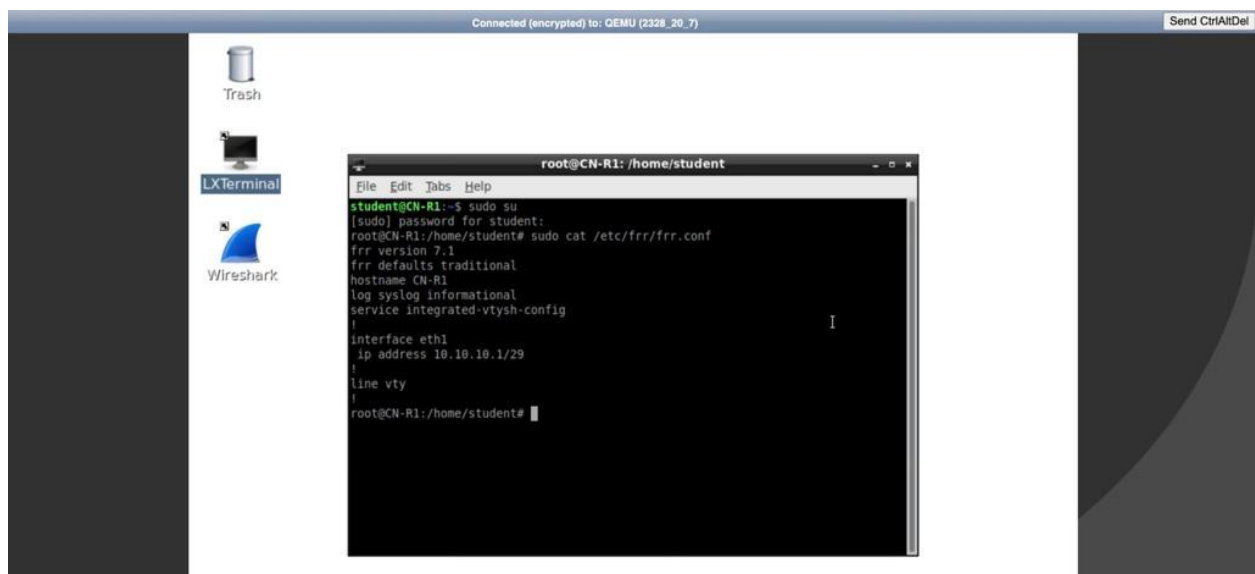
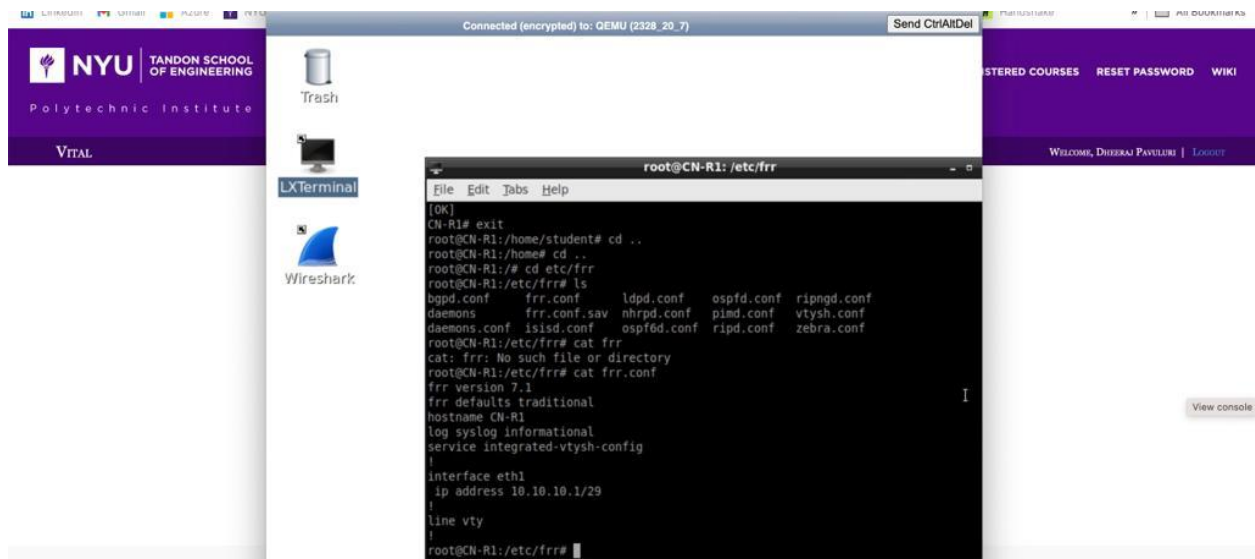
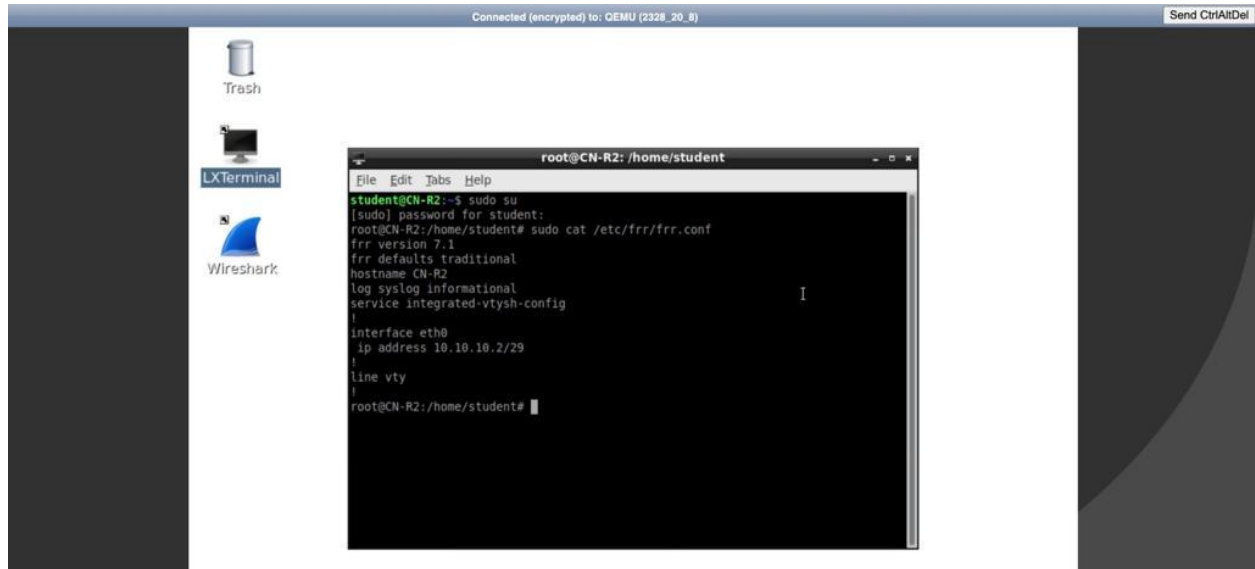


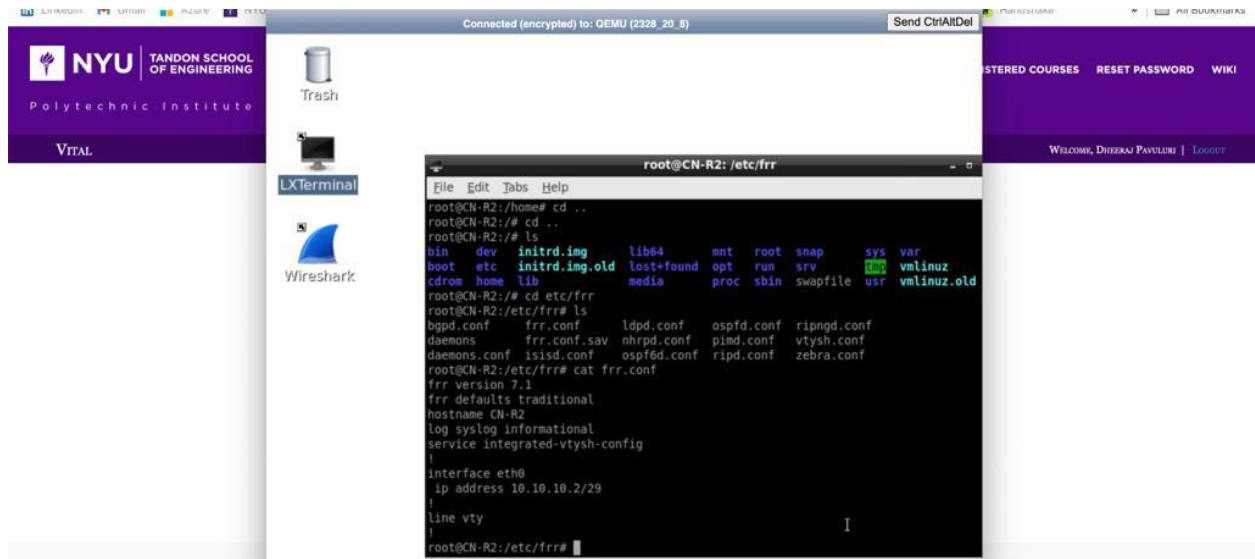
Venkata Naga Dheeraj Pavuluri
N16788493

Device	IP address
Kali	10.10.10.3/29
R1	10.10.10.1/29
R2	10.10.10.2/29

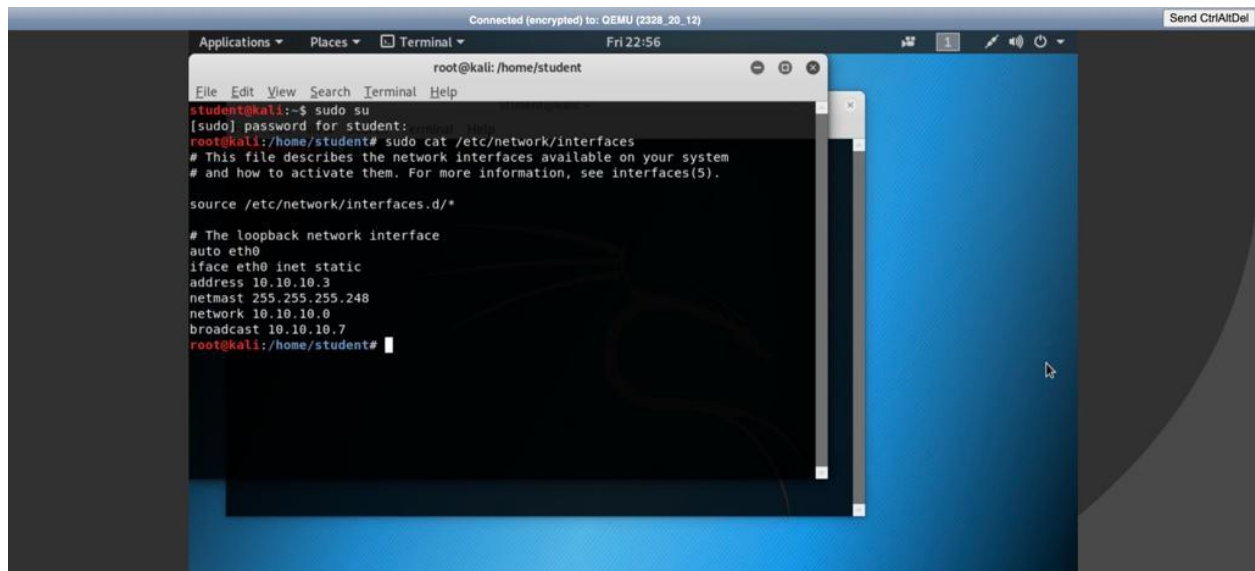
1.

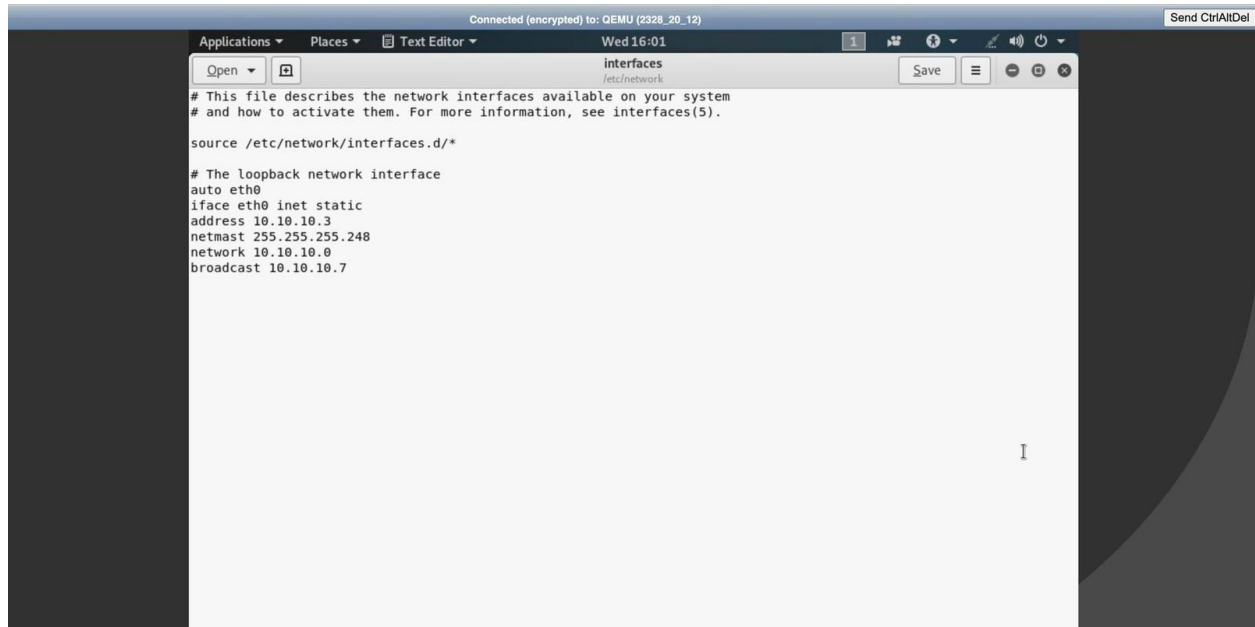






2.





3.

A. In order to provide enough IP addresses for the three devices (R1, R2, and Kali) while reducing IP address waste, we used the /29 subnet mask for Area 0. Despite the fact that a /29 subnet has 8 IP addresses, only 6 of them may actually be utilized for hosts since one address is set aside as the network identification and another is set aside as the broadcast address. This allocation prevents excessive IP address waste by guaranteeing enough addresses for three hosts of devices without resorting to a broader subnet. Here are some detailed information:

1. Number of Devices: A /29 subnet mask can give enough address space without wasting IP addresses if there are less than 6 devices in the region that require unique IP addresses. This is a typical option for networks in tiny offices or remote areas of a larger network. In this scenario, we needed to set up R1, R2, and Kali on three different devices at Area 0. Those devices each need a distinct IP address.

2. Protection: A lower subnet mask, such as /29, can offer some degree of isolation and protection for the devices located there. It restricts the number of gadgets that can speak to one another directly on the same subnet.

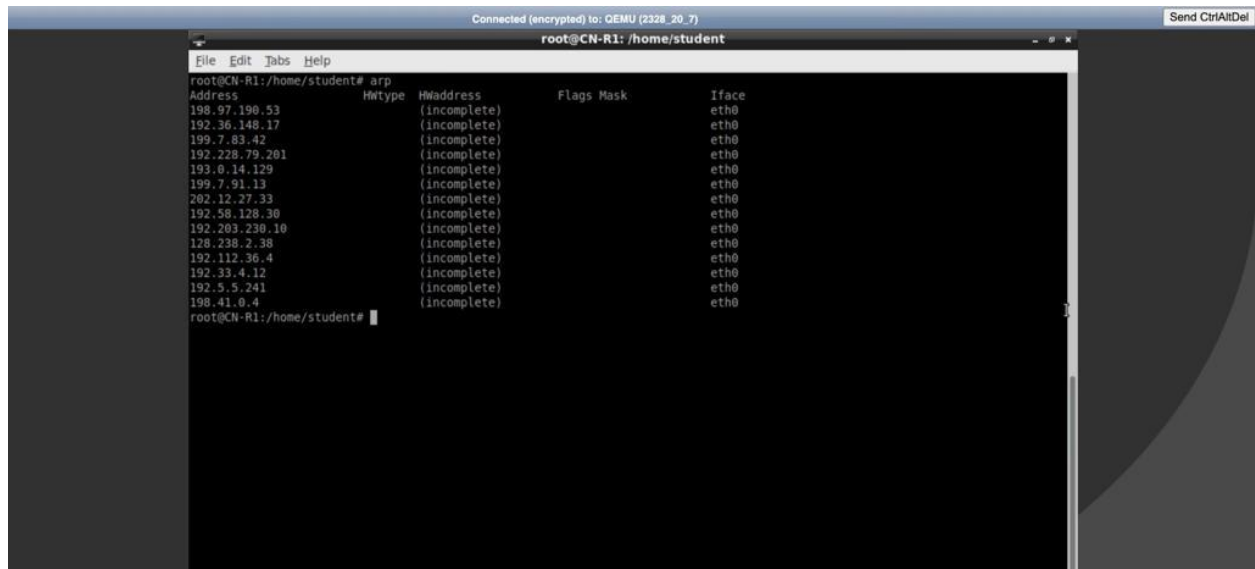
3. IP address conversation: Utilizing smaller subnets can assist in saving IP addresses in situations when there is a shortage of IP address space, such as in IPv4 networks. Fewer IP addresses are used by a /29 subnet than by bigger subnets.

4. Segmentation: You may divide a bigger network into more manageable chunks by using a /29 subnet. A separate division, function, or security area within the broader network may be represented by each subnet.

In conclusion, using the /29 subnet mask achieves a balance between giving the devices you require enough IP addresses while preventing unnecessary IP address waste. It guarantees effective address space use while resolving your immediate network needs.

B.

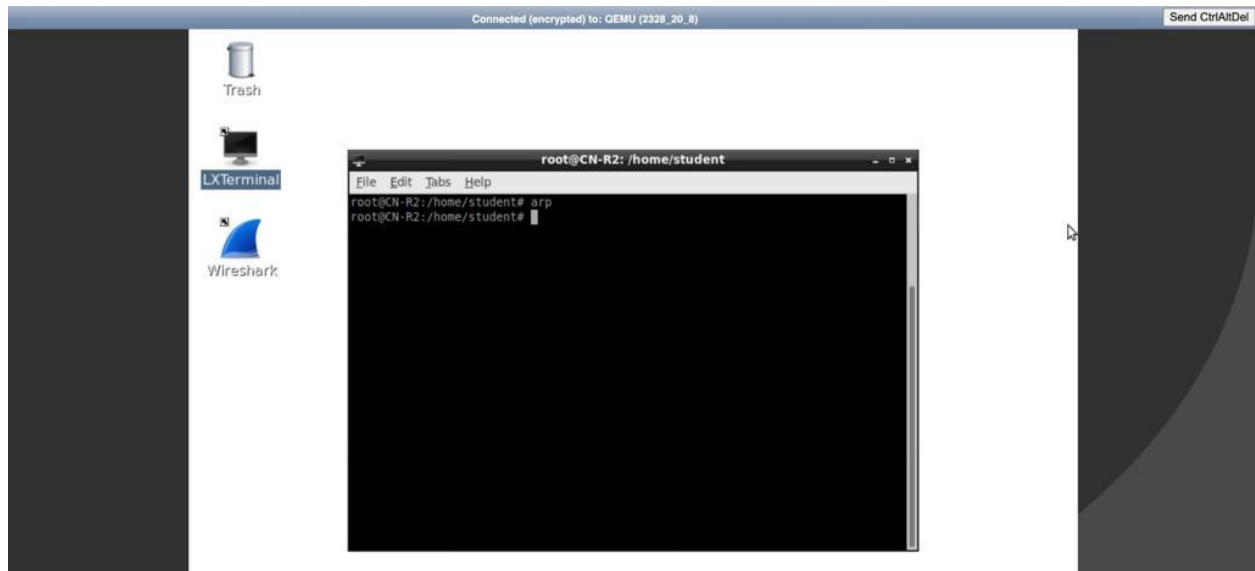
ARP of R1 Before Ping:-



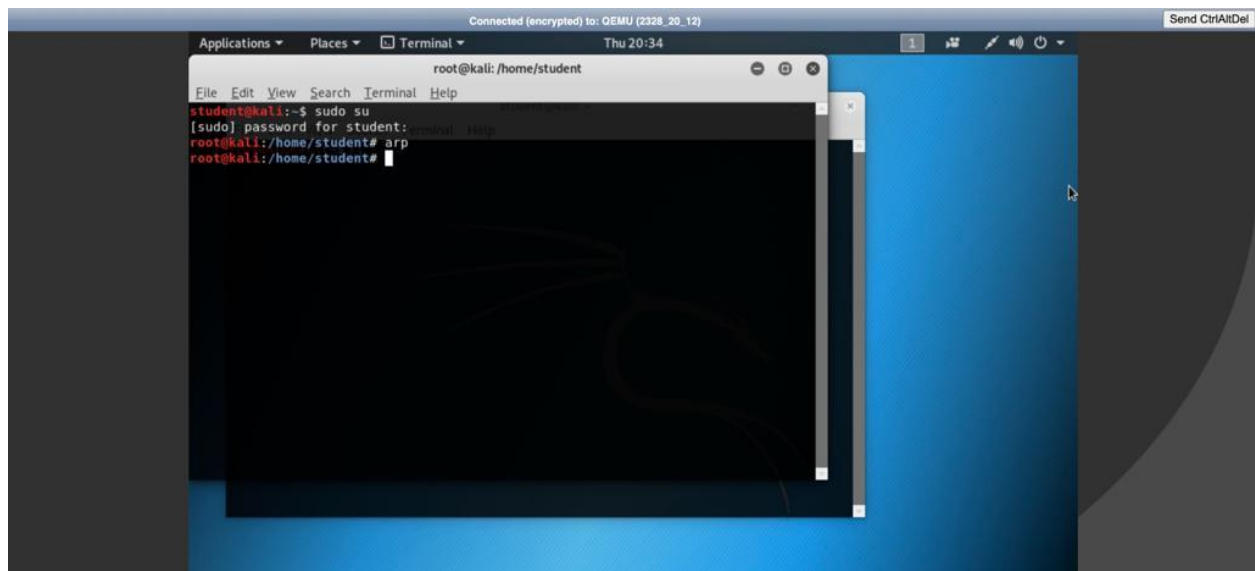
The screenshot shows a terminal window titled "root@CN-R1: /home/student" with a menu bar (File, Edit, Tabs, Help). The terminal output of the 'arp' command is as follows:

Address	Hwtype	Hwaddress	Flags	Mask	Iface
198.97.190.53		(incomplete)			eth0
192.36.148.17		(incomplete)			eth0
199.7.83.42		(incomplete)			eth0
192.228.79.201		(incomplete)			eth0
193.0.14.129		(incomplete)			eth0
199.7.91.13		(incomplete)			eth0
202.12.27.33		(incomplete)			eth0
192.58.128.30		(incomplete)			eth0
192.203.230.10		(incomplete)			eth0
128.238.2.38		(incomplete)			eth0
192.112.36.4		(incomplete)			eth0
192.33.4.12		(incomplete)			eth0
192.5.5.241		(incomplete)			eth0
198.41.0.4		(incomplete)			eth0

ARP of R2 Before Ping



ARP of Kali Before Ping:

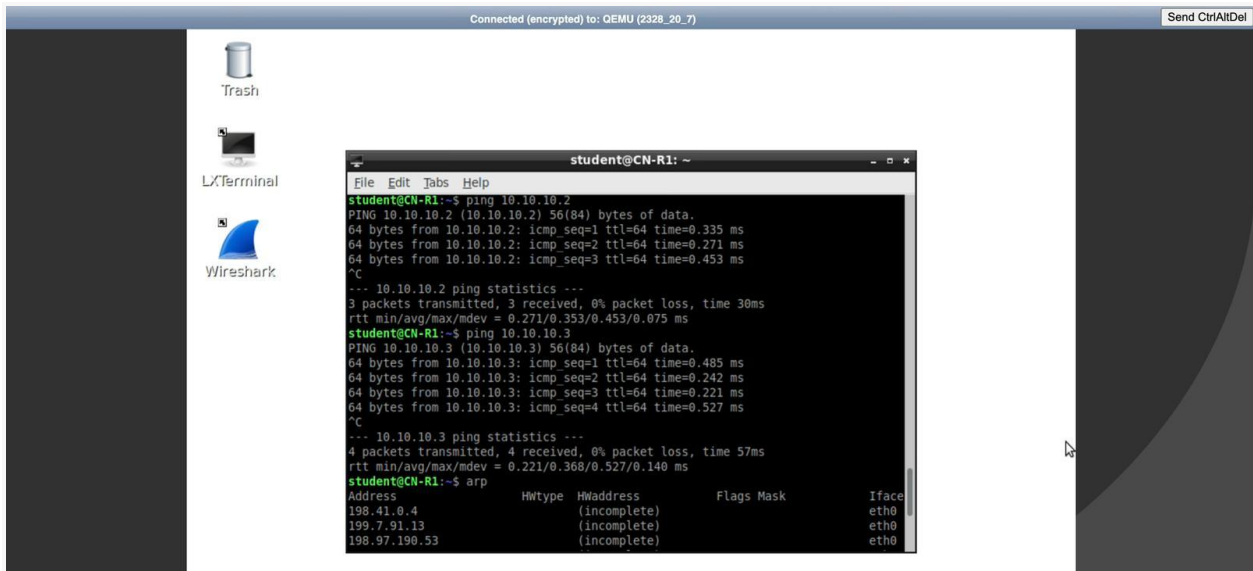


C.

The MAC addresses of the target devices are resolved by ARP procedures when you ping both R2 and Kali from R1.

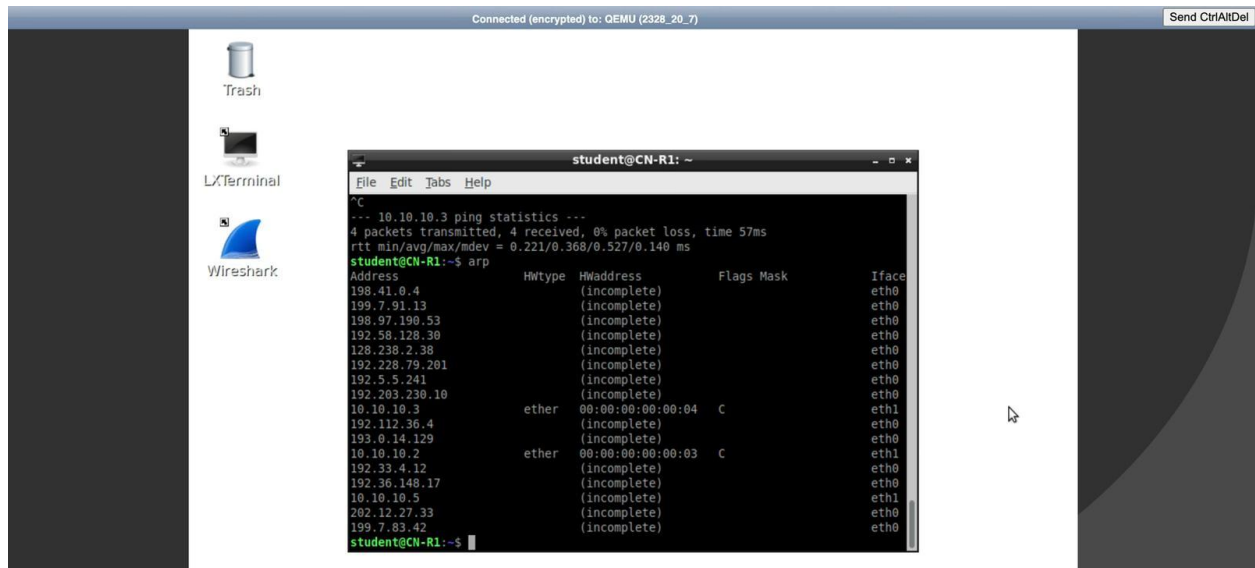
Ping from R1 - R2 & R1 - Kali

- Using, for instance, the IP address 10.10.10.2, R1 wishes to ping R2.
- If R1 already knows the MAC address corresponding to R2's IP address, it will examine its ARP database.
- R1 now intends to ping Kali (using, for instance, the IP address 10.10.10.3).
- Since Kali has not yet spoken with R1 and R1 has not made an ARP request for Kali, when R1 examines its ARP table, it does not include an entry for Kali.

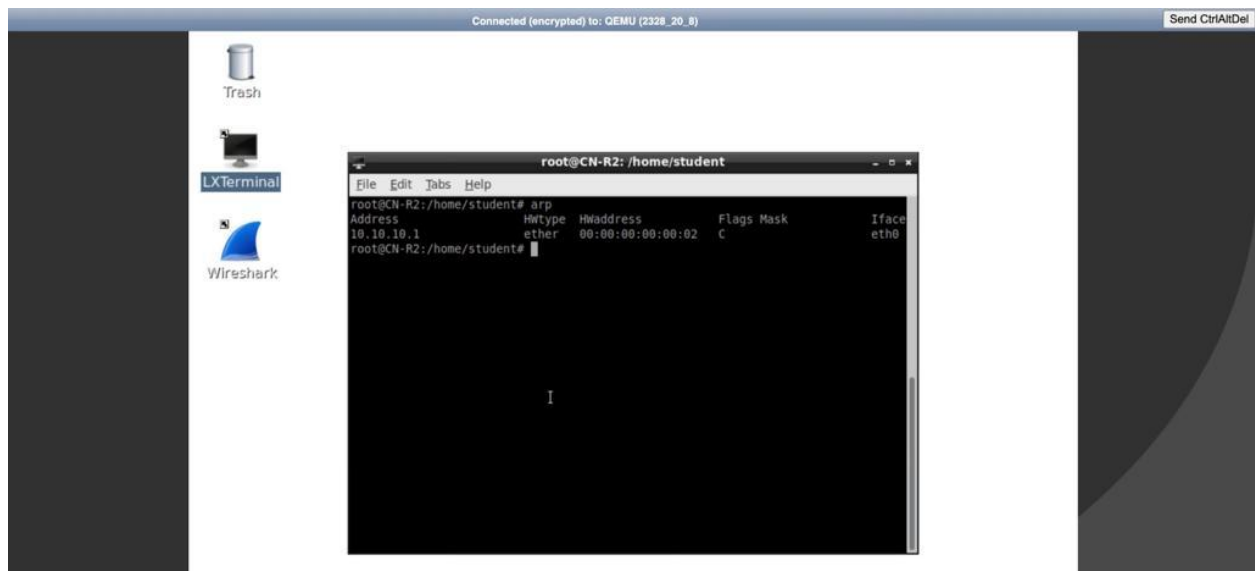


```
student@CN-R1: ~  
File Edit Tabs Help  
student@CN-R1:~$ ping 10.10.10.2  
PING 10.10.10.2 (10.10.10.2) 56(84) bytes of data.  
64 bytes from 10.10.10.2: icmp_seq=1 ttl=64 time=0.335 ms  
64 bytes from 10.10.10.2: icmp_seq=2 ttl=64 time=0.271 ms  
64 bytes from 10.10.10.2: icmp_seq=3 ttl=64 time=0.453 ms  
^C  
--- 10.10.10.2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 30ms  
rtt min/avg/max/mdev = 0.271/0.353/0.453/0.075 ms  
student@CN-R1:~$ ping 10.10.10.3  
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.  
64 bytes from 10.10.10.3: icmp_seq=1 ttl=64 time=0.485 ms  
64 bytes from 10.10.10.3: icmp_seq=2 ttl=64 time=0.242 ms  
64 bytes from 10.10.10.3: icmp_seq=3 ttl=64 time=0.221 ms  
64 bytes from 10.10.10.3: icmp_seq=4 ttl=64 time=0.527 ms  
^C  
--- 10.10.10.3 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 57ms  
rtt min/avg/max/mdev = 0.221/0.368/0.527/0.140 ms  
student@CN-R1:~$ arp  
Address HWtype HWaddress Flags Mask Iface  
198.41.0.4 (incomplete) eth0  
199.7.91.13 (incomplete) eth0  
198.97.190.53 (incomplete) eth0
```

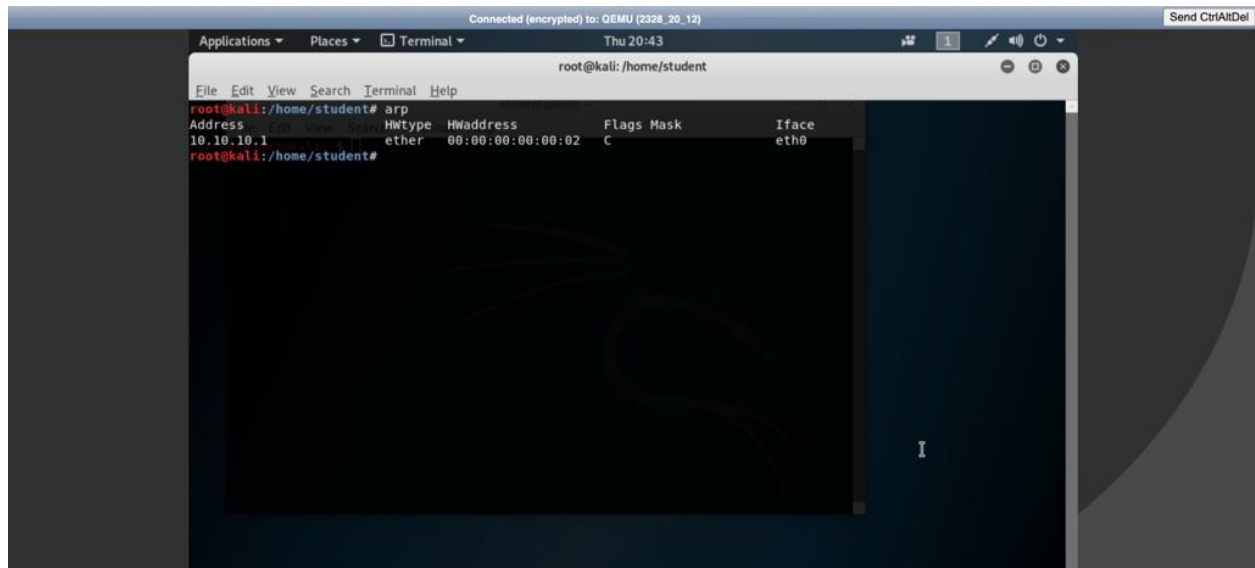
R1 ARP Table after ping:



ARP table of R2:

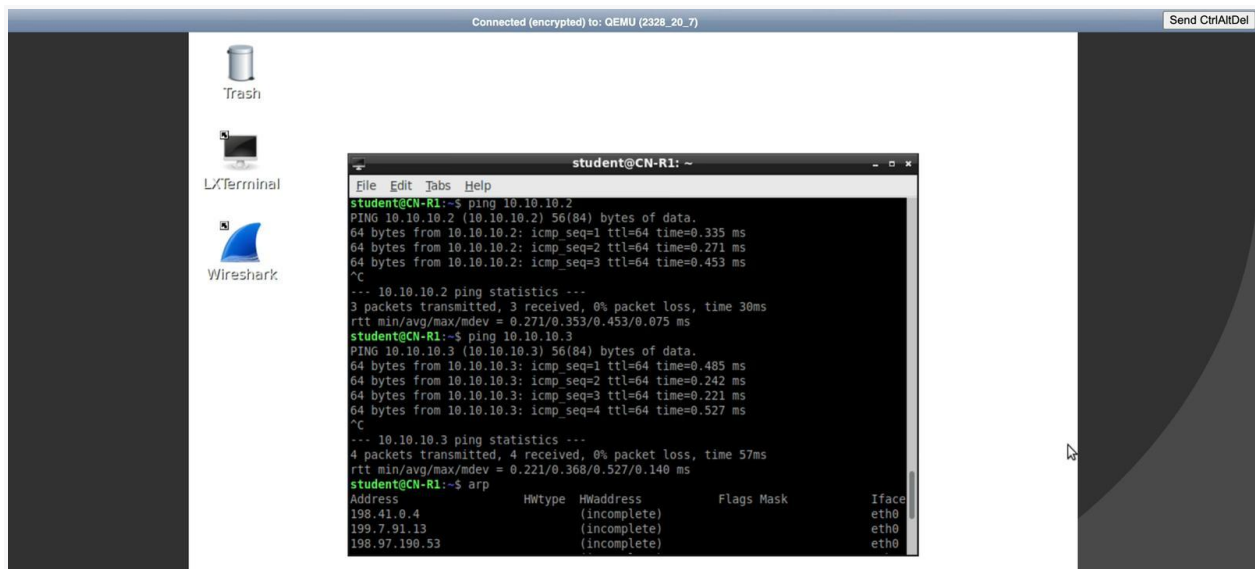


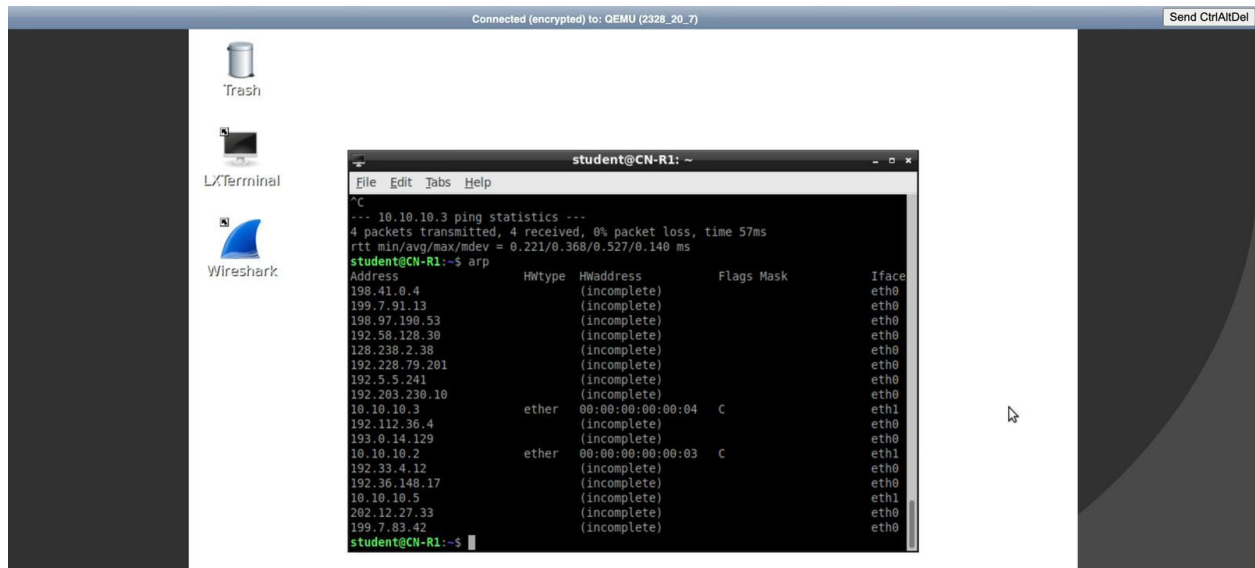
ARP table of Kali:



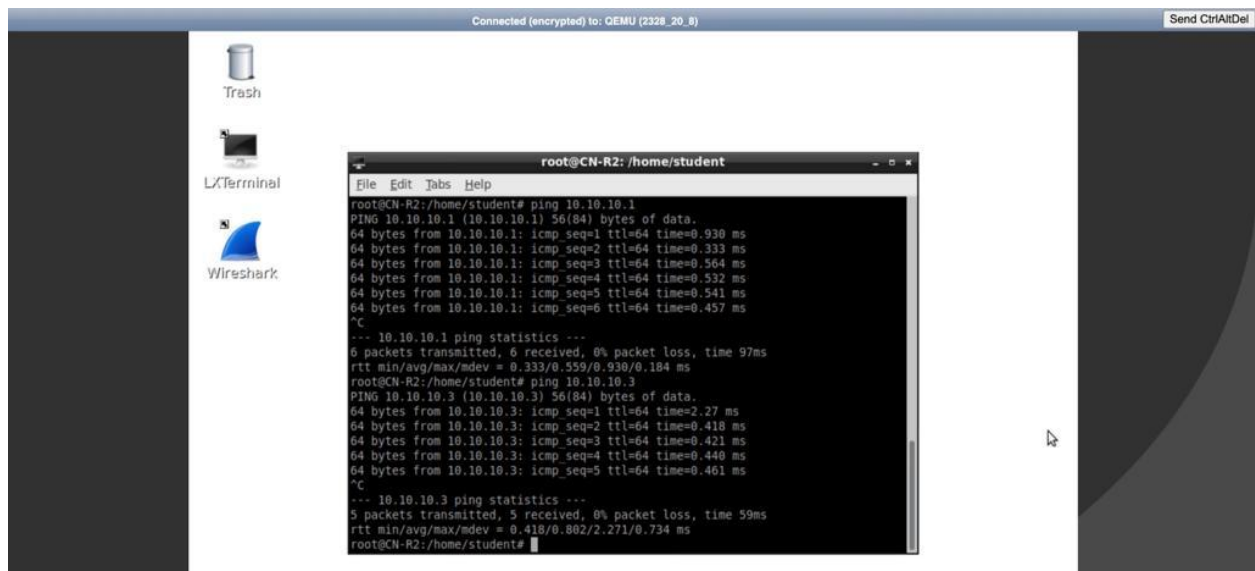
R2 is aware of R1 as a result of these operations since it responded to the ARP request during the ping from R1 to R2. However since Kali did not take in this ARP transaction, R2 doesn't have an entry for Kali in the ARP table. ARP entries are generated when devices interact directly with one another on the same network segment. R2 will have an ARP entry for Kali in its ARP table once they are in direct contact.

Testing ping works from R1 - R2 & R1 - Kali and ARP table:

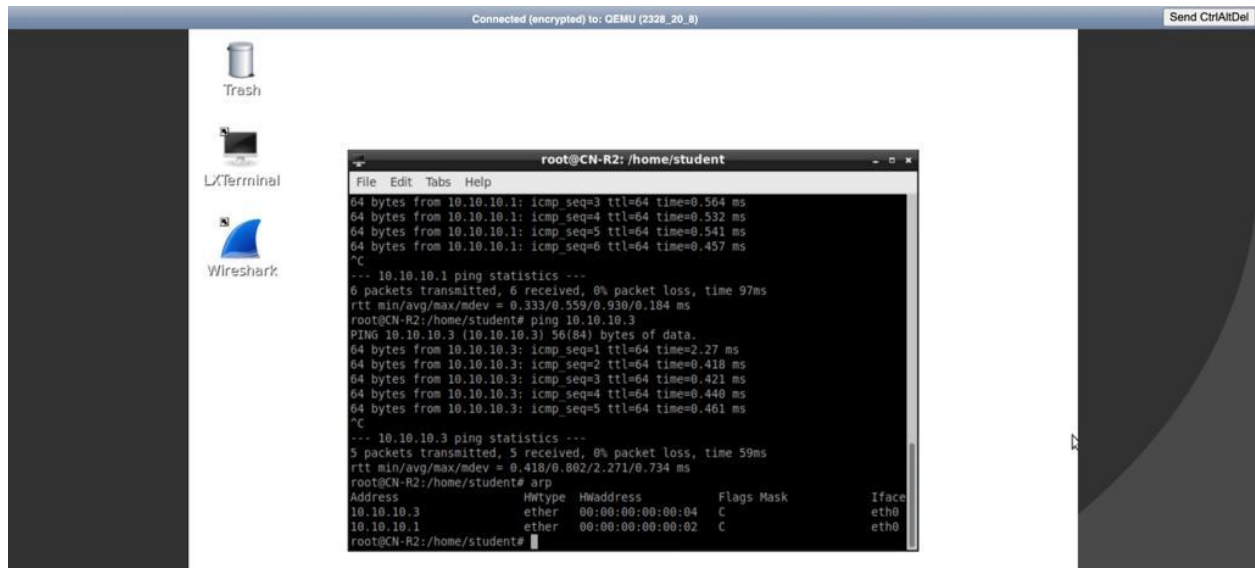




Testing Ping works from R2 - R1 & R2 - Kali



ARP from R2 - R1 & R2 - Kali



Testing Ping works from Kali - R1 & Kali - R2 and ARP Table:

