

TCP/IP Attacks

Klas Mannberg
klaman-8@student.ltu.se



24 September 2020

Contents

1	Lab Tasks	1
1.1	Task 1: SYN Flooding Attack	1
1.2	Task 2: TCP RST Attacks on telnet and ssh Connections	1
1.3	Task 3: TCP RST Attacks on Video Streaming Applications	1
1.4	Task 4: TCP Session Hijacking	2
1.5	Task 5: Creating Reverse Shell using TCP Session Hijacking	2
2	Images	2

1 Lab Tasks

1.1 Task 1: SYN Flooding Attack

Spam start 3-way TCP handshakes on transport layer (TCL handshake on the app layer). Floods with requests asking for replies.

We create three virtual machines (hosts): Client/Attacker/Server.

After I retrieved the IP address of my server VM box, I launched a synflood attack using netwox 76 from the attacker VM. The result was many more half-open connections on the server. The connections before the attack are shown in image 1 and after in image 2.

When the SYN flood was running from the attacker, the client could not telnet connect to the server anymore. See image 3. When I turned on SYN cookies on the server it no longer blocked the clients connection via telnet.

The cookies prevent the server from sending back the response SYN+ACK back until it receives a response ACK which an attacker never sends out with a normal SYN flood. This however opens a new attack to send many ACK's so the server gets overloaded anyway. To prevent this the server calculates the hash from the previous SYN+ACK and see if its a response from someone it has sent an SYN+ACK to, if its not it can just discard it and not waste resources.

1.2 Task 2: TCP RST Attacks on telnet and ssh Connections

Using the netwox 78 command with the filter parameter, we send out one RST packet for each packet from server host with the command. `sudo netwox 78 --filter "src host 10.0.2.9"` The result is the telnet (TCP) connection closing, as seen in image 4.

The values were retrieved using wireshark on the attacker VMbox.

1.3 Task 3: TCP RST Attacks on Video Streaming Applications

Doing Task 2 but targeting the TCP connection of a video streaming service, *Youtube*. The result is the video stops downloading and only the downloaded buffer remains. Eventually this leads to an error response from the video player.

1.4 Task 4: TCP Session Hijacking

I ran wireshark to find the appropriate parameters to make sure my packet is accepted. But also I made sure to be the next packet by assigning my packet the calculated next sequence number, given by wireshark. After I got the needed values for customizing our packet, I created the program shown in in image 5.

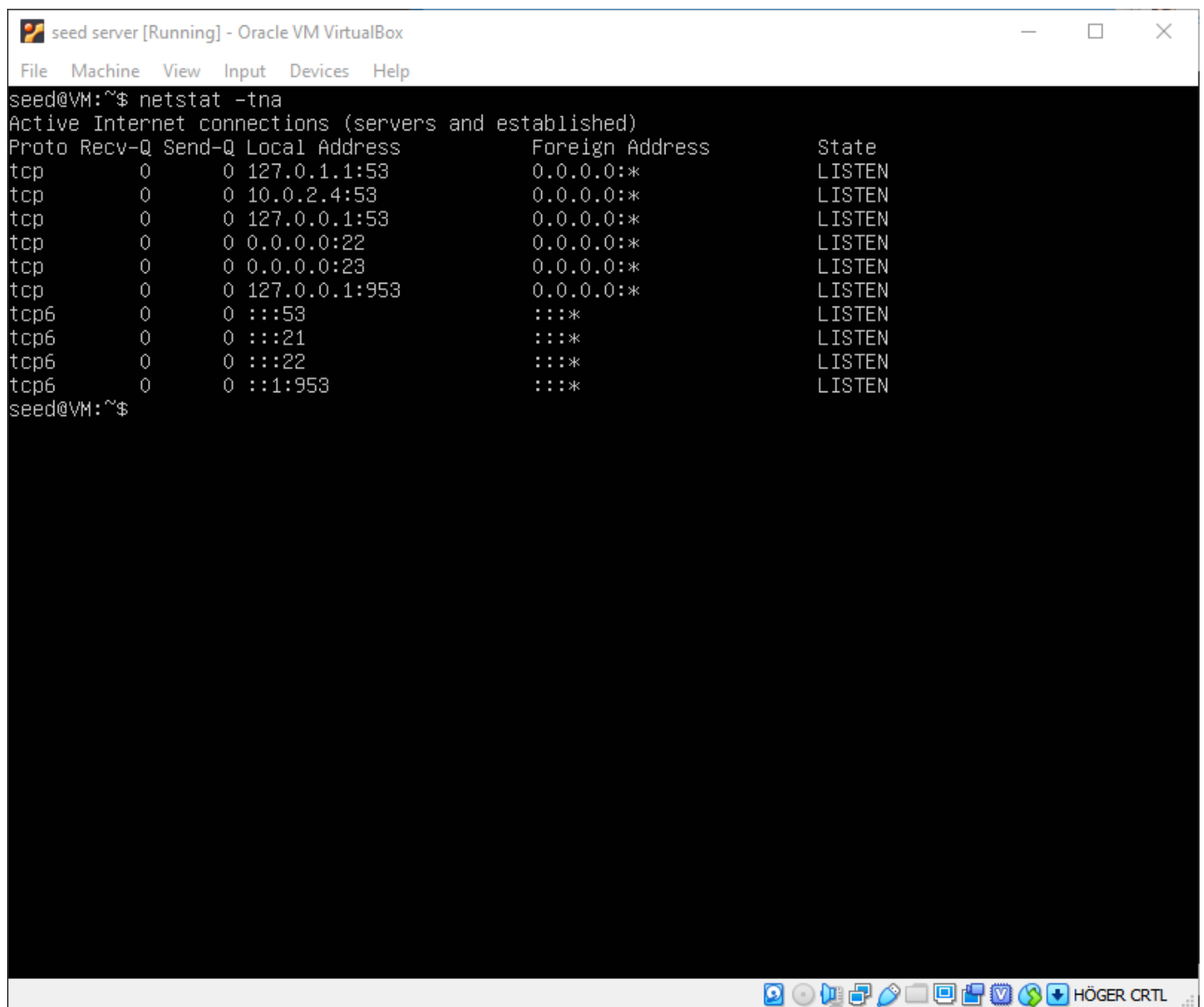
In the image 6 you can see the server accepting my packet with the custom data!

1.5 Task 5: Creating Reverse Shell using TCP Session Hijacking

In this taks we repeat task 4 but with a command that launches a shell session for us! To accomplish this I use the same code as in Task 4 but change the data inside into a command that launches a shell using a port that I can connect to. After I retrieved the necessary information from wireshark I got the server to run the reverse shell command via our spoofed packets! See image 7 for the attacker perspective. The client connection is frozen while we have access to SHELL.

2 Images

Figure 1: Task 1: Connections to server before SYN flood

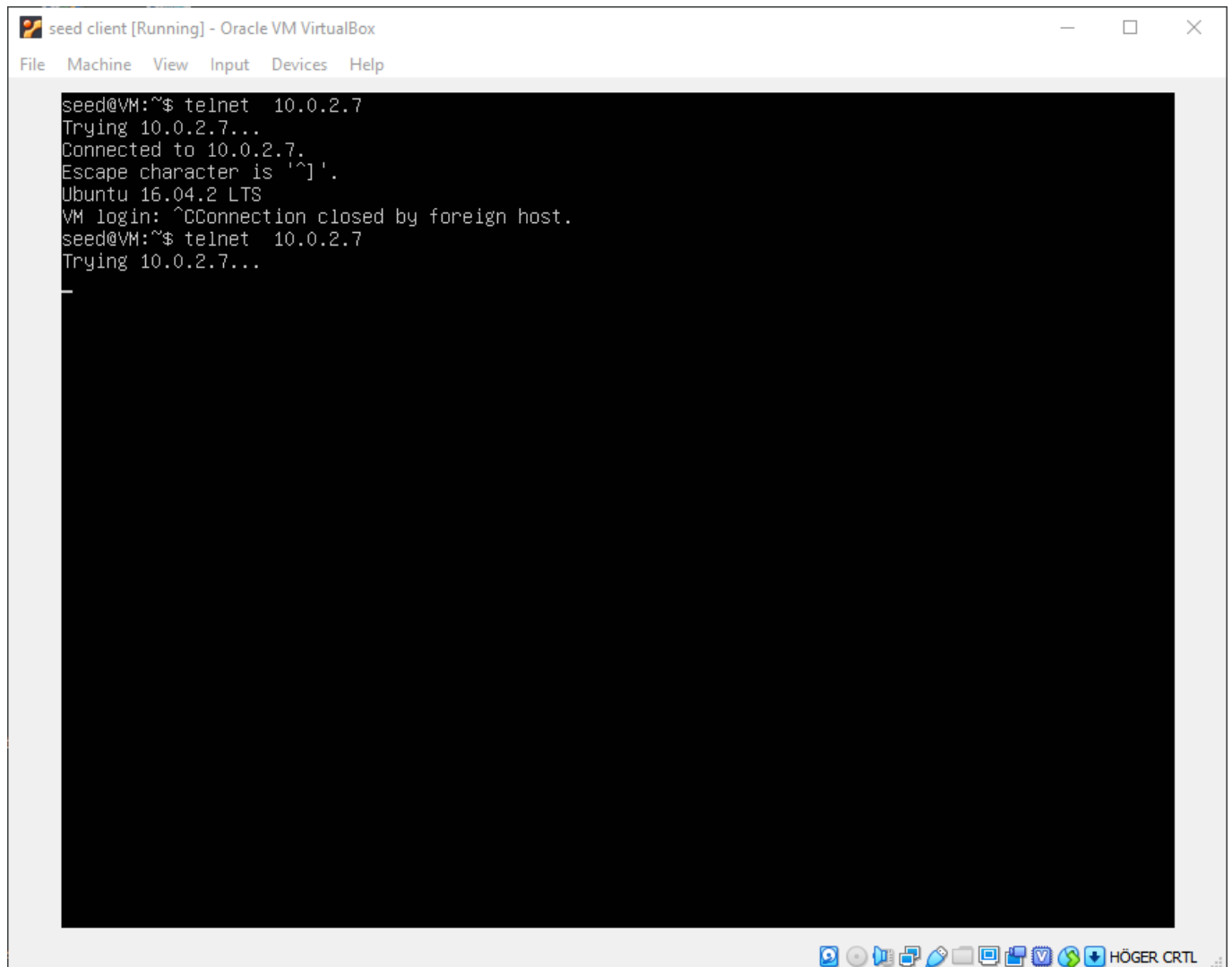


```
seed server [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
seed@VM:~$ netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53             0.0.0.0:*               LISTEN
tcp        0      0 10.0.2.4:53              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:53             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22               0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23               0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:953            0.0.0.0:*               LISTEN
tcp6       0      0 :::53                    :::*                    LISTEN
tcp6       0      0 :::21                    :::*                    LISTEN
tcp6       0      0 :::22                    :::*                    LISTEN
tcp6       0      0 :::1:953                  :::*                    LISTEN
seed@VM:~$
```

Figure 2: Task 1: Connections to server after SYN flood

seed server [Running] - Oracle VM VirtualBox						
File	Machine	View	Input	Devices	Help	
tcp	0	0	10.0.2.4:23	254.96.34.222:58590	SYN_RECV	
tcp	0	0	10.0.2.4:23	251.33.50.150:5531	SYN_RECV	
tcp	0	0	10.0.2.4:23	254.220.234.0:51490	SYN_RECV	
tcp	0	0	10.0.2.4:23	245.36.182.81:21836	SYN_RECV	
tcp	0	0	10.0.2.4:23	243.185.124.204:13084	SYN_RECV	
tcp	0	0	10.0.2.4:23	243.32.114.29:28157	SYN_RECV	
tcp	0	0	10.0.2.4:23	245.199.135.172:45967	SYN_RECV	
tcp	0	0	10.0.2.4:23	245.129.253.160:50926	SYN_RECV	
tcp	0	0	10.0.2.4:23	246.78.68.218:51052	SYN_RECV	
tcp	0	0	10.0.2.4:23	250.110.166.54:1324	SYN_RECV	
tcp	0	0	10.0.2.4:23	240.202.250.248:50956	SYN_RECV	
tcp	0	0	10.0.2.4:23	246.92.137.130:3178	SYN_RECV	
tcp	0	0	10.0.2.4:23	251.116.111.185:47365	SYN_RECV	
tcp	0	0	10.0.2.4:23	251.169.251.208:18134	SYN_RECV	
tcp	0	0	10.0.2.4:23	250.229.167.113:22844	SYN_RECV	
tcp	0	0	10.0.2.4:23	242.15.23.110:5293	SYN_RECV	
tcp	0	0	10.0.2.4:23	240.171.178.197:24430	SYN_RECV	
tcp	0	0	10.0.2.4:23	241.145.120.59:21813	SYN_RECV	
tcp	0	0	10.0.2.4:23	253.210.183.19:37045	SYN_RECV	
tcp	0	0	10.0.2.4:23	249.233.91.223:6487	SYN_RECV	
tcp	0	0	10.0.2.4:23	243.41.234.55:20600	SYN_RECV	
tcp	0	0	10.0.2.4:23	251.194.36.96:18230	SYN_RECV	
tcp	0	0	10.0.2.4:23	243.49.61.120:50044	SYN_RECV	
tcp	0	0	10.0.2.4:23	255.216.10.39:5559	SYN_RECV	
tcp	0	0	10.0.2.4:23	253.149.83.223:13171	SYN_RECV	
tcp	0	0	10.0.2.4:23	244.234.0.251:14263	SYN_RECV	
tcp	0	0	10.0.2.4:23	255.23.16.100:41371	SYN_RECV	
tcp	0	0	10.0.2.4:23	250.128.70.134:11903	SYN_RECV	
tcp	0	0	10.0.2.4:23	246.103.153.236:34548	SYN_RECV	
tcp	0	0	10.0.2.4:23	240.202.23.175:56297	SYN_RECV	
tcp	0	0	10.0.2.4:23	246.110.176.47:1285	SYN_RECV	
tcp	0	0	10.0.2.4:23	252.248.9.88:43477	SYN_RECV	
tcp6	0	0	:::53	:::*	LISTEN	
tcp6	0	0	:::21	:::*	LISTEN	
tcp6	0	0	:::22	:::*	LISTEN	
tcp6	0	0	:::1953	:::*	LISTEN	
seed@VM:~\$ _						

Figure 3: Task 1: Client tries to connect to server under SYN flood attack



```
seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
Connected to 10.0.2.7.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: ^CConnection closed by foreign host.
seed@VM:~$ telnet 10.0.2.7
Trying 10.0.2.7...
_
```

Figure 4: Task 2: Disrupted telnet (TCP) connection!

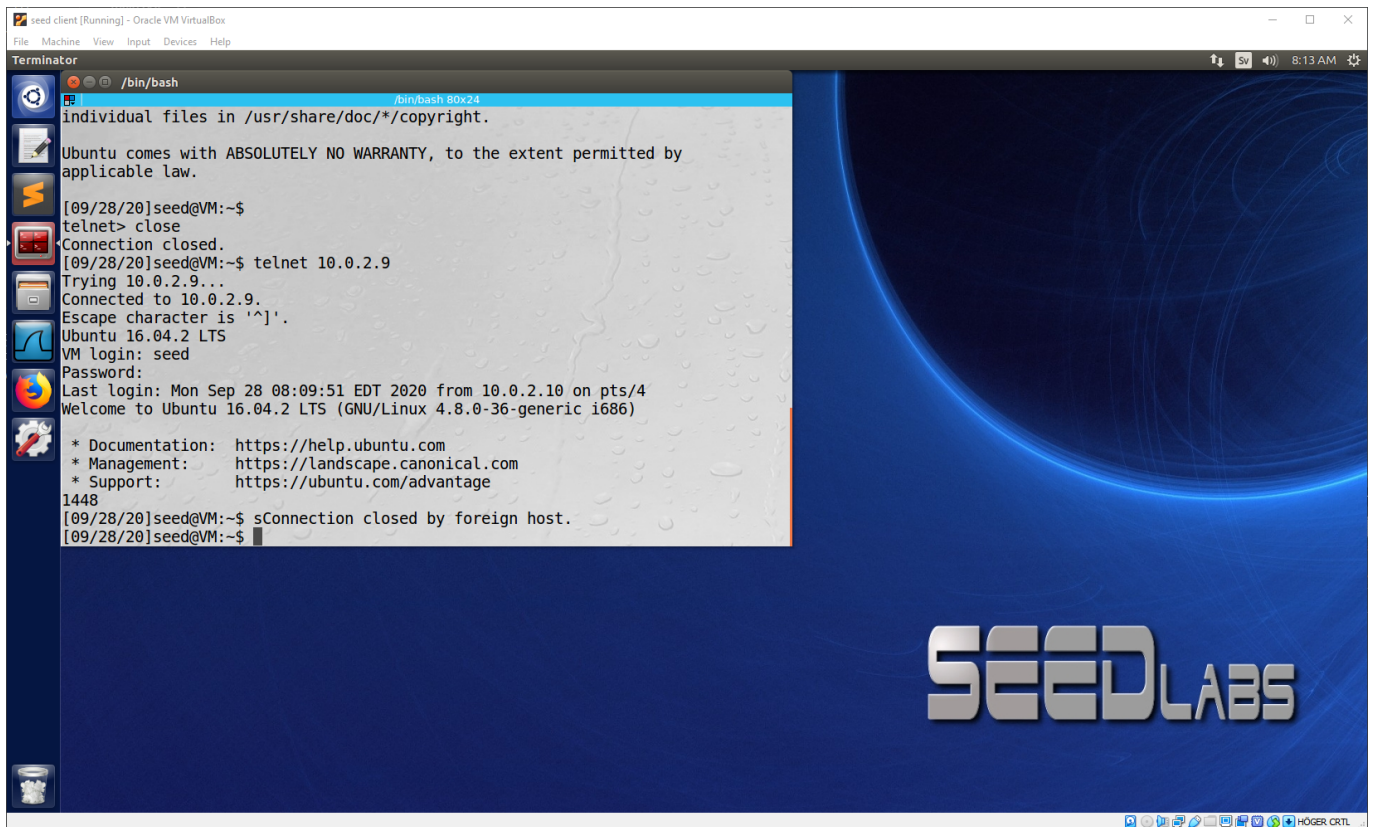


Figure 5: Task 4: Source Code

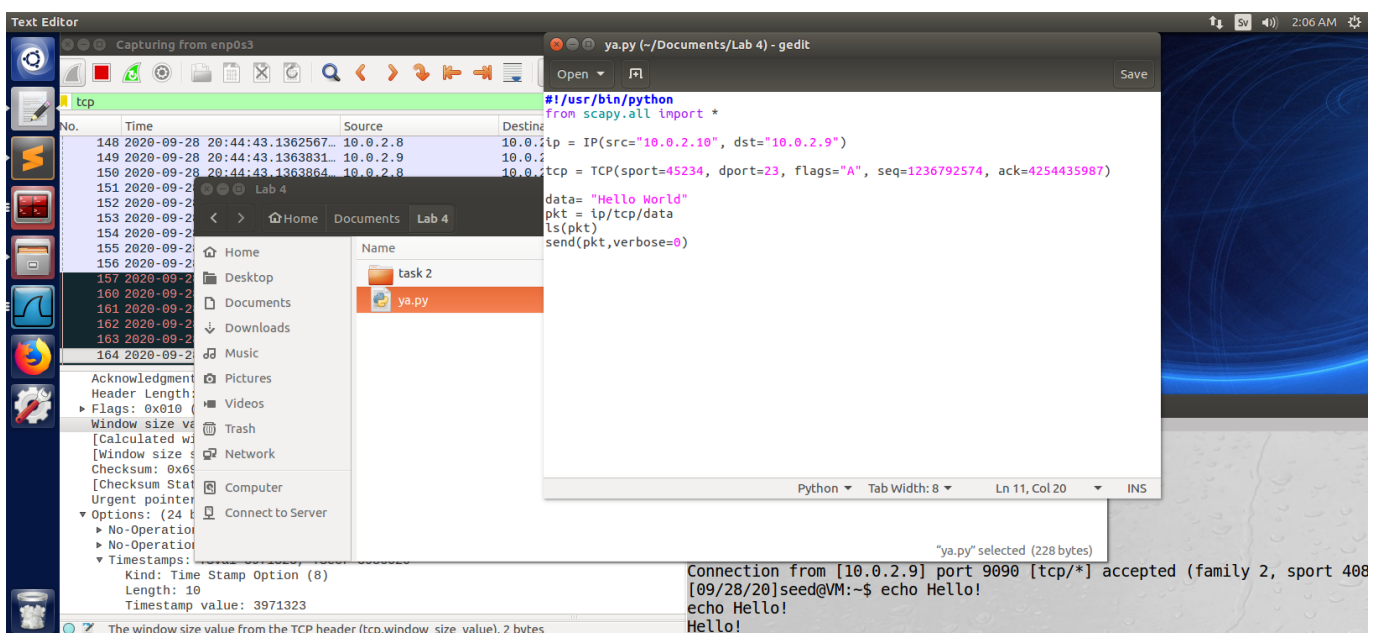


Figure 6: Task 4: Sent data "Hello World" via TCP connection!

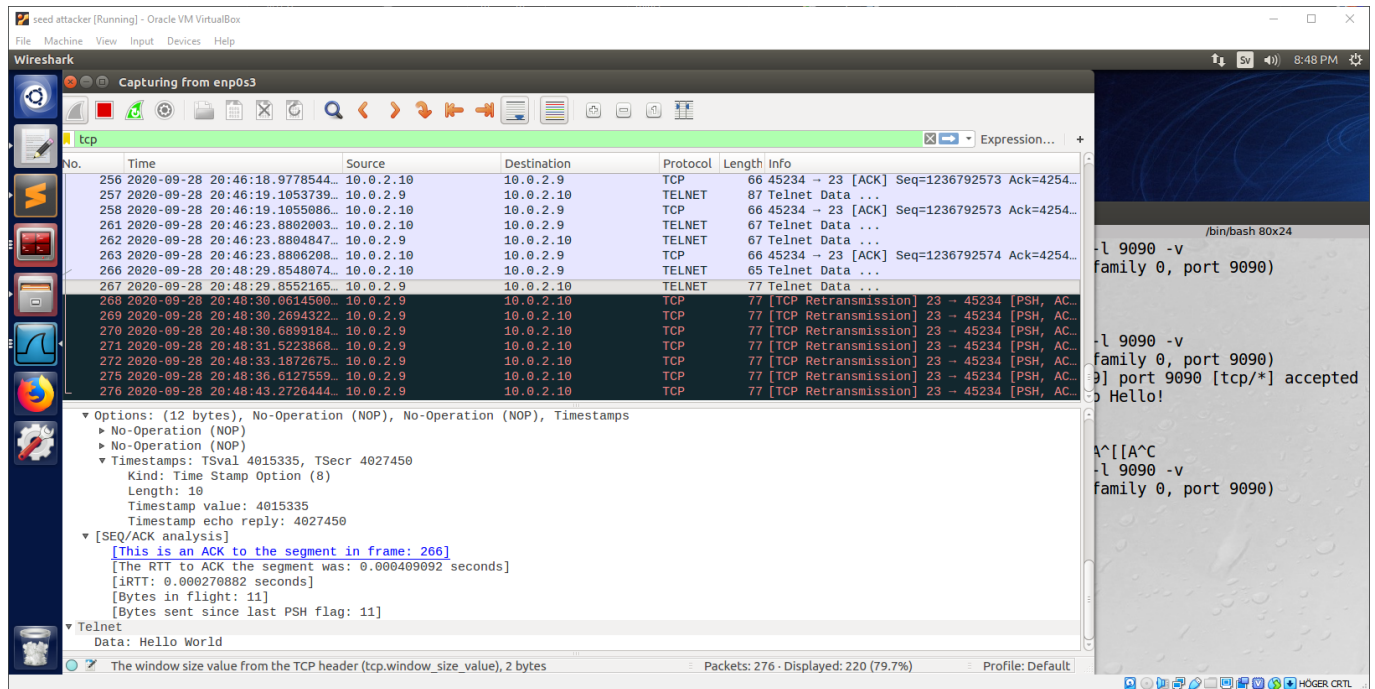


Figure 7: Task 5: Got the server to run the reverse shell command

