Homework 4

CSCE-465-500

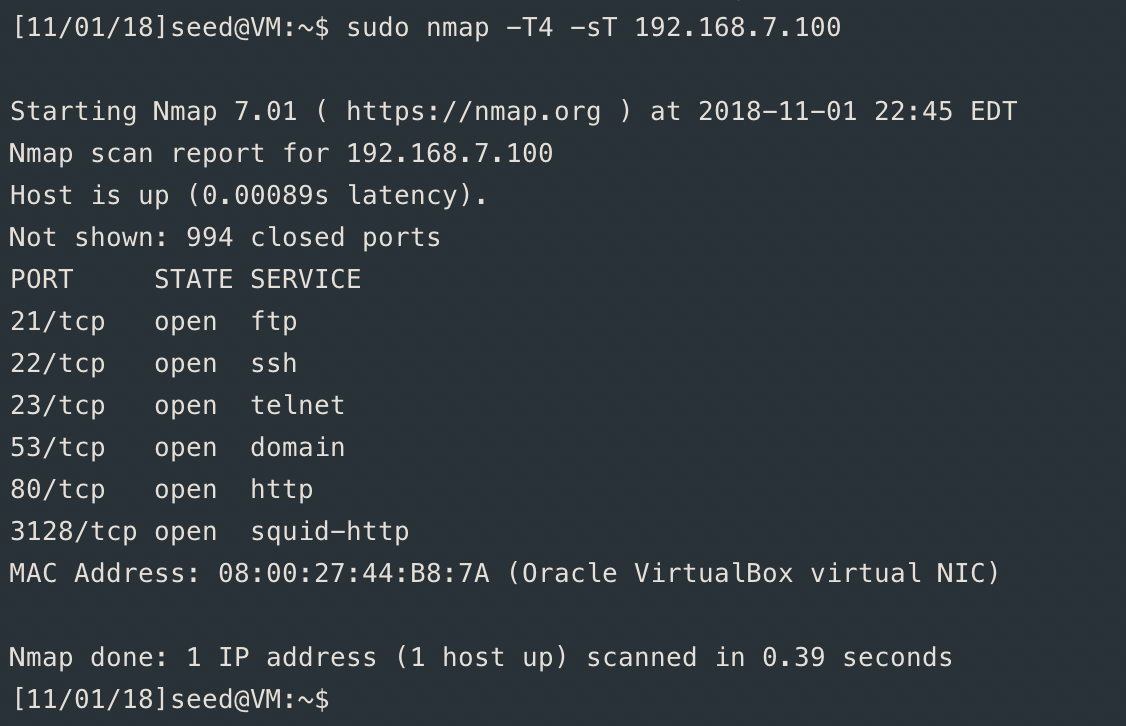
November 1, 2018

Joseph Martinsen

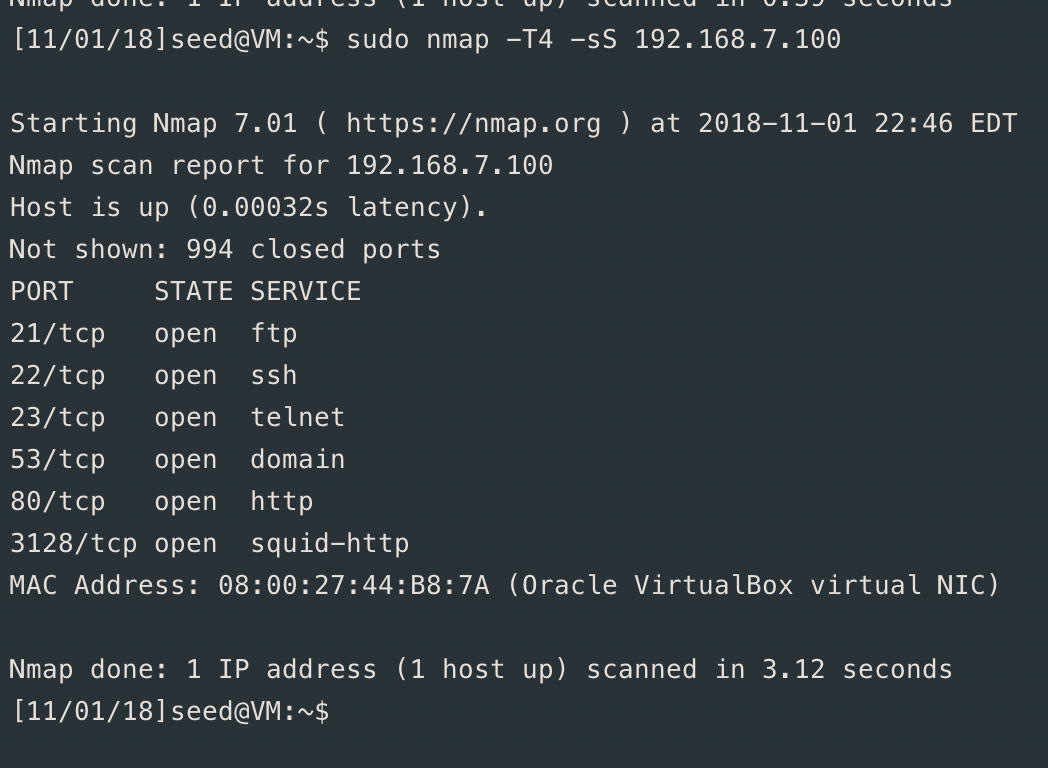
# Task 1 Surveillance Techniques

## Port Scanning

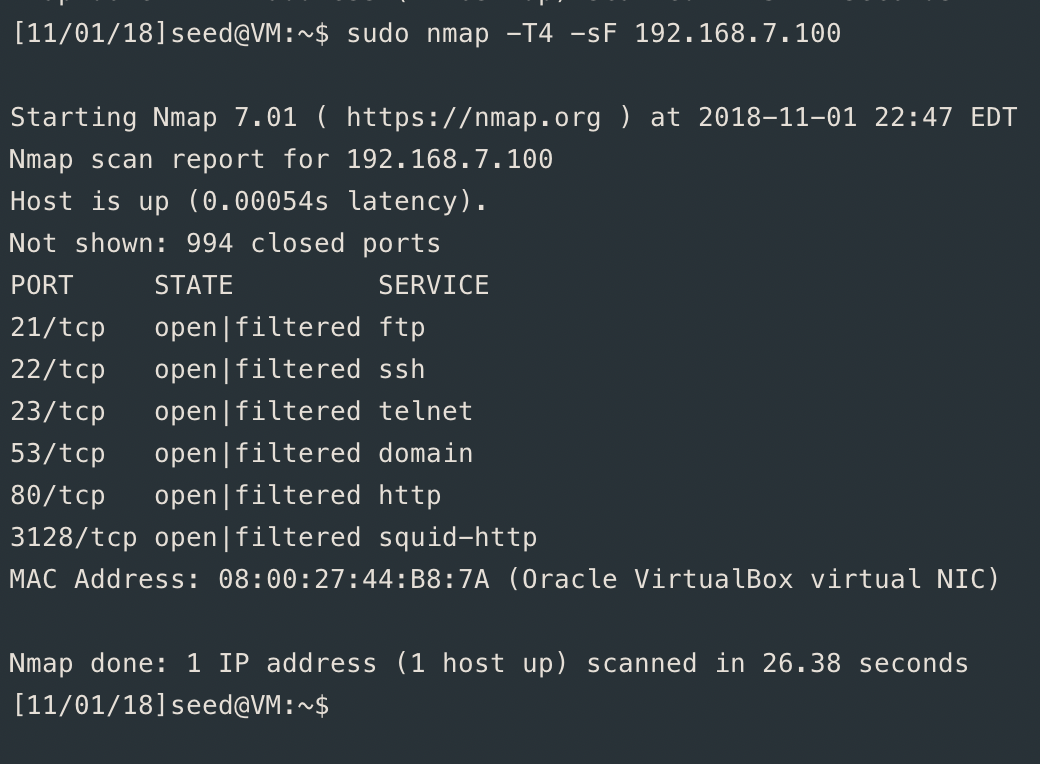
### Tcp Connect Scan



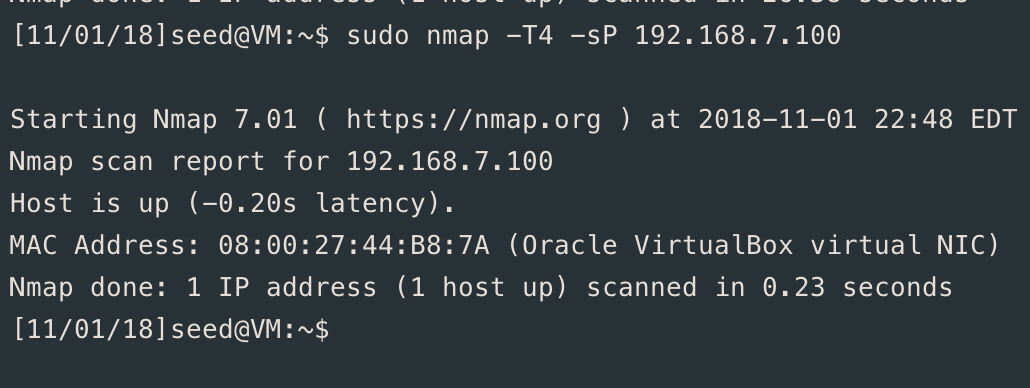
### SYN Stealth Scan



### FIN Scan



### Ping Scan

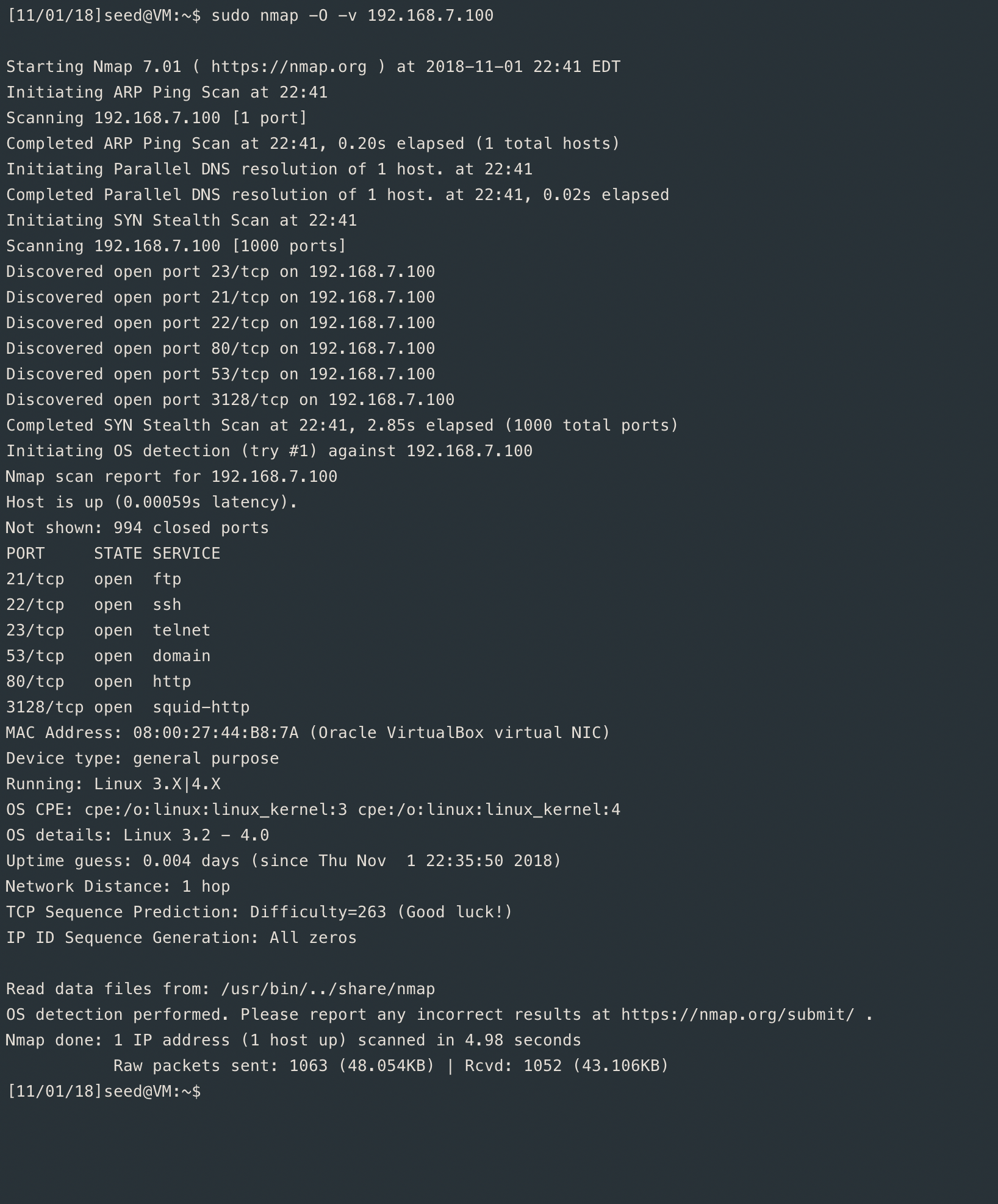


What I observed were that the port ftp, ssh, telnet, domain, and squid-http were picked up. These are ports that’s match to what they are standard for their protocol. Nmap was able to observe that the mac address was a virtual NIC. 6 ports were found and 994 were closed totaling in 1000 ports visited.

Things I noticed were the open ports like telnet, ftp, ssh, ttp, domain, and squid-http. These are all fairly common ports that you would find on a regular computer. Nmap scanned the first 1000 ports, most of which are reserved for system ports. Nmap also sensed that the IP was in a virtual machine, specifically VirtualBox and sensed the range of my kernel version. Interestingly enough it also knew the uptime of the computer, and how far away the virtual machine is in network distance

## Fingerprint

### nmap -O -v



# Task 2: Arp Cache Poising

Router IP: 192.168.7.1

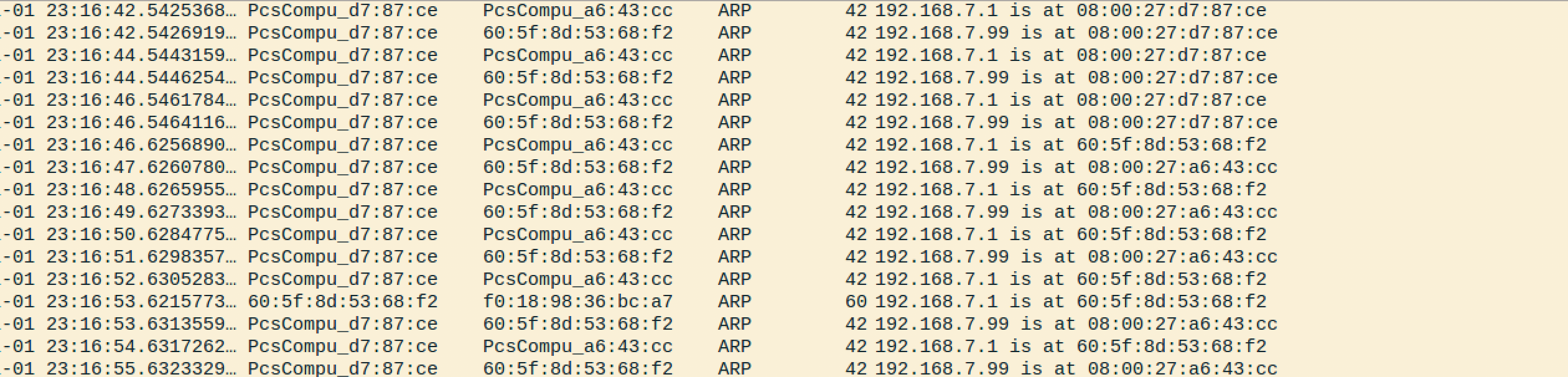
Router Mac: 60:5f:8d:53:68:f2

Victim IP: 192.168.7.99

Network Interface: enp0s3

Below you can see that my mac address is saying it 8:00:27:d7:87:ce while arp attack is running.

The mac address changes back to the real address: 60:5f:8d:53:68:f2



The following command was ran on the attackin machine: sudo arpspoof -i enp0s3 -t 192.168.7.99 -r 192.168.7.1

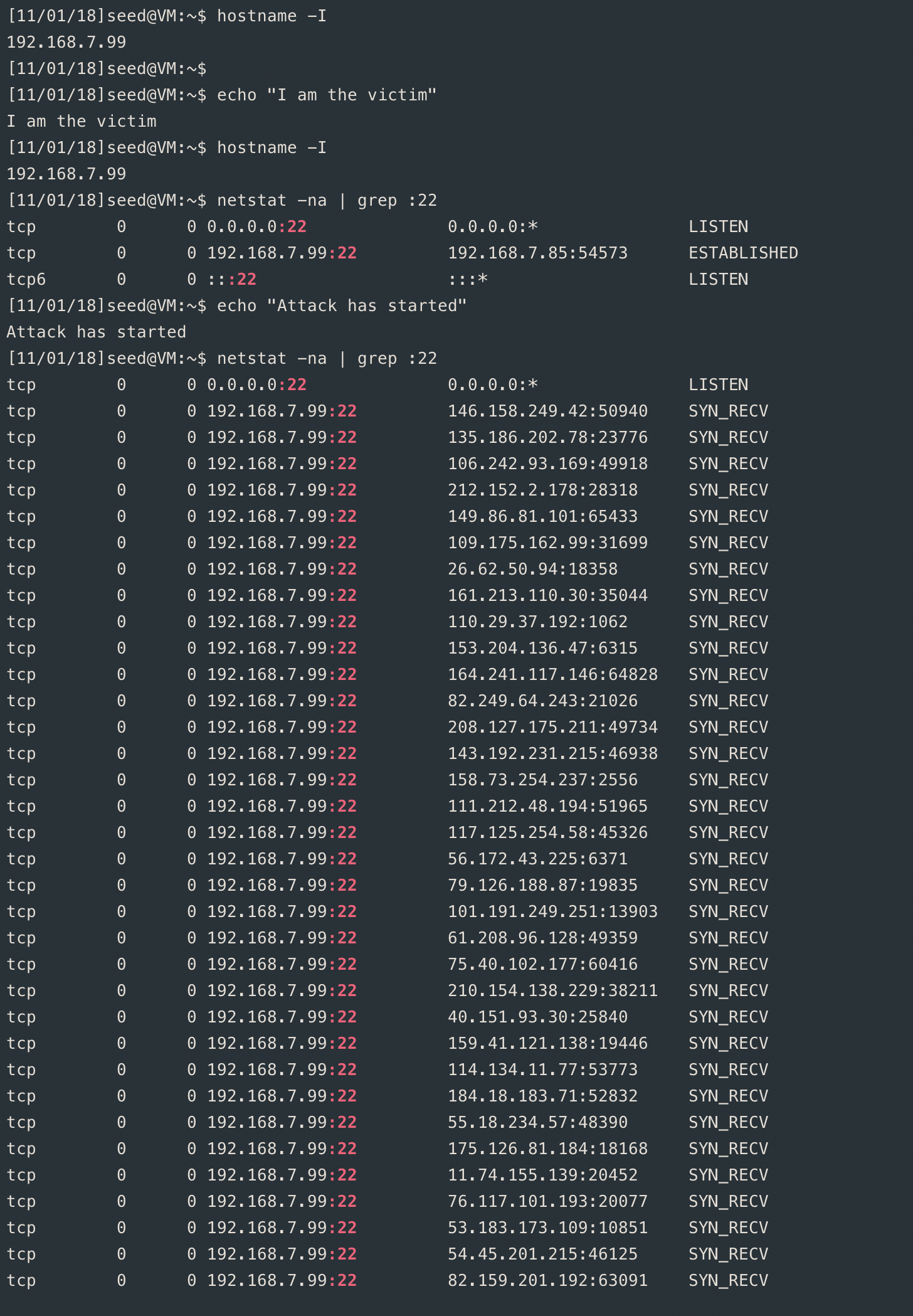
# Task 3: SYN Flooding Attack

First I turned off SYN on the victim machine. With a 3rd machine I was able to ssh into the victim. Victim attack had 3 usages of port 22. Once the attack started, the 3rd machine could no longer ssh into the victim attack. There were 100 instances of port 22 on the victim machine with netstat -na.

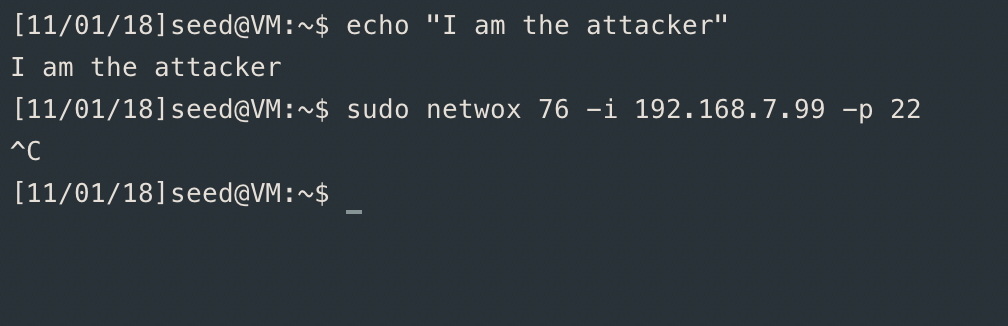
I was able to confirm the attack was working because my 3rd machine could not ssh into the victim while the attack was happening.

I was not able to complete the attack once the SYN coockie was set back to 1 on the victim machine.

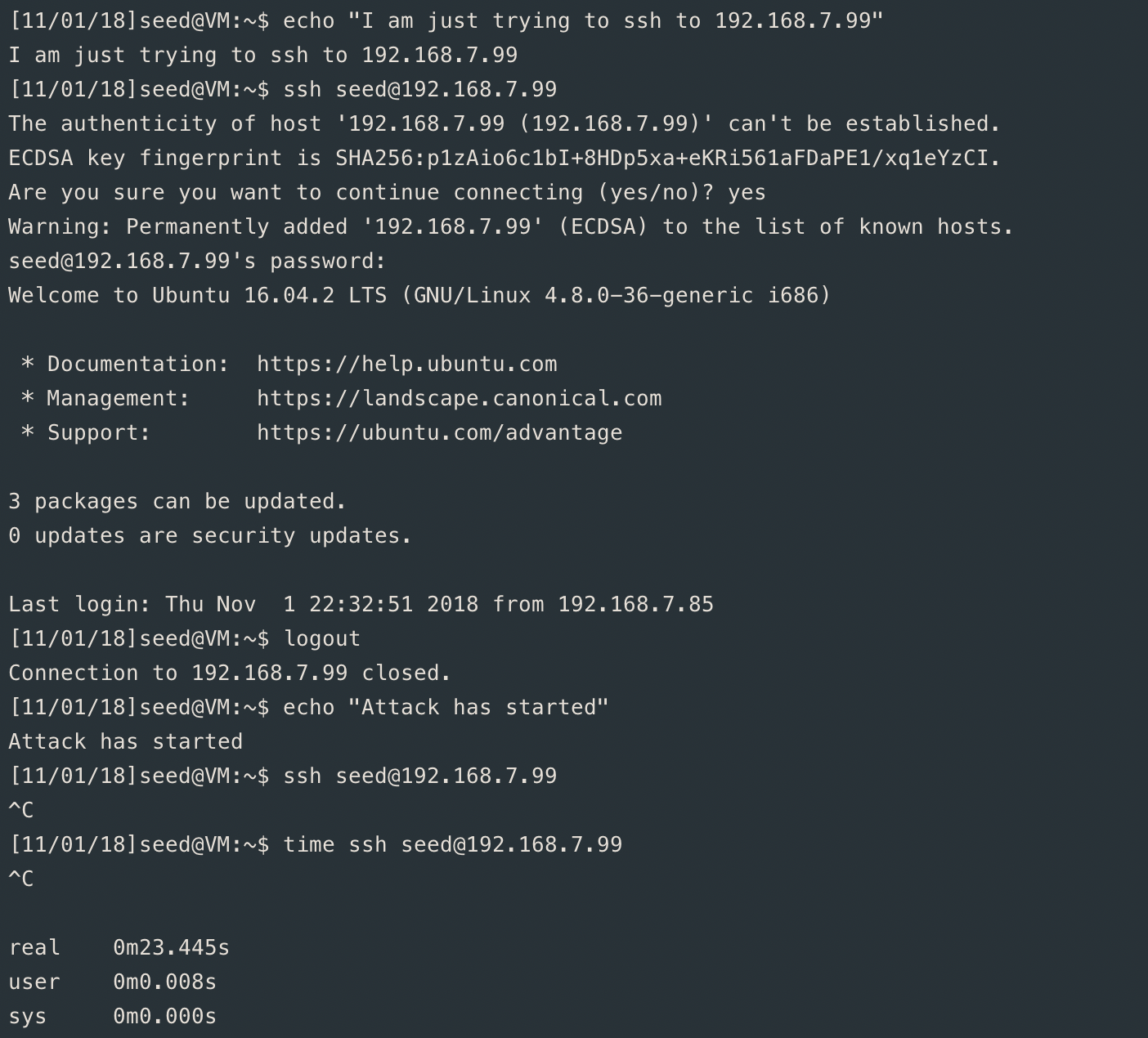
## Victim Machine



## Attacking Machine



## 3rd Party



# Task 4



On the attacker the following was done,

# echo 1 > /proc/sys/net/ipv4/ip\_forward

# sudo arpspoof -i enp0s3 -t 192.168.7.99 192.168.7.98

sudo arpspoof -i enp0s3 -t 192.168.7.98 192.168.7.99

# apt install python-scapy

# scapy

>>> i = IP()

>>> t = TCP()

>>> i.dst = "192.168.7.99"

>>> i.src = "192.168.7.99"

>>> i.src = "192.168.7.98"

>>> t.dport = 38596

>>> t.sport = 23

>>> t.seq = 139280999

>>> t.ack = 277043729

>>> t.window = 0

>>> send(i/t)

.

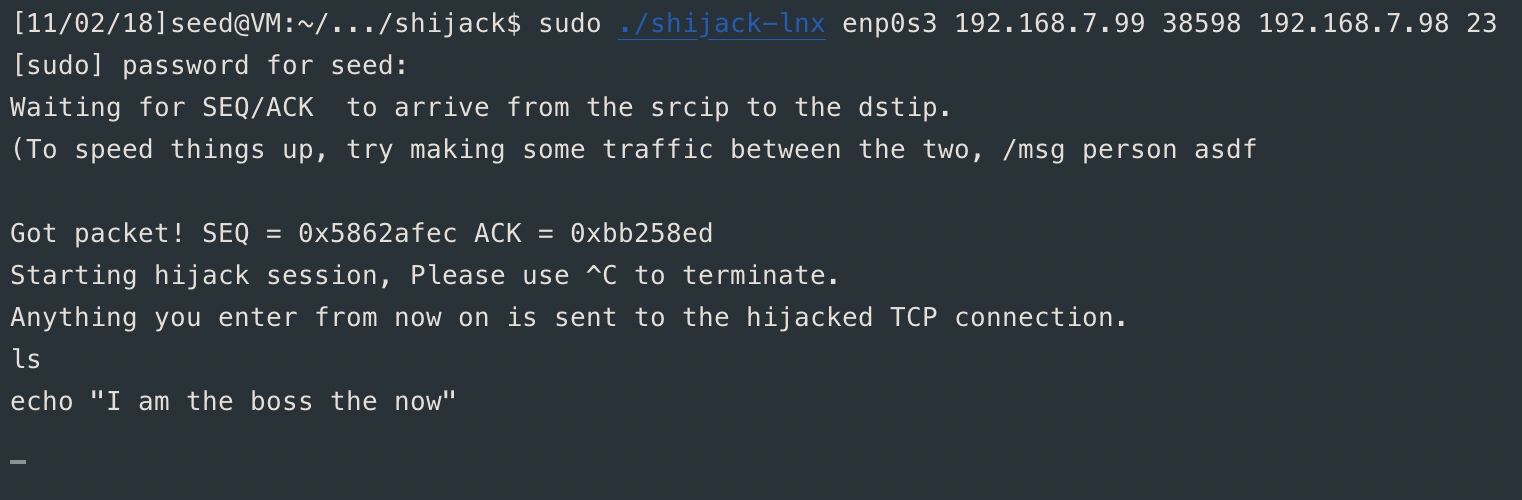
Sent 1 packets.

The connection was then closed between the two machines. Similar behavior was experienced with ssh.

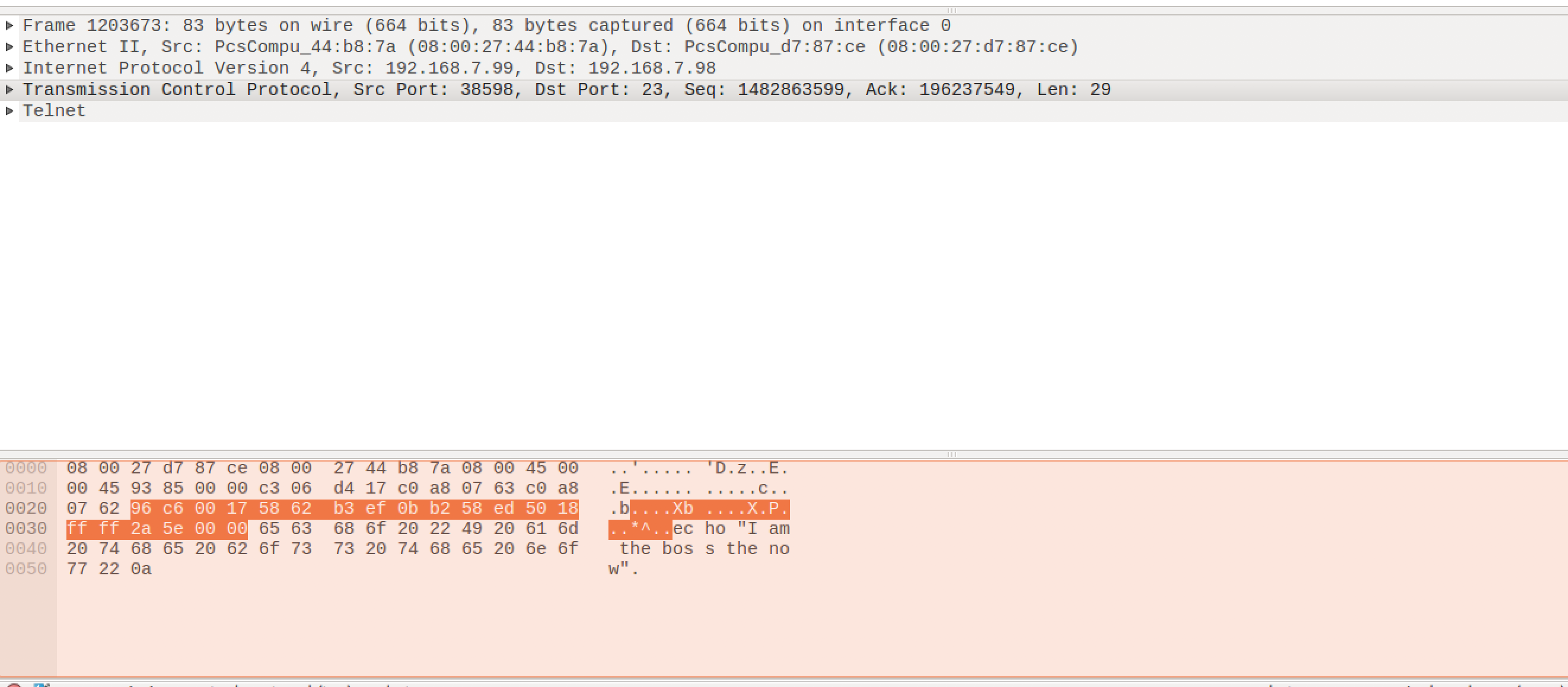
To defend agains this, using some sort of cryptography to encrypt and decrypt the packets will slow down the process but ensure that a unwanted RA won’t be sent maliciously.

# Task 5

ON the victim machine I installed shijack and poisoned the two victim machines that were telneted to each other. Once hijacked I was able to send data. The victim that was hijacked was not able to write anything to the terminal.



The response to this was able to be seen in wireshark.



In order to not experience this you can encrypt the data or add headers to the communication so that the requests can be validated.