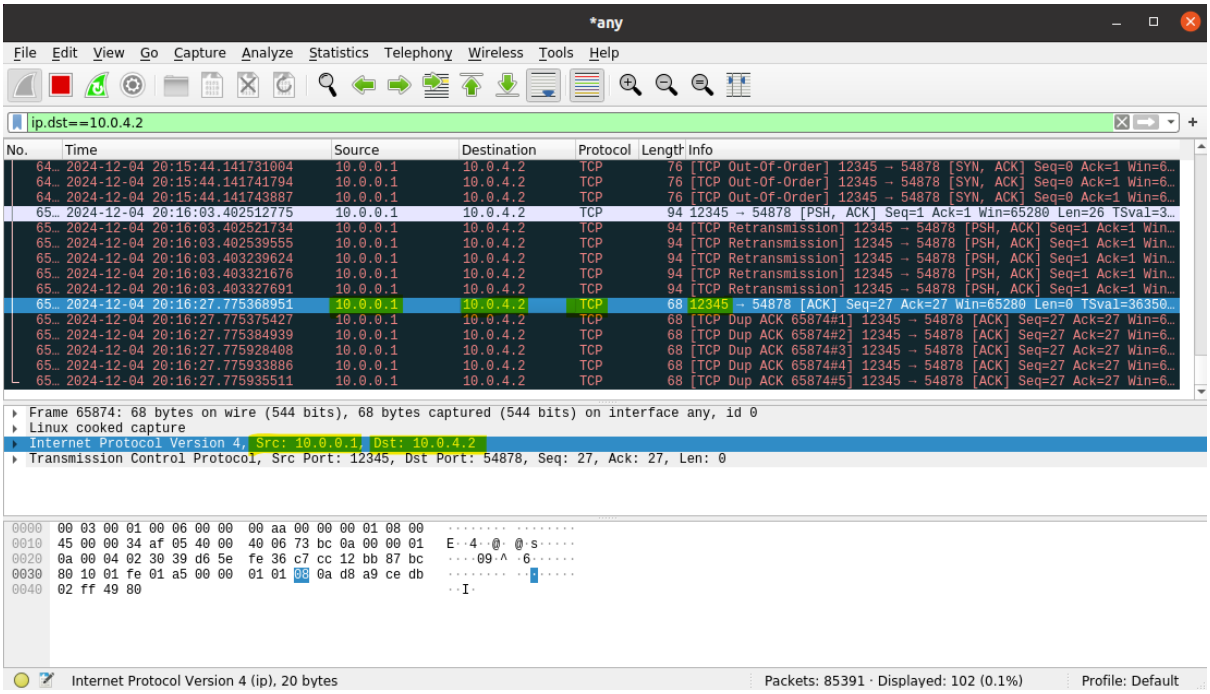
- At n1 server: **nc -k -l 12345**
- At n2 server: **nc -k -l 12345**
- At n5 client: **nc 10.0.0.1 12345**
- At n6 client: **nc 10.0.0.1 12345**

After implementing load balancing, packets from client at node n5 are redirected to server at node n2 instead of n1, while packets from client at node n6 continue to be handled by server at node n1 as shown in the screenshot below:

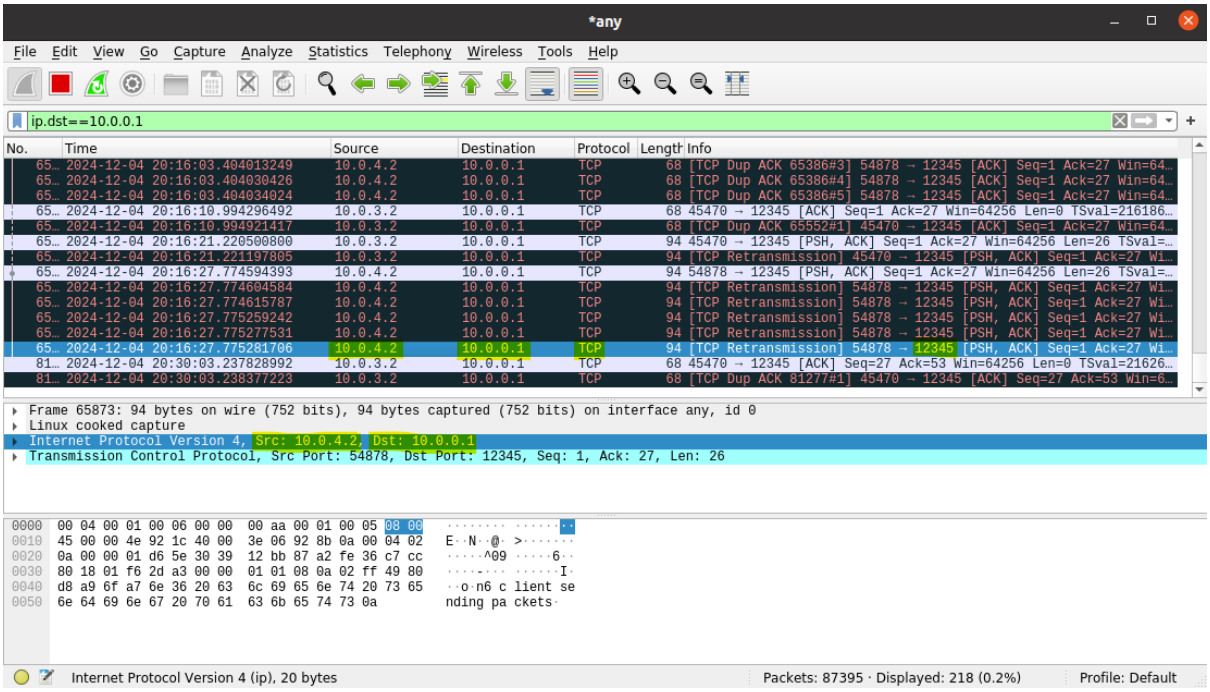


- **The link between n1 and n4 is indicated via the bidirectional link between n1 to n6:**

The Wireshark capture showing n4 as the link between packet transfers from n1 to n6 is shown below:

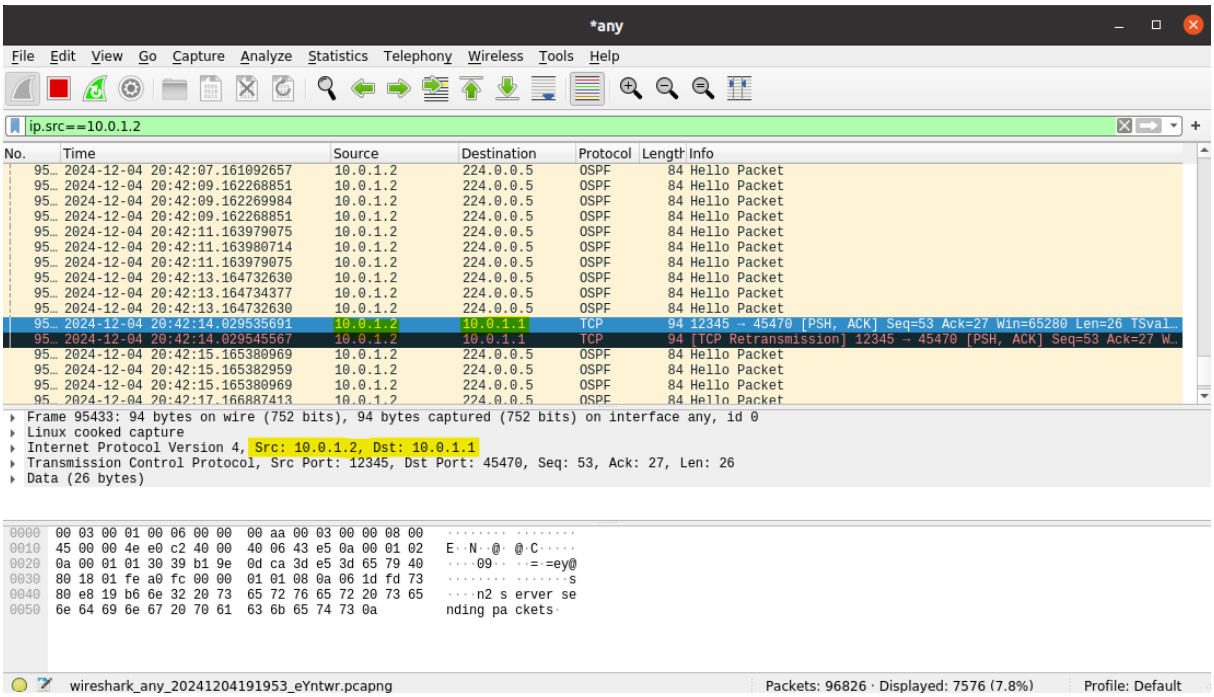


The Wireshark capture showing n4 as the link between packet transfers from n6 to n1 is shown below:

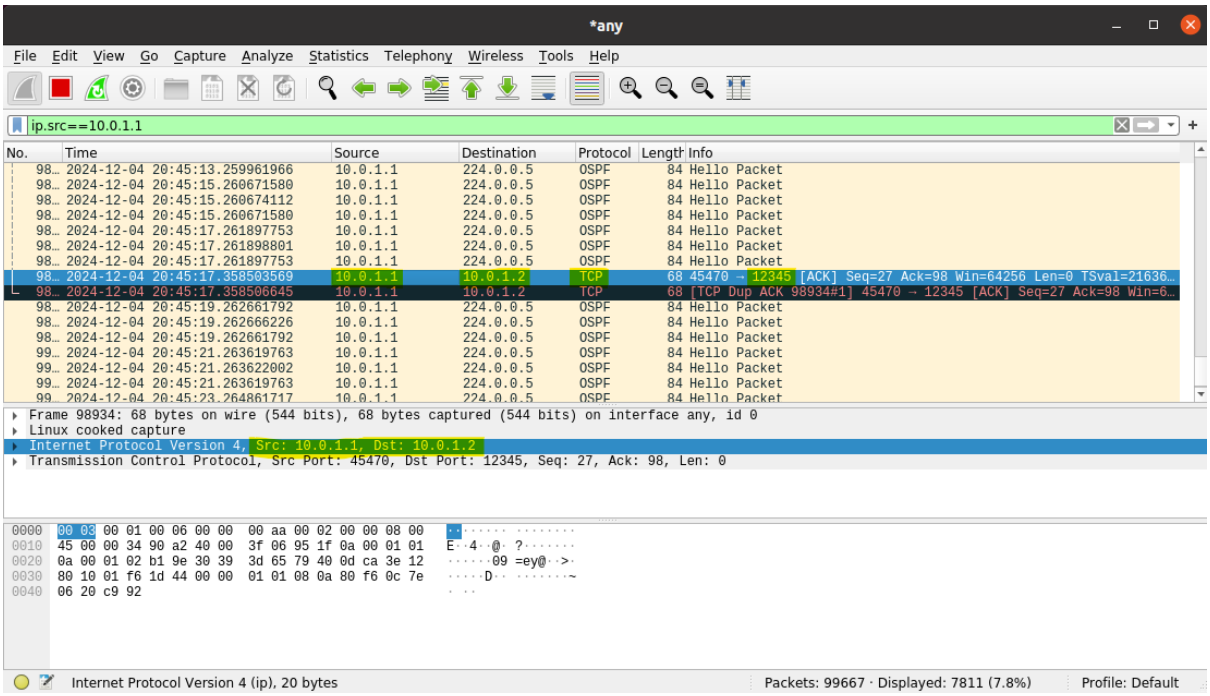


- **The bidirectional link between n2 and n4:**

The Wireshark capturing the link and packet transfers from n2 to n4 is shown below:

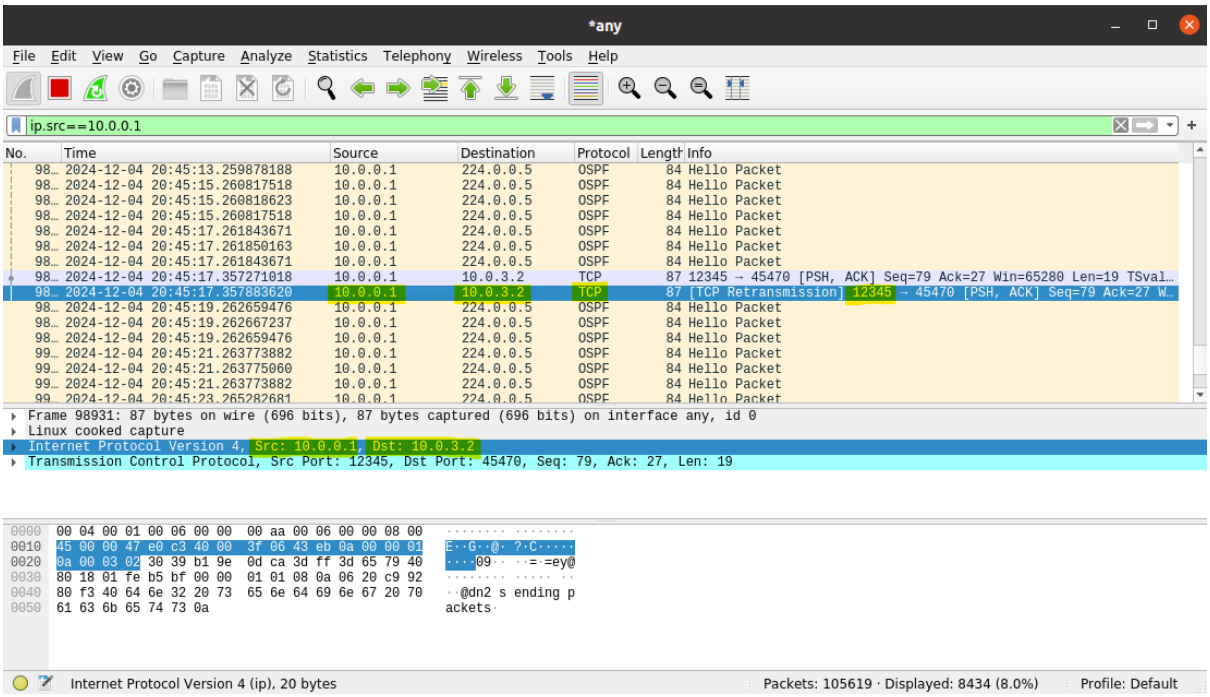


The Wireshark capturing the link and packet transfers from n4 to n2 is shown below:

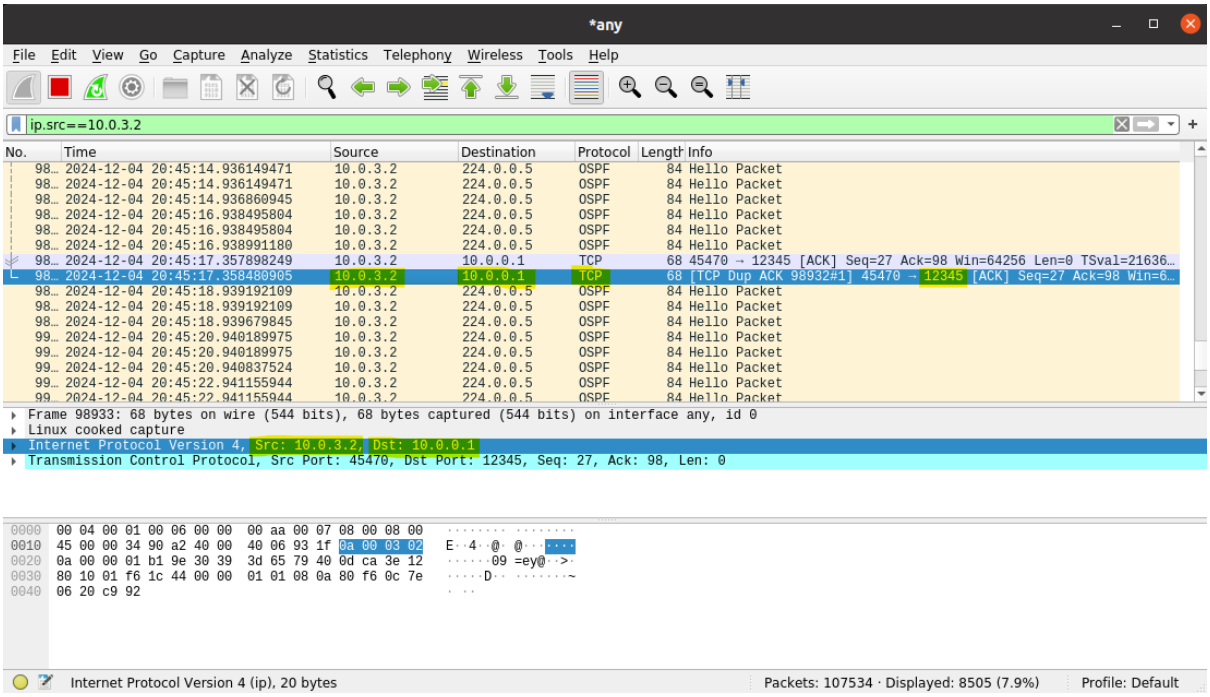


- **The link between n4 to n5 is represented via the bidirectional link between n1 to n5:**

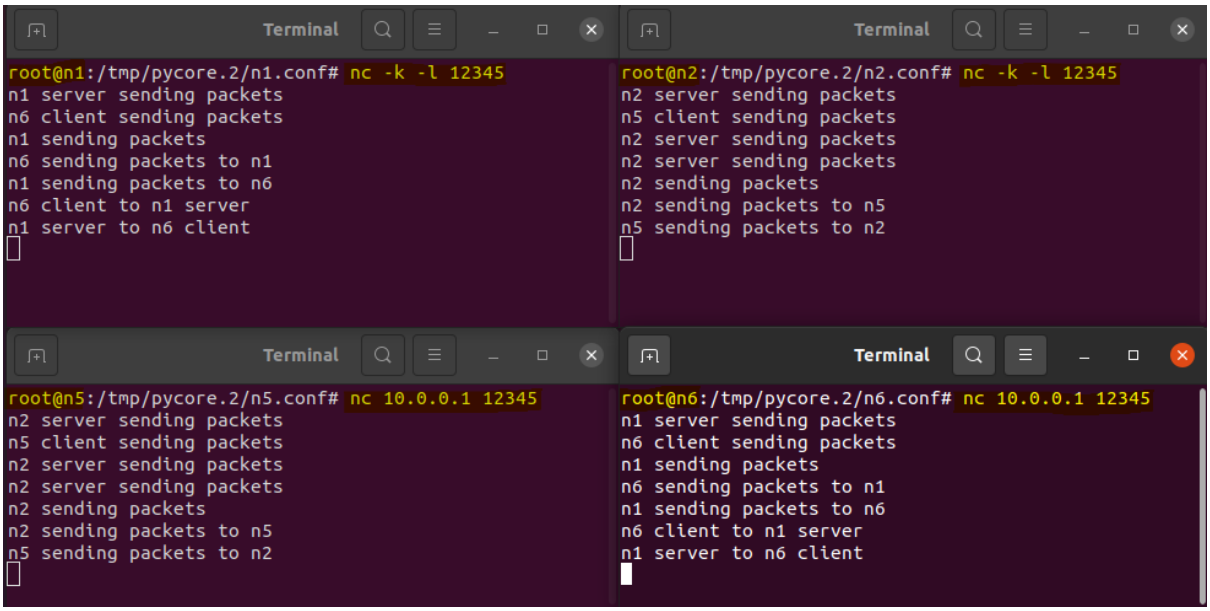
  The Wireshark capturing the link between n4 to n5 via packet transfers from n1 to n5 is shown below:



The Wireshark capturing the link between n4 to n5 via packet transfers from n5 to n1 is shown below:



A summary of the connections established at roots n1, n2, n5 and n6 are shown below:

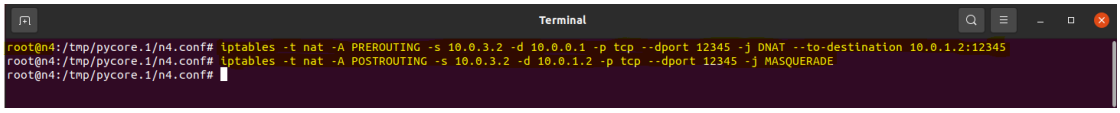


d. Provide the iptables commands you used.

The commands used:

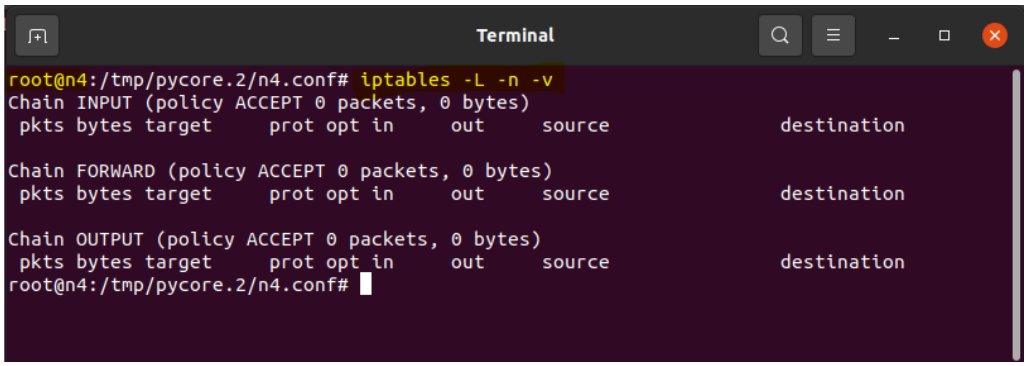
- **iptables -t nat -A PREROUTING -s 10.0.3.2 -d 10.0.0.1 -p tcp –dport 12345 -j DNAT –to-destination 10.0.1.2:12345**
- **iptables -t nat -A POSTROUTING -s 10.0.3.2 -d 10.0.1.2 -p tcp –dport 12345 -j MASQUERADE**

Combination of DNAT and NAT commands used to change traffic and allow bidirectional traffic to go from n5 to n2 is shown below:

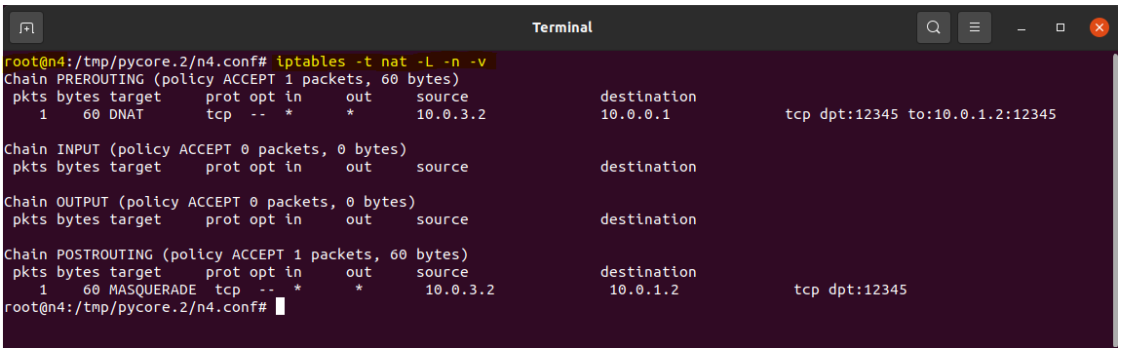


e. Show the output of "iptables -L -n -v", the output of "iptables -t NAT -L -n -v", and the output of "iptables -t mangle -L -n -v".

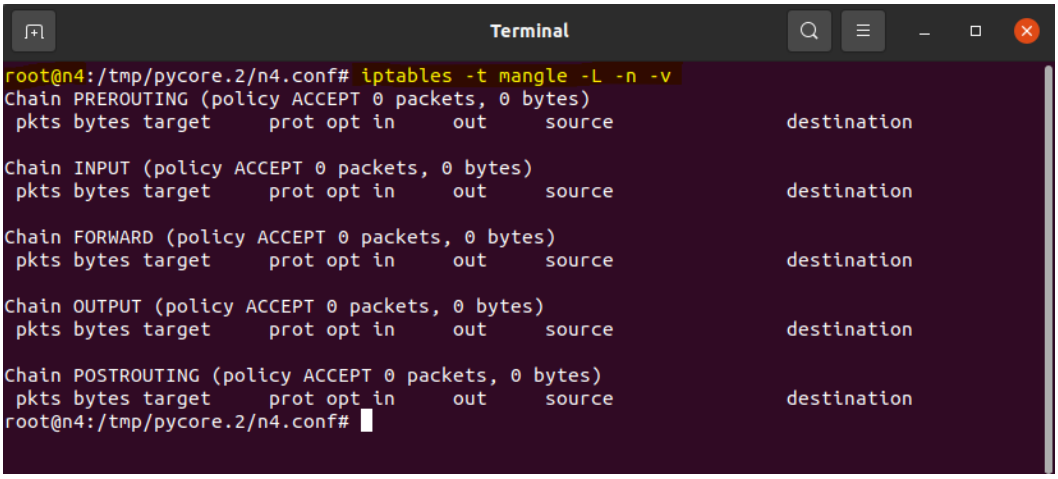
Output of the command **iptables -L -n -v** is shown below:



Output of the command **iptables -t nat -L -n -v** is shown below:



Output of the command **iptables -t mangle -L -n -v** is shown below:



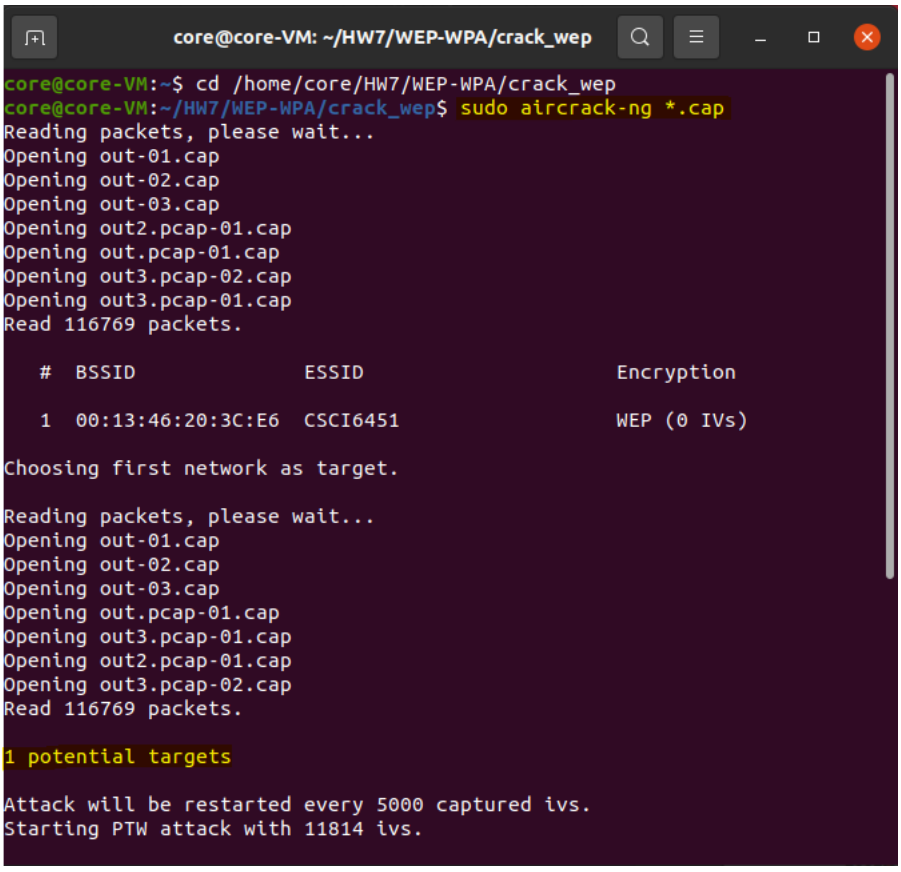
Wireless Section (10pts)

A tar file is included with this exercise. It includes two folders: crack_wpa and crack_wep. You will be using aircrack-ng to crack WPA and WEP. This tool can be installed on any Linux box (using apt-get on Ubuntu for instance) or you can download Kali Linux.

1: WEP

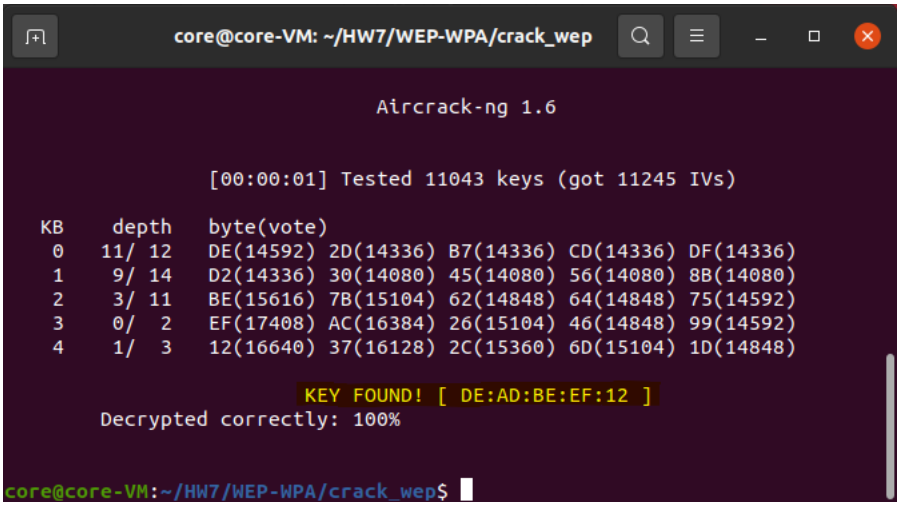
In the directory, there are several captures (*.cap files) from a WLAN network configured for WEP. To crack the password using the run.sh script. All it does is run: aircrack-ng *.cap. The tool will consume all the files and attempt to crack the WEP password.

The output of running the command **sudo aircrack-ng *.cap** is shown below:



Q1: What is the WEP password? It should take you about a second to crack, but this is because you have all the data already. Show screenshot of the cracked password

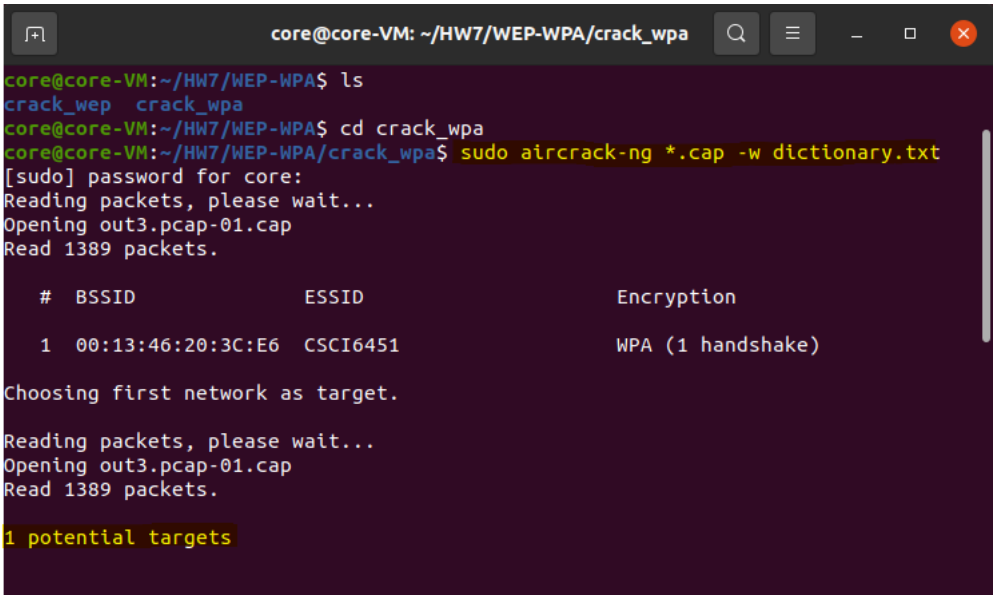
The WEP password is **DE:AD:BE:EF:12** as shown below:



2: WPA

In this directory, there are several captures (*.cap files) from a WLAN network configured for WPA. To crack the password using the run.sh script. It runs a dictionary attack using the dictionary file provided.

The output of running the command **sudo aircrack-ng *.cap -w dictionary.txt** is shown below:



Q2: What is the WPA password? Show screenshot of the cracked password

The WPA password is **DEADBEEF12** as shown below: