PRANITA CAMASAMUDRAM

LAB REPORT - 1
Traffic Sniffing

INTRODUCTION : This lab introduces us to the concept of network traffic analysis, in particular, the data collection done in that process and it is an extremely important step.

1. Describe the layers and the fields of a packet captured from your network.

Ans. The layers and fields of a packet captured from the network are:
Transport layer - The transport layer builds on the network layer and has data passed from application layer. It contains many fields including src, dst port number, acknowledgement number (ack), sequence number, header length, flags and options.

Application Layer - It communications with the applications of the software to establish communication. This contains all the upper layer information.

Internet Layer - This layer takes care of sending the packets from any network. It can be seen as IPv4 in the wireshark. This layer contains the src and dst address, header, version, identification, flags, total length, header checksum, fragment, time to live. Many routing protocols belong to this network.

Ethernet Layer- This is the physical layer that contains all detail about the packet and where it has arrived from (source machine). It also has information about the destination machine.

2. Explain the characteristics and functioning of a hub, switch and router in network Science.

Ans. Hub - It is a common connection point for multiple devices in a network. It is mostly used when there is a LAN network. It has multip;e ports and connections in a hub are made through attaching cables in a port. It is the least expensive and least complicated of all three.

Switch- It is very similar to a hub as it is a connection point to many networks. It is comparatively better at passing the information of a packet across the network. As it records all the MAC addresses of the machines, it knows where the pocket came from and where to send it to. If the switch does not know the destination address, it sends it to all the devices in connection.

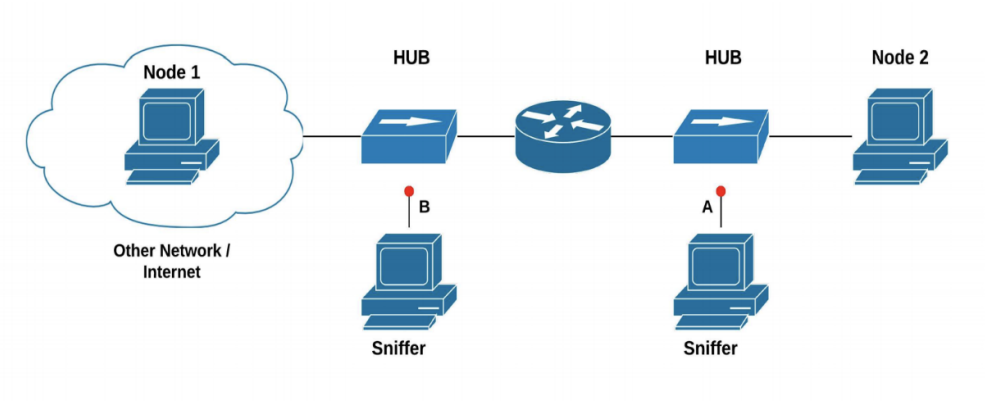
Router - It is the smartest of all the three. It is connected to at least 2 networks, the LAN and WAN. The router has the ability to understand the data that packets. It learns all the information through just the IP addresses. With the help of headers and tables, it learns the best way to route a data packet.

3.Why is the use of SDNs as a tool necessary for DDoS?

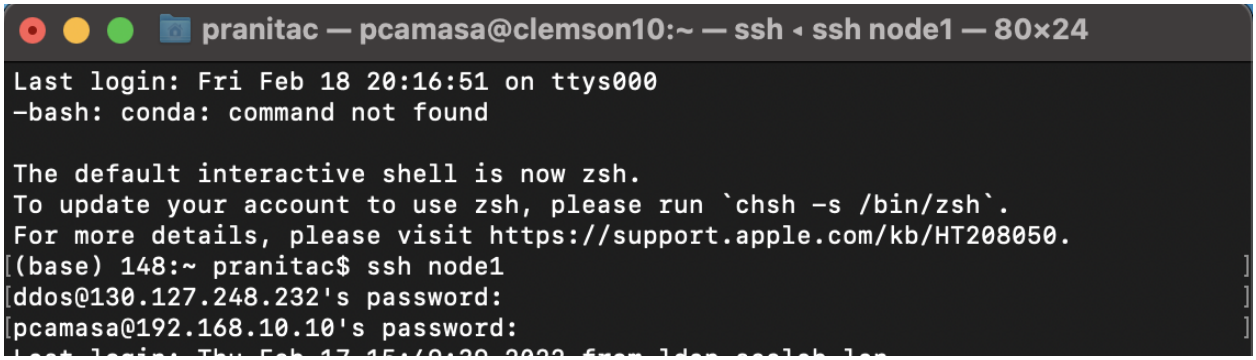
Ans. SDN has an architecture which is central and the entire network can be controlled. One of the uses of SDN that I found interesting was how efficient it becomes when it is merged with cloud computing technologies. This reduces the complexity of the cloud and improves the scalability. This means that when a ddos attack is launched on the cloud, due to improved manageability, it becomes easier to detect the attack and mitigate it.

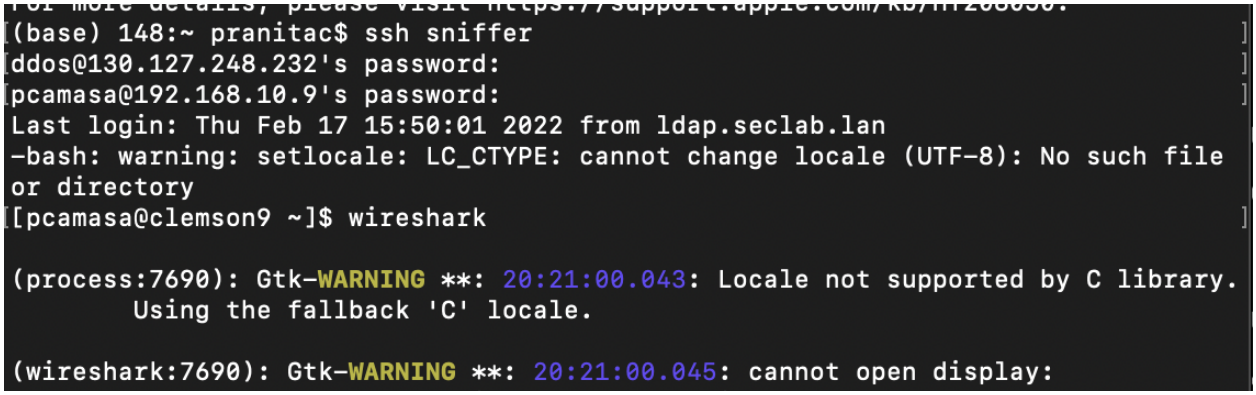
LAB SETUP:

The below picture shows the setup of the lab machines. The node 1 above is the machine in clemson with the IP address 192.168.10.10 and node 2 is supposed to be the Charleston computer but due to technical difficulties we are using the virtual machine 192.168.30.2 which acts similar to the above.

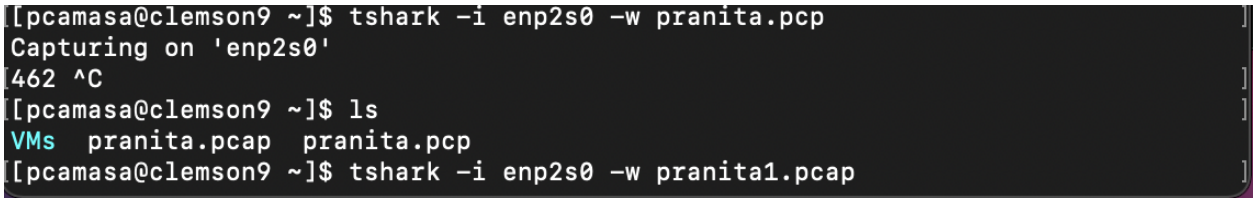


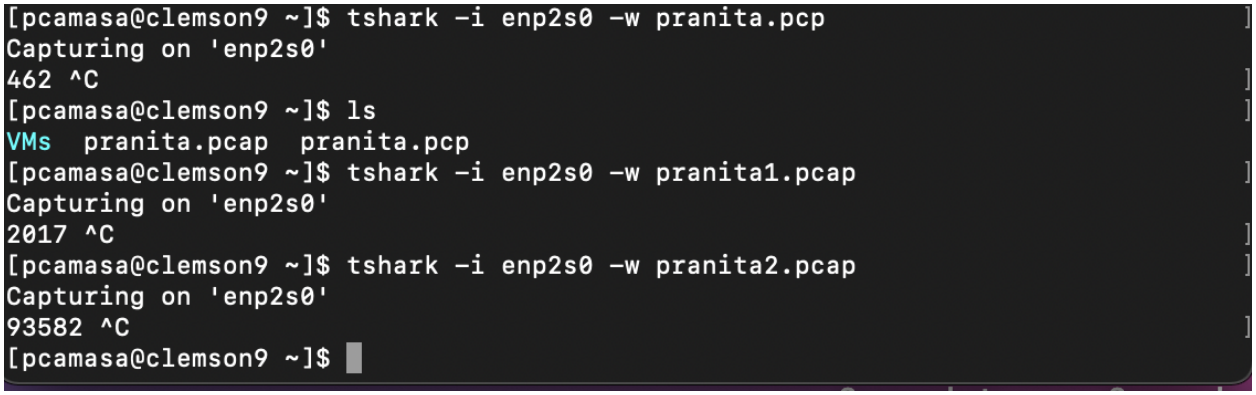
First I opened the ssh node 1 and ssh sniffer on two different terminals.





As I was not able to open wireshark during this lab on my macbook. I directly opened tshark using the command tshar -i enp20 -w pranita1.pcap and started capture.





On node1, I ran the command ping 192.168.30.2 and captured these on the sniffer.

```
[pcamasa@clemson10 ~]$ ping 192.168.30.2
PING 192.168.30.2 (192.168.30.2) 56(84) bytes of data.
64 bytes from 192.168.30.2: icmp_seq=1 ttl=63 time=7.86 ms
64 bytes from 192.168.30.2: icmp_seq=2 ttl=63 time=1.22 ms
64 bytes from 192.168.30.2: icmp_seq=3 ttl=63 time=1.26 ms
64 bytes from 192.168.30.2: icmp_seq=4 ttl=63 time=1.27 ms
64 bytes from 192.168.30.2: icmp_seq=5 ttl=63 time=1.25 ms
64 bytes from 192.168.30.2: icmp_seq=6 ttl=63 time=1.25 ms
64 bytes from 192.168.30.2: icmp_seq=7 ttl=63 time=1.01 ms
64 bytes from 192.168.30.2: icmp_seq=8 ttl=63 time=1.24 ms
64 bytes from 192.168.30.2: icmp_seq=9 ttl=63 time=1.04 ms
64 bytes from 192.168.30.2: icmp_seq=10 ttl=63 time=1.08 ms
```

```
--- 192.168.30.2 ping statistics ---
34 packets transmitted, 34 received, 0% packet loss, time 33037ms
rtt min/avg/max/mdev = 0.871/1.296/7.865/1.147 ms
```

Similarly I did the tshark command and generated another file called pranita2.pcap.

I then downloaded them on the sniffer machine using scp command as shown below.

```
(base) 148:~ pranitac$ scp sniffer:/home/pcamasa/pranita1.pcap ~/
ddos@130.127.248.232's password:
pcamasa@192.168.10.9's password:
/etc/profile.d/lang.sh: line 19: warning: setlocale: LC_CTYPE: cannot cha
ale (UTF-8): No such file or directory
pranita1.pcap                                 100%  277KB 281.0KB/s   00:
```

```
(base) 148:~ pranitac$ scp sniffer:/home/pcamasa/pranita2.pcap ~/
ddos@130.127.248.232's password:
pcamasa@192.168.10.9's password:
/etc/profile.d/lang.sh: line 19: warning: setlocale: LC_CTYPE: cannot change loc
ale (UTF-8): No such file or directory
pranita2.pcap                                 100%  109MB   1.3MB/s   01:26
```

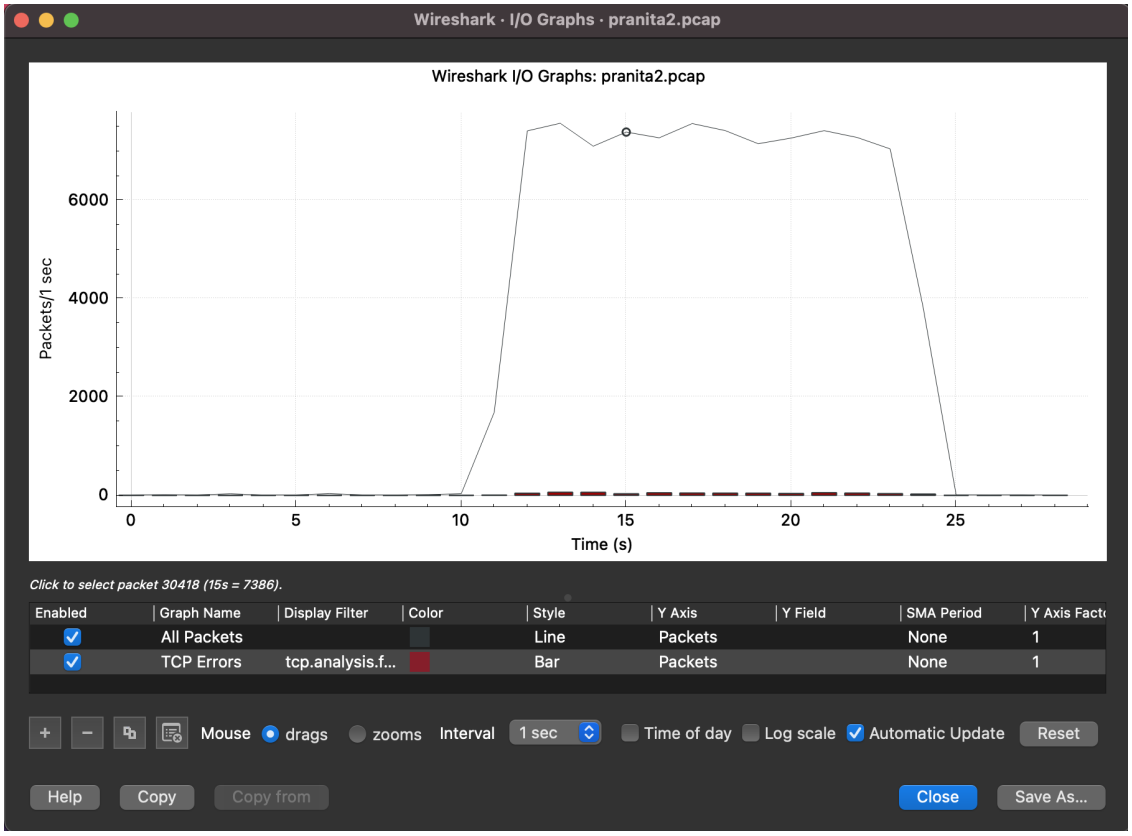
I also generated a large trash file as file.txt as dd if=/dev/urandom of=file.txt bs=1048576 count=100, on the sniffer machine and downloaded it using scp.

```
[pcamasa@clemson10 ~]$ scp file.txt pcamasa@192.168.30.2:/home/pcamasa
The authenticity of host '192.168.30.2 (192.168.30.2)' can't be established.
ECDSA key fingerprint is SHA256:U7l0manBX2Im21VarmHZXc9G/eYY56xSA0WADAwv2gQ.
ECDSA key fingerprint is MD5:1a:e3:3a:36:67:c5:11:7a:ef:3e:6e:9d:3f:f8:0a:14.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.30.2' (ECDSA) to the list of known hosts.
pcamasa@192.168.30.2's password:
file.txt                                      100%  100MB   7.8MB/s   00:12
[pcamasa@clemson10 ~]$ scp file.txt pcamasa@192.168.30.2:/home/pcamasa
pcamasa@192.168.30.2's password:
file.txt                                      100%  100MB   7.9MB/s   00:12
```

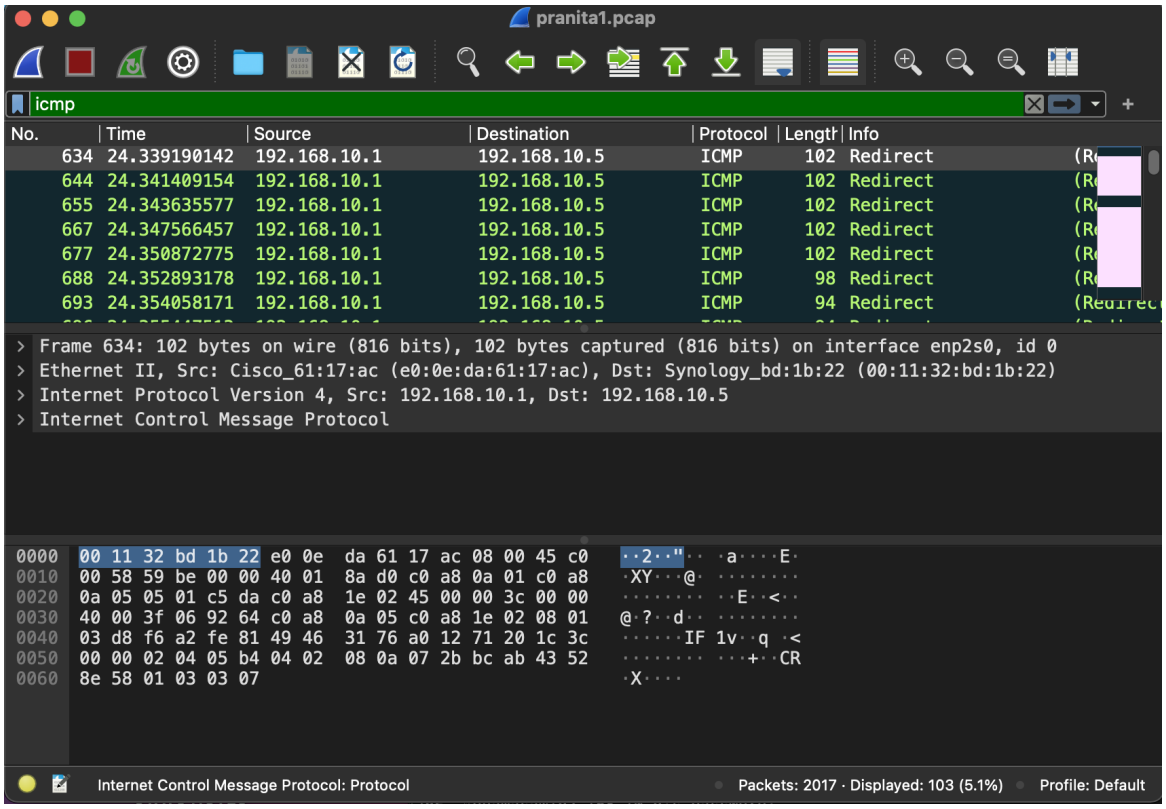
```
[pcamasa@clemson10 ~]$ dd if=/dev/urandom of=file.txt bs=1048576 count=100
100+0 records in
100+0 records out
104857600 bytes (105 MB) copied, 0.493481 s, 212 MB/s
[pcamasa@clemson10 ~]$ ls
VMs  file.txt
```

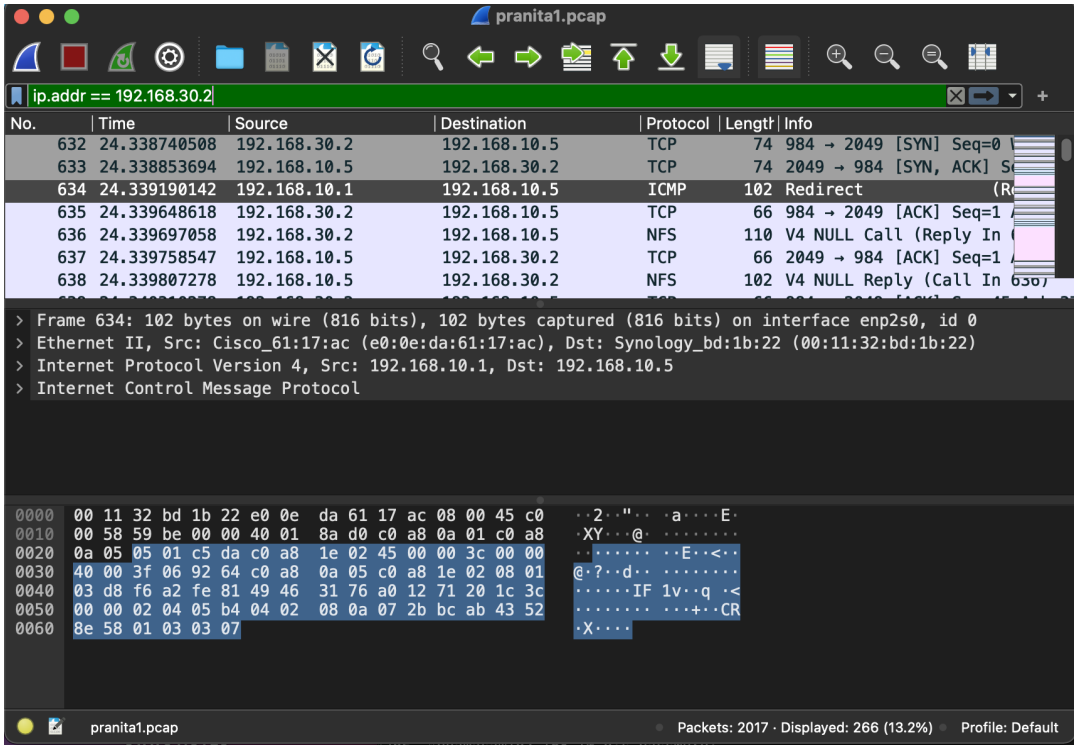
After deleting the file.txt, I opened pranita1.pcap and pranita2.pcap on the wireshark manually. Below is the time series graph. The first one shows the graph generated when ping command was used and the second graph shows when the pranita2.pcap was opened on the wireshark during the transfer.

PRANITA CAMASAMUDRAM



Below is the wireshark when icmp filter was used on the pranita1.pcap file. As you can see below we can see the redirects of the IP addresses.





PRANITA CAMASAMUDRAM

Spoofing Exercise

INTRODUCTION : This lab introduces us to the concept of spoofing which is the scenario where a spoofed packet pretends to be the original packet all the while getting access to the victim's system. We are introduced to tools like scapy and nmap in this exercise which help create the spoofed packets in order to send it over the network. We used wireshark to capture the spoofed packets.

Questions:

1. How can you evade detection when spoofing others? Is it possible to ascertain the identity of the sender?

Ans. While spoofing others few techniques can be used so that in order to avoid getting detected. As I did in the experiment below, they can try and use the active hosts on the network as the spoofing address. This will keep the attacker under radar and see the IP address coming up as less suspicious.

There is no definitive way to ascertain the identity of the sender as the spoofed packets location and IP address both are masked. Hence even if we get that information we don't know for sure if it is reliable.

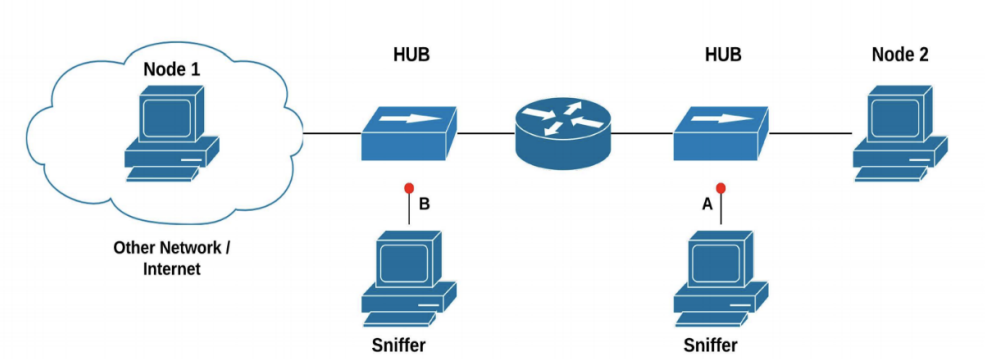
2. Take a look at the RFC for the Internet Protocol, RFC791(https://www.ietf.org/rfc/rfc791.txt)
Explain what ip address spoofing is, and what a host on the network must do to spoof its ip address.

Ans. IP address spoofing is a technique where the attacker creates such a packet where it's source and destination address is masked and there is no definitive way of telling where the spoofed packet came from.

To spoof an IP address in this lab we used scapy and nmap which detects the active hosts and then give one of the hosts as a spoofed address using scapy. This is how we can lie to machine that we have the spoofed IP address. As the protocols in the network layers wrap the header details, there is no definitive way of telling the spoofed packet.

3.Explain why an attacker cannot just grab any existing IP packet carrying UDP or TCP, change only the IP addresses in there, and expect the target host to accept the packet. Especially for TCP, you don't have to read the entire RFC but focus on the header.

Ans. In my opinion this kind of packet grabbing is very inefficient as the header of the packets are wrapped multiple times. And then each packet is attached with an IP protocol header. This way the host would come to know if there was any malicious attempt at the packet.



The above setup is the overview of the lab machines setup. The node 1 above is the machine in clemson with the IP address 192.168.10.10 and node 2 is supposed to be the Charleston computer but due to technical difficulties we are using the virtual machine 192.168.30.2 which acts similar to the above. This lab makes use of the spoofing machine too, that creates packets and sends them over to the network. It has the IP address 192.168.10.111.

On the wireshark interface we use the enp20 interface and open it through the sniffer machine. This immediately opens up and starts capturing the traffic.

We first connect to the sniffer and spoof machines by giving the commands:

ssh sniffer ssh spoof

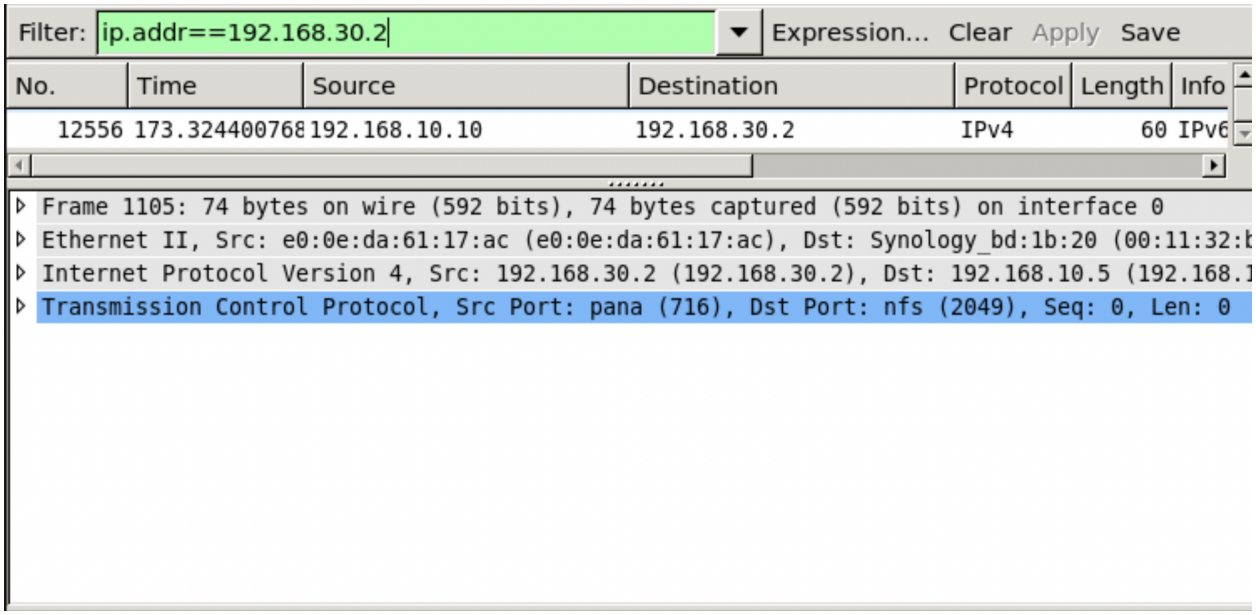
Now going to the spoof machine we execute the nmap command in order to discover all active hosts in the subnet.
$ nmap -sn 192.168.10.0/24

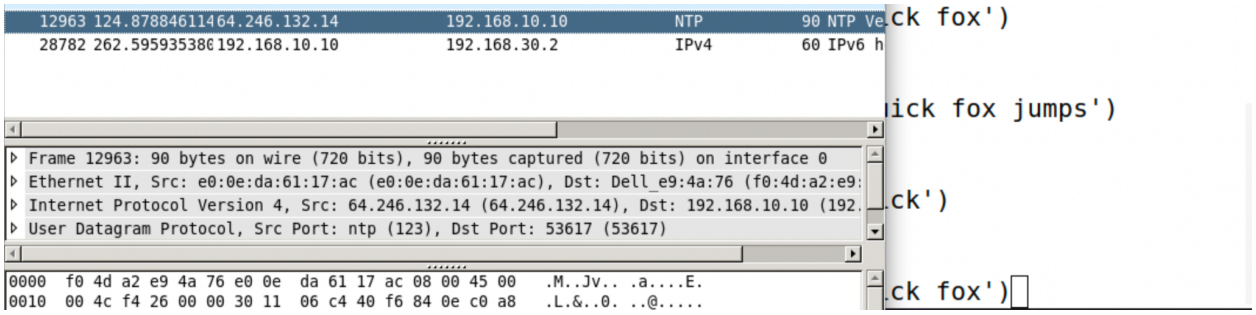
```
root@cnc:~# nmap -sn 192.168.10.0/24

Starting Nmap 7.01 ( https://nmap.org ) at 2022-03-11 09:58 EST
Nmap scan report for 192.168.10.1
Host is up (0.00040s latency).
MAC Address: E0:0E:DA:61:17:AC (Unknown)
Nmap scan report for 192.168.10.2
Host is up (0.00016s latency).
MAC Address: 78:45:C4:FE:8E:63 (Dell)
Nmap scan report for 192.168.10.3
Host is up (0.00017s latency).
MAC Address: 5C:F9:DD:6C:D4:54 (Dell)
Nmap scan report for 192.168.10.5
Host is up (0.00016s latency).
MAC Address: 00:11:32:BD:1B:1F (Synology Incorporated)
Nmap scan report for 192.168.10.6
```

The above screenshot shows the active hosts available. Now that we know these hosts we can start spoofing the packets. For this we open the spoof terminal and run the scapy command to open up an interactive session. For spoofing the below command is used:

$send(IP(dst='address', src='address')/'message')

For the first spoofing packet I sent a packet from a clemson machine(192.168.10.10) to the virtual machine(192.168.30.2) and with the message 'quick fox' using the above command. Now opening wireshark, if we type in the filter as the destination IP address as ip.addr==192.168.30.2 we see the packet has been captured by it.



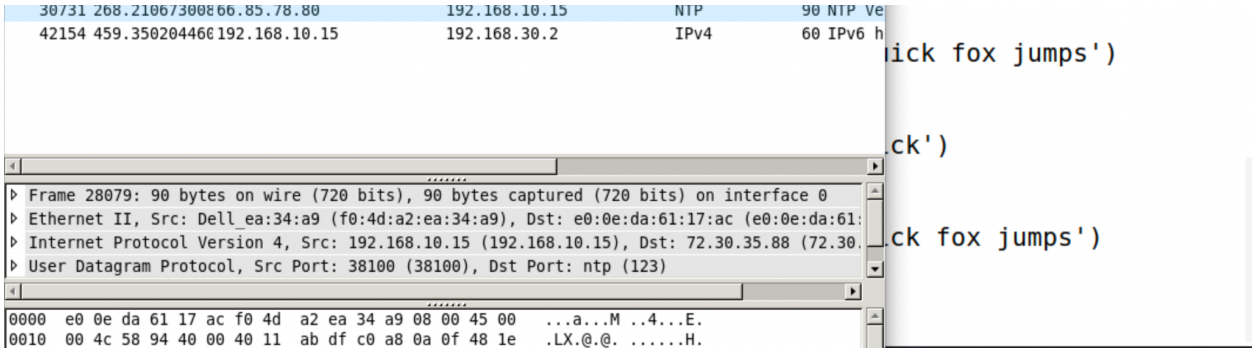


In the above screenshot we see that the ip address of the source is the spoofed address and not the ip address of the machine we sent it from which is 192.168.10.111. This means that the attack has been successful,

Similarly, I spoof a second address 192.168.10.150 with the message ‘quick fox jumps’ as below.

```
.
Sent 1 packets.
>>> send(IP(dst='192.168.30.2', src='192.168.10.15')/'quick fox jumps')
```

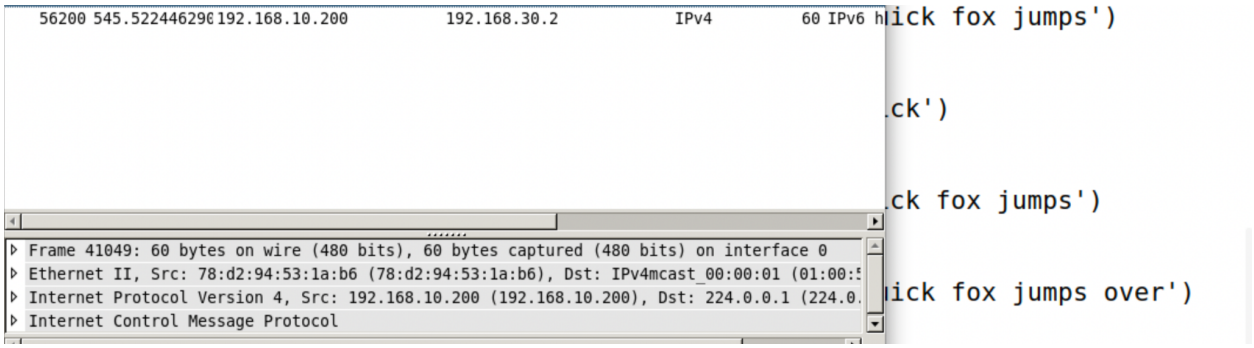
Below we see the success of the attack as we can see the spoofed address instead of the original address.



Now the third address that I spoofed is 192.168.10.200 with the message ‘quick fox jumps over’.

```
Sent 1 packets.
>>> send(IP(dst='192.168.30.2', src='192.168.10.200')/'quick fox jumps over')
.
Sent 1 packets.
```

The success of the above command can be seen in the below screenshot.



After successful spoofing attempts, I tried to intercept the traffic to see if there were any spoofing attempts. On wireshark, I opened the enp20 filter and captured the traffic for a few minutes and stopped. I waited for some time but I couldn't find any spoofing attempts.